A Framework for Classification of Data Stream Application in Vehicular Network Computing

Ling Yu[1], Yang Gao[2], Yu Zhang[2], and Li Guo[2(✉)]

[1] Network Information Center, Dalian Polytechnic University, Dalian 116000, Liaoning, China
[2] School of Information Science and Engineering, Dalian Polytechnic University, Dalian 116000, Liaoning, China
guoli@dlpu.edu.cn

Abstract. Due to the fast developing of intelligent vehicles and the dynamic sensor network, a huge amount of data streams are generated continuously. How to manage these data stream will play a very important role in the development of next-generation intelligent vehicular networks. Data stream classification problem poses many new challenges to the data mining area. One of the most basic challenges is how to make a tradeoff between data stream generating and classifying process. In this paper, we address this problem using a new framework and a new classification method, namely Extreme Support Vector Machine (ESVM), for large-scale data stream classification for specialty vehicle network. Moreover, a new forgetting strategy is designed for the proposed TFS-PESVM model, which employs the residual weight matrix to optimal residual value and improves the effectiveness of new data samples. The experimental results show that the proposed data stream classification model can improve the speed of classification process and less affected by noise interference.

Keywords: Data stream classification · ESVM · Incremental learning · MapReduce · Vehicular network

1 Introduction

The specialty vehicles industry has recently shifted from developing advanced vehicles to concentrating on safety and functionality, which stimulates the development of new intelligent vehicles with advanced assistance systems and even autonomous driving. However, the current technology development is still far from the practical requirements of intelligent vehicles. This is because the existing control approaches are mainly based on traditional sensors and cameras employed in the specialty vehicle that hinder the development of intelligent vehicles. Thanks to wireless communication networks, vehicles can be connected and can communicate with each other. Thus, the control of vehicles will become more reliable with the communication between vehicles. Therefore, managing the data stream obtained from the sensor network of vehicles has become one of the key research issues in the field of specialty vehicle control.

Currently, data stream mining has become a new research topic in the data mining community. The large-scale and real-time classification nature of data streams require

J. Jin et al. (Eds.): GreeNets 2019, LNICST 282, pp. 57–67, 2019.
https://doi.org/10.1007/978-3-030-21730-3_6

an efficient and effective method that is significantly different from the traditional static data mining methods [1]. However, a data stream is a fast and continuous phenomenon, which can be assumed to have infinite length. Therefore, it is impractical to store and use all the historical data for classifier training [2].

For the problem presented in the area of data stream mining for a specialty vehicle network, the most obvious alternative is the incremental learning method. Several incremental learning based data stream classification technologies have been proposed to address this problem. Also, concept drift occurs in the stream when the underlying concepts of the stream change over time, which means that the classifier needs to be updated in real time [3]. However, the existing data stream classification methods are still weak for some practical applications.

Currently, several frameworks have been introduced into the data stream classification area, e.g., MapReduce. For the parallel nature of MapReduce, most researchers are studying how to modify the traditional classification method into a MapReduce style [4, 5], and now some related methods have been proposed. A MapReduce framework based on the K-NN algorithm was proposed [6], which decreases the running time of the K-NN algorithm significantly and improves the ability of multi-classification. However, its inner problems, such as choosing a suitable distance function and related parameters, are still unsolved. An analogous method was proposed [7] that combines a decision tree with the MapReduce framework for classification. To employ the Support Vector Machine (SVM) model in MapReduce, a new minimal sequence optimization method [8] was proposed to improve the speed of computing quadratic programming problems. A parallel method [9] was proposed to compute hyperplane bias, which would accelerate the best hyperplane selection process. Based on these methods, for the concept drift problem, Zhao et al. proposed an incremental learning based SVM model [10], but in a parallel environment, the performance of this method is a little weak. Although the methods above are trying to modify the traditional classification methods in parallel, their performance still has much potential for some real applications [11].

