Improving Simplification of Support Vector Machine for Classification

Pham Quoc Thang, Hoang Thi Lam, and Nguyen Thanh Thuy

***Abstract*—The efficient classification ability of support vector machines (SVMs) has been shown in many practical applications, but currently they are considerably slower in testing phase than other approaches with similar classification performance due to a large number of support vectors included in the solution. Among different approaches, simplification of support vector machine (SimpSVM) speeds-up the testing phase by replacing original SVM with a simplified one that consists of a smaller number of support vectors. However, the ultimate goal of the simplification is to keep the simplified solution as similar to the original solution as possible. To improve this similarity, in this paper, we propose two improved SimpSVMs that are based on stochastic gradient descent. Experiments on some datasets show improved results by our algorithms.**

***Index Terms*—Support vector machines, simplification of support vector machine, stochastic gradient descent, solution optimization.**

I. Introduction

In recent years, the classification problem has been widely studied. Many factors can affect the results of classification such as incomplete data, selecting parameter values for a particular model. To solve the classification problem more efficiently, several methods have been proposed: the decision trees [1], the back-propagation neural networks [2] and the support vector machines (SVM) [3].

SVM (Vapnik, 1995) is an efficient machine learning method to solve classification and regression problems. SVMs have been shown to be successful in many pattern recognition problems such as speech recognition, digits recognition, handwriting recognition… By combining with the kernel function method, SVMs provide effective models for classification and nonlinear regression problems in practice. However, SVMs is considerably slower in testing phase than other learning methods with similar generalization performance such as decision trees, neural networks [4]-[8].

The solution of an SVM is parameterized by a set of input vectors called support vectors and their corresponding weights. To classify a new test example, SVMs compare it with these support vectors via kernel calculations; this number of comparison scales linearly with the number of support vectors and becomes very expensive if the number of support vectors is large. While SVMs are robust techniques for classification, the large size and slow query time of a trained SVM is one of the obstacles to their practical application. Therefore, reducing this comparison will increase the speed of the testing phase.

There have been some proposed algorithms to reduce this computational complexity, either by removing less important support vectors or by constructing a new smaller set of vectors, often with minimal impact on accuracy. [4] has developed the first constructive reduced set methods, by approximately the original SVM with a new one includes a much smaller number of newly constructed vectors, called the reduced vectors set. This approach is also described in [8] and further developed in [9]. [6], [9] start from approximately the solution includes all original support vectors by a new vector, and then incrementally construct the reduced set by finding vectors that minimize the differences between the original vector expansion and the reduced set expansion in feature space. The authors in [10] extended this method by greedily choosing the binary SVM with the lowest accuracy to receive the next reduced-set vector, retraining each binary SVM using the original SVM objective function to obtain optimal weights, and then share reduced-set vectors between multiple component binary SVMs in a multiclass SVM for additional gains.

Ref. [11] proposed the reduction process is iteratively selecting two nearest support vectors belonging to the same class and replacing them with a newly constructed reduced vector. [12] extend this bottom-up method for simplifying binary SVM to the multi-class case by calculating for optimally combining two multi-weighted support vectors, selecting heuristic for choosing a good pair of support vectors for replacing them with a newly created vector.

However, the ultimate goal of the simplification is to keep the simplified solution as similar to the original solution as possible. To improve this similarity, [12] proposed to adjust all reduced vectors globally concerning norms of solution's hyperplanes by minimizing the difference between them and then the authors use a gradient descent for minimizing this difference, but the speed is quite slow. In this paper, we will introduce our improved versions of the SimpSVM algorithm in [12]. They have ideas of stochastic gradient descent method to solve the above solution optimization problem. Experimental results on different datasets show that the improved SimpSVMs can reduce the time for simplifying SVM while keeping the predictive performance of simplified SVMs has not been changed much.

The remainder of this paper is organized as follows. Section II, III present SVM, SimpSVM. Section IV describes the proposed methods. Section V presents experimental results and conclusions are presented in the last section.

Manuscript received May 6, 2018; revised July 10, 2018.

Pham Quoc Thang, Hoang Thi Lam are with Tay Bac University, Vienam (e-mail: thangpq@utb.edu.vn).

Nguyen Thanh Thuy is with VNU University of Engineering and Technology, Vietnam.

doi: 10.18178/ijmlc.2018.8.4.714

II. Support Vector Machines

Support vector machines (SVMs) work in feature space F via a kernel function $K(x,y)=\Phi(x).\Phi(y)$ where $\Phi: R^d \rightarrow F$ is a map from the d-dimensional input space to a possibly high-dimensional feature space [3]. $K(x,y)=\Phi(x).\Phi(y)$ is a kernel function calculating the dot product of two vectors $\Phi(x)$ and $\Phi(y)$ in the feature space.