In this paper, a new data stream classification method, called TFS-PESVM, is proposed which employs extreme support vector machine (ESVM) as a classifier [12], MapReduce as a parallel framework, an incremental learning method to track concept drift and classifier updates, and a forgetting strategy to improve the effectiveness of new data. We apply our method to some synthetic data streams for specialty vehicle networks, which includes four data sets. Also, the proposed model is also compared with two distinct data stream models. The experiment on big stream data is used to test the proposed method and the framework.

2 Framework of Data Stream Classification

Due to the high mobility of vehicles and the dynamic change of the network topology, it is difficult to provide satisfying data analysis services for vehicles through a single network. Hence, providing heterogeneous vehicular networks may support the communication, control and computation requirements of intelligent vehicles well. However, there are multiple critical challenges in designing efficient and flexible data stream

mining methods since this requires the joint design and optimization of both analysis, computing and data mining model.

In this paper, we develop a MapReduce based framework, shown in Fig. 1. Suppose there are several vehicles in the vehicular network. The data stream is generated from the sensors mounted on the vehicle. There are two basic structures for analyzing the data stream named "Key" and "Value", which are correlative. The basic process contains two steps. Firstly, in the Map step, the mission is divided into some Map missions. Each map mission relies on a sensor. In the mapping step, the process is distributed to distinct computers. Secondly, in the Reduce step, the mission is grouped with a vehicle according to the "Key" and "Value" of the data stream. Distinct groups are distributed to different Reduce missions, and finally, the result is collected and outputted according to "Key" and "Value". Then, we can obtain the overview of the whole vehicle network.

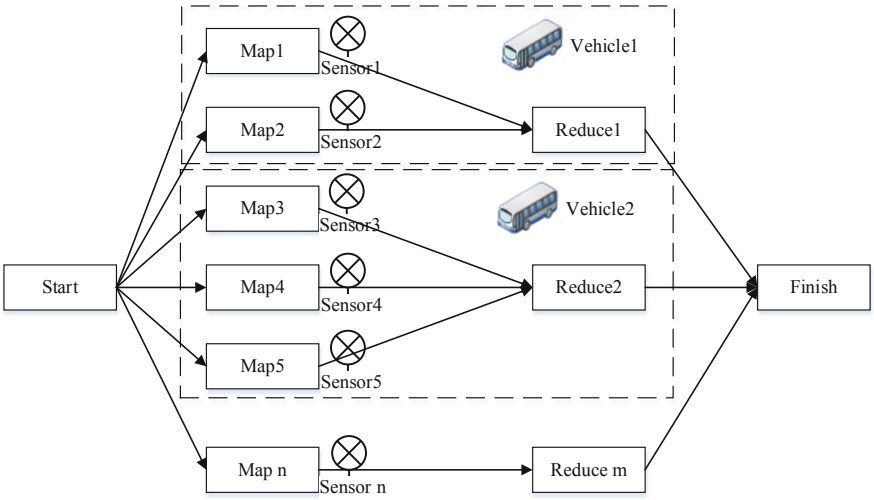

Fig. 1. 2 Framework of data stream classification.

3 The Enhanced ESVM Method

In this paper, we proposed a novel method based on the ESVM model. The ESVM model is a classification model in data mining [13], which is inspired by the Extreme learning machine (ELM) model [14]. The ESVM model employs random value and matrix computing methods to solve model parameters. Since the computation does not require iteration, ESVM is faster in the training process than the traditional SVM model.