For a binary-class classification problem, the decision rule takes the form:

$$f(x)=sign\left(\sum_{i=1}^{N_S}\alpha_i K(x,x_i)+b\right) \quad (1)$$

where N_S is the number of support vectors, α_i is the weight of support vector x_i, $i = 1,\ldots,N_S$, x is the input vector needed to classify, and b is the bias. The task of the SVMs training process is to determine all the parameters (x_i, α_i, b, N_S). The results we have a set of x_i, $i = 1,\ldots,N_S$ is a subset of the training set, they are called *support vectors*.

As support vector learning is principally designed for binary-class classification, more than one SVM are required to form a classifier for a multi-class application. The most popular way is to use T binary one-versus-rest SVMs or $T(T-1)/2$ one-versus-one SVMs, where T is the number of classes. In other words, the final decision is based on not one, but a set of T functions

$$f_t(x)=\sum_{i=1}^{Ns}\alpha_{ti}K(x_i,x)+b_t \quad , \quad t=1,...,T \quad (2)$$

with $\alpha_{ti}\neq 0$ means vector x_i is one support vector of the t^{th} SVM and its corresponding weight is α_{ti}, otherwise $\alpha_{ti}=0$ means x_i is not related to the t^{th} SVM [12].

III. Simplification of Support Vector Machine

For both binary-class and multi-class SVM, the most expensive procedure in testing a new object vector x is to compare it with the whole set of support vectors via kernel function K. This computation scales linearly with the number of support vectors N_S. To reduce this computation cost, or to speed-up the testing phase, reduced set method tries to replace N_S, the number of original support vectors, by N_Z, a smaller number of new vectors, called *reduced vector* set. The decision functions then become (T = 1 for the binary-class case)

$$f'_t(x)=\sum_{i=1}^{N_Z}\beta_{ti}K(z_i,x)+b_t \quad , \quad t=1,...,T \quad (3)$$

There have been some studies done on constructing reduced vectors in which [11] proposed to iteratively select two support vectors and replace them with one combined vector. Call (x_i, x_j) be the selected pair of support vectors, the combined vector z for replacing x_i and x_j is found by solving

$$\min \leftarrow \left\|\beta\Phi(z)-(\alpha_i\Phi(x_i)+\alpha_j\Phi(x_j))\right\|^2 \quad (4)$$

For multi-class, supposing that we want to replace two multi-weighted support vectors (x_i, α_{ti}) and (x_j, α_{tj}) by a single new vector (z, β_t), t = 1,..,T, the 2-norm optimal solution for all single SVMs will be the one that minimizes

$$\min \leftarrow \sum_{t=1}^{T}\left\|\beta_t\Phi(z)-(\alpha_{ti}\Phi(x_i)+\alpha_{tj}\Phi(x_j))\right\|^2 \quad (5)$$

The simplification procedure iteratively selects two support vectors (including newly created vectors) x_i and x_j and replaces them with a new vector z using the method described in [12].

The combination procedure described above aims at constructing one new vector to replace one selected pair of support vectors. The combination criterion, or the objective function in (5), is locally optimized for the two support vectors in a pair. However, the ultimate goal of the simplification is to keep the simplified solution as similar to the original solution as possible. To improve this similarity, [12] proposed to adjust all reduced vectors globally concerning norms of solution's hyperplanes by minimizing the difference between them:

$$\min \leftarrow \rho=\sum_{t=1}^{T}\left\|\sum_{i=1}^{N_S}\alpha_{ti}\Phi(x_i)-\sum_{i=1}^{N_Z}\beta_{ti}\Phi(z_i)\right\|^2 \quad (6)$$

IV. Improved SimpSVM Algorithms

This section introduces two improved approaches, which are developed for improving the reducing time of original SimpSVM.

A. Original SimpSVM Using Gradient Descent

In [12], the authors applied the gradient descent for minimizing ρ concerning all reduced vectors z_i, $i=1,\ldots,N_Z$. The search directions for Gaussian RBF and polynomial kernels are:

$$\frac{\partial\rho_{RBF}}{\partial z_i}=\sum_{t=1}^{T}\left(\sum_{j=1}^{N_Z}-2\gamma\beta_{ti}\beta_{tj}K(z_i,z_j)(z_i-z_j)-\sum_{j=1}^{N_S}-2\gamma\beta_{ti}\alpha_{tj}K(z_i,x_j)(z_i-x_j)\right) \quad (7)$$

$$\frac{\partial\rho_{Poly}}{\partial z_i}=\sum_{t=1}^{T}\left(\sum_{j=1}^{N_Z}p\beta_{ti}\beta_{tj}(z_i.z_j)^{p-1}z_j-\sum_{j=1}^{N_S}p\beta_{ti}\alpha_{tj}(z_i.x_j)^{p-1}x_j\right) \quad (8)$$

At each iteration, updating all z_i as follows:

$$z_i^{(t+1)}=z_i^{(t)}-\eta\frac{\partial\rho}{\partial z_i^{(t)}} \quad , \quad i=1,...,N_Z \quad (9)$$

B. Improved SimpSVM Using Stochastic Gradient Descent

We propose the first improved SimpSVM based on stochastic gradient descent, called SimpSVM-SGD. At each iteration, instead of adjusting all reduced vectors globally, we adjust a single randomly picked z_i with a single randomly chosen direction. The update rule is then

$$z_{ik}^{(t+1)}=z_{ik}^{(t)}-\eta\frac{\partial\rho}{\partial z_{ik}^{(t)}} \quad , \quad i\in[1,...,N_Z];\ k\in\dim(z_i) \quad (10)$$

By adjusting only a single reduced vector, at each iteration, we only need to recalculate the kernel function K in (6-8) related to chosen z_i without having to recalculate the entire kernel function K. It can be shown that SimpSVM-SGD

minimizes the generalization error quicker than original SimpSVM using gradient descent.