Let $S = \{X, Y\}$ be the training sample, let $X = [x_1, x_2, \ldots, x_N]^T \in R^{N \times d}$ be training data, let $Y = [y_1, y_2, \ldots, y_N]^T \in R^{N \times 1}$ be the class label, let $y_i \in [-1, 1]$, let N denote the

size of the training sample, and let d denotes the feature size. The ESVM can be formulated by the following optimal condition:

$$\begin{cases} min \frac{v}{2}\|\varepsilon\|^2 + \frac{1}{2}\left\|\begin{bmatrix} w \\ b \end{bmatrix}\right\|^2 \\ s.t. Y(w\varphi(X) - be) = e - \varepsilon \end{cases} \tag{1}$$

where $\varepsilon = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N]^T$ is the ESVM model error vector, v denotes the error punish parameter, w and b denote weight vector and threshold, respectively, e denotes a unit vector, and $\varphi(x)$ denotes the kernel function of the ELM model, which can be formulated as

$$\varphi(x) = G(Ax) = \left(g\left(\sum\nolimits_{j=1}^{n} A_{1j}x_j + A_{1(n+1)}\right), \ldots, g\left(\sum\nolimits_{j=1}^{n} A_{nj}x_j + A_{n(n+1)}\right)\right) \tag{2}$$

where the matrix $A \in R^{d\times(d+1)}$ denotes the input bias and threshold matrix, which is randomly generated.

For ESVM incremental learning, given a dataset $S_1 = \{X_1, Y_1\}$, the trained ESVM model can be formulated as

$$\begin{bmatrix} w \\ b \end{bmatrix} = \left(\frac{1}{v} + E_{\varphi 1}^T E_{\varphi 1}\right)^{-1} E_{\varphi 1}^T Y_1 \tag{3}$$

Given a new dataset $S_2 = \{X_2, Y_2\}$, the ESVM incremental learning is formulated as

$$\begin{aligned} \begin{bmatrix} w \\ b \end{bmatrix} &= \left(\frac{1}{v} + \begin{bmatrix} E_{\varphi 1}^T & E_{\varphi 2}^T \end{bmatrix}\begin{bmatrix} E_{\varphi 1} \\ E_{\varphi 2} \end{bmatrix}\right)^{-1} \begin{bmatrix} E_{\varphi 1}^T & E_{\varphi 2}^T \end{bmatrix}\begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} \\ &= \left(\frac{1}{v} + E_{\varphi 1}^T E_{\varphi 1} + E_{\varphi 2}^T E_{\varphi 2}\right)^{-1}\left(E_{\varphi 1}^T Y_1 + E_{\varphi 2}^T Y_2\right) \end{aligned} \tag{4}$$

where $E_{\varphi 1}^T E_{\varphi 1}$ and $E_{\varphi 1}^T Y_1$ is same as in Eq. (3) and does not need to be recomputed. Therefore, we only need to compute $E_{\varphi 2}^T E_{\varphi 2}$ and $E_{\varphi 2}^T Y_2$ in the incremental learning process.

3.1 Parallel ESVM Model

The kernel idea of the parallel ESVM (PESVM) model can be formulated as

$$\mathrm{E}_{\varphi}^T \mathrm{Y} = [\mathrm{a}_1, \ldots, \mathrm{a}_N][\mathrm{y}_1, \ldots, \mathrm{y}_N]^T = \sum\nolimits_{i=1}^{N} \mathrm{y}_i a_i \in \mathbb{R}^{(d+1)\times 1} \tag{5.1}$$

$$\mathrm{E}_{\varphi}^T \mathrm{Y} = [\mathrm{a}_1, \ldots, \mathrm{a}_N][\mathrm{y}_1, \ldots, \mathrm{y}_N]^T = \sum\nolimits_{i=1}^{N} \mathrm{y}_i a_i \in \mathbb{R}^{(d+1)\times 1} \tag{5.2}$$

Where $E_{\varphi}^T E_{\varphi}$ and $E_{\varphi}^T Y$ are the same as in Eq. (4).

In most practical applications, the sample size is usually larger than the feature size so that the main computing process can focus on $E_\varphi^T E_\varphi$ and $E_\varphi^T Y$. According to Eqs. (5.1 and 5.2), the computing process of $E_\varphi^T E_\varphi$ and $E_\varphi^T Y$ is equivalent to the sum of a small-scale matrix, which is suitable for applying in the MapReduce framework. Therefore, the PESVM model can be illustrated as shown in Table 1.

Table 1. PESVM model algorithm.

Algorithm 1 PESVM model

Step 1 Input weighted matrix A

Step 2 Samples in the dataset are inputted into the Map process, compute $\varphi(x_i)$ using Eq. (2), and then construct $a_i=[\varphi(x_i),-1]^T$ as

$$\text{local\_aa}^T = \sum a_i a_i^T,\ \text{local\_ya} = \sum y_i a_i \tag{6}$$

Step 3 In the Reduce step, compute

$$\text{global\_aa}^T = global\_\text{aa}^T + \sum local_{a_i a_i^T}, \text{Global\_ya}^T = global\_\text{ya}^T + \sum local\_ya_i^T \tag{7}$$

Step 4 According to Eq. (4), parallel ESVM obtains the values of the parameters w and r .

3.2 Time-Forgetting Strategy PESVM Model

Although the PESVM model employs an incremental learning method and the MapReduce framework to accelerate the data stream classification process, there are still two basic problems. First, the parameter of the kernel function is randomly selected, which would highly affect the final classification result and make the output of classification unstable. Second, although the PESVM model updates itself by an incremental learning method, the training samples in incremental learning are not distinguished, i.e., all of them have the same learning weight.

Therefore, for the above two problems, in this paper, we propose a time-forgetting strategy PESVM model (TFS-PESVM). TFS-PESVM mainly contains two steps: firstly, it employs a kernel density method to evaluate the residual probability distribution of the training data; and secondly, it uses a weight matrix to correct the residual values and improve reliability. Also, the proposed forgetting strategy decreases the effect of old samples and increases the effect of the new samples at the same time.

3.3 TFS-PESVM Modeling

TFS-PESVM model training process can be formulated as

$$\varphi(\mathrm{X})^T w - be = [\varphi(\mathrm{X}) \quad -\mathrm{e}] \begin{bmatrix} w \\ b \end{bmatrix} = \mathrm{E}_\varphi \begin{bmatrix} w \\ b \end{bmatrix} = Y + \varepsilon \tag{8}$$

where the error ε is computed using the Least Square method in the original PESVM model. To correct the error, we use the residual matrix method. Let the residual matrix is P, the objective function is formulated as

$$\min \frac{v}{2}\varepsilon^T P\varepsilon + \frac{1}{2}\left\|\begin{bmatrix} w \\ b \end{bmatrix}\right\|^2 \tag{9}$$

Combining Eqs. (9) with (8), the final TFS-PESVM is

$$\begin{bmatrix} w \\ b \end{bmatrix} = \left(I + vE_\varphi^T P\mathrm{E}_\varphi\right)^{-1} E_\varphi^T PY \tag{10}$$

where $P = diag\{p(\varepsilon_1), p(\varepsilon_2), \cdots, p(\varepsilon_N)\}$ and $p(\varepsilon)$ is probability distribution function of the residual ε. However, in some practical situations, the residual ε cannot be obtained in advance, so we compute estimates of $\varepsilon(\widetilde{\varepsilon})$ using Eq. (8), and then use kernel density estimation to obtain $\mathrm{p}(\varepsilon)$,

$$\mathrm{p}(\varepsilon) = \frac{1}{N}\sum\nolimits_{i=1}^{N} f\left(\frac{\varepsilon - \widetilde{\varepsilon}_i}{h_N}\right) \tag{11}$$

where N is the sample size, $h_N = h/\sqrt{N}$ is the size of the sliding window, and *f(x)* is the window function, which is employed to obtain a smooth probability density estimation and formulated as

$$\mathrm{f}(\mathrm{x}) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{1}{2}x^2\right) \tag{12}$$

According to the Eq. (10), the incremental learning of the TFS-PESVM can be formulated as

$$\begin{aligned} \begin{bmatrix} w \\ b \end{bmatrix} &= \left(I + v\begin{bmatrix} E_{\varphi 1}^T & E_{\varphi 2}^T \end{bmatrix}\begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}\begin{bmatrix} \mathrm{E}_{\varphi 1} \\ \mathrm{E}_{\varphi 2} \end{bmatrix}\right)^{-1}\begin{bmatrix} E_{\varphi 1}^T & E_{\varphi 2}^T \end{bmatrix}\begin{bmatrix} P_1 & 0 \\ 0 & P_2 \end{bmatrix}\begin{bmatrix} \mathrm{Y}_1 \\ \mathrm{Y}_2 \end{bmatrix} \\ &= \left(I + vE_{\varphi 1}^T P_1 \mathrm{E}_{\varphi 1} + vE_{\varphi 2}^T P_2 \mathrm{E}_{\varphi 2}\right)^{-1}\left(E_{\varphi 1}^T P_1 Y_1 + E_{\varphi 2}^T P_2 Y_2\right) \end{aligned} \tag{13}$$

In Eq. (13), the old and new samples have the same weight in the learning process. However, as we know, the new sample has more current information than the old samples. Therefore, we employ a time-forgetting parameter θ to improve the efficiency of the new sample.

Then $E_{\varphi 1}$ in Eq. (13) can be changed to

$$\mathrm{E}_{\varphi 1}^{'} = \theta E_{\varphi 1} \tag{14}$$

where $0 \leq \theta \leq 1$. The modified Eq. (13) is

$$\begin{bmatrix} w \\ b \end{bmatrix} = \left(I + v\theta^2 E_{\varphi 1}^T P_1 \mathrm{E}_{\varphi 1} + v E_{\varphi 2}^T P_2 \mathrm{E}_{\varphi 2} \right)^{-1} \left(\theta E_{\varphi 1}^T P_1 Y_1 + E_{\varphi 2}^T P_2 Y_2 \right) \tag{15}$$

The expressions for $E_\varphi^T P E_\varphi$ and $E_\varphi^T P Y$ in Eq. (15) are

$$E_\varphi^T P \mathrm{E}_\varphi = [a_1, \ldots, a_N] \begin{bmatrix} p_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p_n \end{bmatrix} [a_1, \ldots, a_N]^T = \sum_{i=1}^{N} p_i a_i a_i^T \in \mathbb{R}^{(d+1)\times(d+1)}$$

$$E_\varphi^T P \mathrm{Y} = [a_1, \ldots, a_N] \begin{bmatrix} p_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & p_n \end{bmatrix} [y_1, \ldots, y_N]^T = \sum_{i=1}^{N} y_i p_i a_i \in \mathbb{R}^{(d+1)\times 1} \tag{16}$$

As in the PESVM model, the proposed TFS-PESVM model can also compute local $\sum_{i=1}^{\tilde{n}} p_i a_i a_i^T$ and $\sum_{i=1}^{\tilde{n}} y_i p_i a_i$ in the Map step, and collect the result of $\sum_{i=1}^{N} a_i a_i^T$ and $\sum_{i=1}^{N} y_i a_i$ in the Reduce step (Table 2).

Table 2. TFS-PESVM model algorithm.

Algorithm 2 TFS-PESVM model
Step 1 Input weighted matrix A
Step 2 Use Algorithm 1 to estimate the residual $\tilde{\varepsilon}$
Step 3 Use Eq. (11) to estimate p(ε) and compute weight matrix P
Step 4 Samples in the dataset are inputted into Map process, compute φ(xi) using Eq. (2), and then construct $a_i=[\varphi(x_i),-1]^T$ by: $\text{local\_paa}^T = \sum p_i a_i a_i^T$, $\text{local\_ypa} = \sum y_i p_i a_i$ (17)
Step 5 In Reduce step, compute: $\text{global\_paa}^T = \theta^2 global\_\text{paa}^T + \sum local\_\text{paa}^T$, $\text{lobal\_ypa}^T = \theta global\_\text{ypa} + \sum local\_ypa$ (18)
Step 6 According to Eq. (15), TFS-PESVM obtains the values of parameters w and r.

3.4 Complexity

The model of this paper is based on ESVM, which could reduce the complexity greatly. For ESVM model, it's a Compared with the original ESVM, we remove the unnecessary process for better data stream computing. The complexity of SVM is from o (m^2n) to o(n^3), according to the size of simples (n) and number of features (m). The proposed complexity of no more than that of SVM.

4 Experimental Result

To test the validity of the proposed model, we simulate a specialty vehicle network and generate a data stream with this network. For the application, the scene of this study is some specialty vehicle, such as a tank. Thus, it is hard to obtain a real-world data stream. In the experiment, we employ four large-scale synthetic data sets, which are generated with the tools of concept drift. The whole dataset contains 100,000,000 samples and approximately 7.43 GB of data. The environment of the experiment contains 27 computing nodes, including one main node, one scheduling node, one backup node, and 24 data nodes. Each node is running on a 2.5 GHz Dual-core Intel CPU, 8 GB Memory, Ubuntu 12.04 OS and Hadoop 0.23.0.

4.1 TFS-PESVM Performance Experiment

In this experiment, the main objective is to demonstrate the performance of the proposed model. Therefore, we employ three types of measurement, i.e., speed up, scalability and scale-growth. The speedup denotes the efficiency of acceleration with increasing number of computing nodes. The scalability shows the effect of the data set for the proposed model. The scale-growth shows the complexity of the proposed model. The experimental results are shown in Fig. 2, which contains three sub-figures.

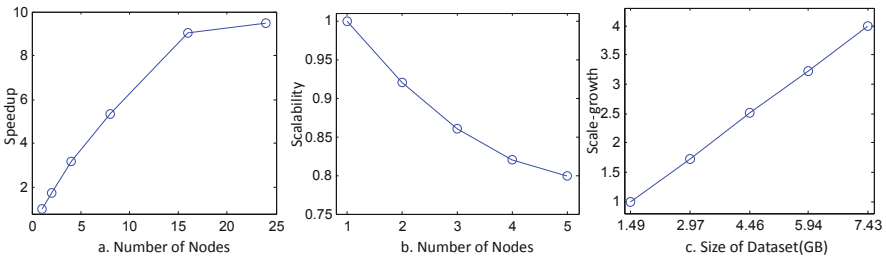

Fig. 2. Speedup, scalability and scale-growth of TFS-PESVM model.

$$\text{Speedup (m)} = \frac{time \ - \ cost \ of \ 1 \ node \ for \ 1 \ data \ set}{time \ - \ cost \ of \ m \ node \ for \ 1 \ data \ set}$$

$$\text{Scalability (m)} = \frac{time \ - \ cost \ of \ 1 \ node \ for \ 1 \ data \ set}{time - cost \ of \ m \ node \ for \ m \ data \ set}$$

$$\text{ScaleGrowth (m)} = \frac{time \ - \ cost \ of \ 1 \ node \ for \ m \ data \ set}{time \ - \ cost \ of \ m \ node \ for \ 1 \ data \ set}$$

From Fig. 2a, the speedup value is increasing with the number of computing nodes in an almost linear fashion. That means in the MapReduce framework, we can employ more computing nodes to accelerate the model's running speed. Therefore, for this

natural feature of the MapReduce framework, the Fig. 2a shows that the proposed model has a good speedup ability. For the Fig. 2b, the scalability decreases with an increased number of computing nodes. The reason is that in the MapReduce framework, the time cost depends on the number of nodes and the data set, so when different computing nodes work for different data sets, the whole time-cost equals to the sum of the costs of each node, which is more than one node for one data set. Finally, for the Fig. 2c, when increasing the data size, the scale-growth also increases linearly, which means the proposed model can handle a "Big Data" problem.

4.2 Comparing Experiment

In this experiment, the proposed model is compared with two distinct models, namely the forgetting factor incremental ELM (FF-IELM) and the forgetting factor incremental ESVM (FF-IESVM). The experimental results are shown in Fig. 3.

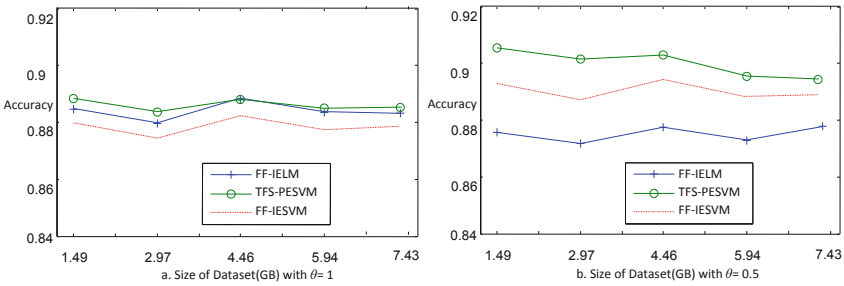

Fig. 3. Accuracy of FF-IELM, FF-IESVM and TFS-PESVM.

In Fig. 3, for $\theta = 1$, the three models have nearly the same classification accuracy, the best one is the TFS-PESVM model and the worst one is the FF-IESVM model. However, for $\theta = 0.5$, the accuracy of the proposed model TFS-PESVM is significantly better, and the worst is FF-IELM. After the analysis, two conclusions can be drawn:

After comparing with two different classification models, it can be seen that the proposed TFS-PESVM model can improve the accuracy of data stream classification significantly;

The forgetting strategy can improve the accuracy of data stream classification, which validates the new samples as more useful than the old ones.

For $\theta = 0.5$, we also repeat this experiment five times to test the stability of the proposed model, which is shown in Table 1. In Table 3, there are two new measurements, namely sensitive and specificity, which can be formulated as

$$\text{Sensitive} = \frac{pp}{pp + pn}$$

$$\text{Specificity} = \frac{nn}{pp + np}$$

Table 3. Comparing of FF-IELM, FF-IESVM and TFS-PESVM model.

Model	Measurement	Data set 1	Data set 2	Data set 3	Data set 4
FF-IELM	Accuracy	87.58 ± 1.23%	87.18 ± 1.22%	87.74 ± 1.20%	87.3 ± 1.15%
	Sensitive	82.38 ± 1.63%	82.2 ± 1.39%	88.76 ± 1.39%	90.62 ± 1.45%
	Specificity	91.98 ± 0.96%	92.14 ± 1.13%	86.9 ± 1.23%	84.84 ± 1.11%
FF-IESVM	Accuracy	89.3 ± 1.33%	88.7 ± 1.23%	89.44 ± 1.3%	88.92 ± 1.23%
	Sensitive	84.86 ± 2%	83.3 ± 1.8%	88.76 ± 1.7%	91.58 ± 1.72%
	Specificity	**93.14 ± 1.22%**	**94.06 ± 1.13%**	**90.02 ± 1.25%**	86.94 ± 1.19%
TFS-PESVM	Accuracy	**90.56 ± 1.64%**	**90.14 ± 1.77%**	**90.3 ± 1.19%**	**89.54 ± 0.89%**
	Sensitive	**87.6 ± 4.01%**	**86.32 ± 3.38%**	**91.94 ± 2.28%**	**91.96 ± 4.77%**
	Specificity	93.12 ± 1.27%	94 ± 0.98%	88.92 ± 0.9%	**87.02 ± 1.88%**

where *pp* and *nn* denote the number of correctly classified samples with positive or negative label. Additionally, *pn* and *np* denote the number of wrongly classified samples with positive or negative label.

In Table 1, for data set 1 to data set 3, the best accuracy and sensitive measurements are found with the proposed model TFS-PESVM, which means the proposed model is more stable and efficient than the others. However, for specificity, the best is the FF-IESVM model. The reason is that the FF-IESVM model is more suitable for negative label samples. For data set 4, the proposed TFS-PESVM model obtains the best classification results. Overall, the experimental results show that the forgetting strategy and the MapReduce framework used in original ESVM model can improve the accuracy of classification and model stability, which is more suitable for data stream classification in real time.

5 Conclusion

Data stream classification is a new research topic in the data mining community. The application of data stream classification on specialty vehicle network is also a novel idea. However, due to the basic features of a specialty vehicle network, the data stream classifier needs to not only improve accuracy but also improve the running speed of the classification process. Therefore, in this paper, a new data stream classification model, namely TFS-PESVM, is proposed, which employs a residual weight matrix to correct the residual matrix in ESVM model and designs a new forgetting factor to control the effect of samples and decrease the effect of noise in the data stream. The experimental results on a vehicle network simulation show that the proposed model can improve the accuracy of classification. Compared with two other models, namely FF-IELM and FF-IESVM, the proposed method is also more stable in accuracy than the others.

References

1. O'Connor, P., Neil, D., Liu, S.C., Delbruck, T., Pfeiffer, M.: Real-time classification and sensor fusion with a spiking deep belief network. Neuromorphic Eng. Syst. Appl. 61(2015)
2. Gao, Z., Cecati, C., Ding, S.X.: A survey of fault diagnosis and fault-tolerant techniques—part II: fault diagnosis with knowledge-based and hybrid/active approaches. IEEE Trans. Ind. Electron. **62**(6), 3768–3774 (2015)
3. Haque, A., Khan, L., Baron, M., Thuraisingham, B., Aggarwal, C.: Efficient handling of concept drift and concept evolution over stream data. In: 2016 IEEE 32nd International Conference on Data Engineering (ICDE), pp. 481–492. IEEE, May 2016
4. Triguero, I., Peralta, D., Bacardit, J., García, S., Herrera, F.: MRPR: a MapReduce solution for prototype reduction in big data classification. Neurocomputing **150**, 331–345 (2015)
5. Zhai, J., Zhang, S., Wang, C.: The classification of imbalanced large data sets based on MapReduce and ensemble of ELM classifiers. Int. J. Mach. Learn. Cybern. 1–9 (2016)
6. Zhang, C., Li, F., Jestes, J.: Efficient parallel kNN joins for large data in MapReduce. In: Proceedings of the 15th International Conference on Extending Database Technology, pp. 38–49. ACM, March 2012
7. Lee, T., Im, D.H., Kim, H., Kim, H.J.: Application of filters to multiway joins in MapReduce. Math. Probl. Eng. (2014)
8. Alham, N.K., Li, M., Hammoud, S., Liu, Y., Ponraj, M.: A distributed SVM for image annotation. In: 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), vol. 6, pp. 2983–2987. IEEE, August 2010
9. Feng, W., Zhang, Q., Hu, G., Huang, J.X.: Mining network data for intrusion detection through combining SVMs with ant colony networks. Future Gener. Comput. Syst. **37**, 127–140 (2014)
10. Zhao, J., Liang, Z., Yang, Y.: Parallelized incremental support vector machines based on MapReduce and bagging technique. In: 2012 IEEE International Conference on Information Science and Technology, pp. 297–301. IEEE, March 2012
11. Dean, J., Ghemawat, S.: MapReduce: simplified data processing on large clusters. Commun. ACM **51**(1), 107–113 (2008)
12. Doulkeridis, C., Nørvåg, K.: A survey of large-scale analytical query processing in MapReduce. VLDB J. **23**(3), 355–380 (2014)
13. Zhu, W., Miao, J., Qing, L.: Extreme support vector regression. In: Sun, F., Toh, K.-A., Romay, M.G., Mao, K. (eds.) Extreme Learning Machines 2013: Algorithms and Applications. ALO, vol. 16, pp. 25–34. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-04741-6_3
14. Teo, T.T., Logenthiran, T., Woo, W.L.: Forecasting of photovoltaic power using extreme learning machine. In: 2015 IEEE Innovative Smart Grid Technologies-Asia (ISGT ASIA), pp. 1–6. IEEE, November 2015