C. Improved SimpSVM Using Stochastic Vector Descent

We propose the second improved SimpSVM based on stochastic vector descent, called SimpSVM-SVD. As seen in (6-8), if using stochastic gradient descent in one direction, when recalculating the kernel function K, we still have to calculate by all directions. It makes more consuming unnecessary time. Therefore, we propose to use the stochastic gradient descent with all directions to reduce this calculation. At each iteration, instead of adjusting all reduced vectors globally, we adjust a single randomly picked z_i but with all directions. The update rule is then

$$z_i^{(t+1)} = z_i^{(t)} - \eta \frac{\partial \rho}{\partial z_i^{(t)}} \quad , \quad i \in [1,...,N_Z] \tag{11}$$

This approach has a computational complexity that scales linearly with the number of reduced vectors of the problem. It is significantly quicker than original SimpSVM using gradient descent when the number of reduced vectors is large.

V. Experiment

In this section, we evaluate the effects of the proposed SimpSVM-SGD and SimpSVM-SVD algorithms. We have implemented them by using programming languages C/C++. We compare the performance of the SimpSVM-SGD and SimpSVM-SVD with original SimpSVM on six benchmarks and one sign language datasets. All the programs were run on a PC with CPU Intel Core i3 3.3 GHz, 2GB RAM.

A. Benchmark Datasets

The performance of the SimpSVM-SGD and SimpSVM-SVD has been studied for six benchmark datasets, namely the *DNA*, *Letter Recognition*, *Shuttle*, *Vowel*, *Pendigits*, and *Mnist*. All datasets are publicly available from the UCI Machine Learning Repository [13], and their details are given in Table I. The selection of parameters C, γ for SVM models is very important. We used a grid search to select the parameters so that the trained (original) SVM classifiers have good predictive accuracy on independent test datasets. For the reproducibility of experiments, we also report the parameter values set in Table I.

In the first experiment, we run the proposed algorithms with different values of N_Z, indicating different speed-up rates of simplified SVMs. Then we compare the accuracy of improved SimpSVMs and original one on the prepared test data. Experimental results reported in Table II show that when the number of reduced vectors increased, the proposed SimpSVMs have performance as close to the original one, with 25% of reduced vectors or more, they have the almost identical performance with the original one.

TABLE I: Characteristics of Datasets and Parameter Setting

Name	# Attributes	# Class	# Training	# Testing	Parameters
dna	180	3	1400	1186	C = 10, γ = 0.01
letter	16	26	10500	5000	C = 10, γ = 2
shuttle	9	7	30450	14500	C = 10, γ = 0.1
vowel	10	11	528	462	C = 2 , γ = 2
pendigits	16	10	7494	3498	C = 4 , γ = 0.25
mnist	780	10	60000	10000	C = 10, γ = 0.0128

TABLE II: Predictive Accuracy of SimpSVMs with Different Speed-up Rates on Six Datasets

Data		Percentage of support vectors				
		100%	50%	25%	10%	5%
Dna	# SV	654	327	164	65	33
	Original SimpSVM	94.44	94.52	94.18	94.44	94.86
	SimpSVM-SVD	94.44	94.44	94.18	93.59	91.91
	SimpSVM-SGD	94.44	94.01	93.42	93.09	90.73
Letter	# SV	6014	3007	1504	601	301
	Original SimpSVM	97.04	97.00	96.74	91.54	80.86
	SimpSVM-SVD	97.04	97.02	96.68	87.98	71.40
	SimpSVM-SGD	97.04	96.88	96.26	87.02	68.54
Shuttle	# SV	3208	1604	802	321	160
	Original SimpSVM	98.98	98.98	98.98	98.98	98.98
	SimpSVM-SVD	98.98	98.98	98.98	98.98	98.98
	SimpSVM-SGD	98.98	98.98	98.98	98.98	98.98
Vowel	# SV	393	197	98	39	20
	Original SimpSVM	60.82	60.82	59.52	38.31	37.45
	SimpSVM-SVD	60.82	61.26	58.66	38.74	35.71
	SimpSVM-SGD	60.82	60.82	58.44	39.83	39.18
Pendigits	# SV	948	474	237	95	47
	Original SimpSVM	98.48	98.48	98.46	98.26	95.14
	SimpSVM-SVD	98.48	98.48	98.37	97.94	95.74
	SimpSVM-SGD	98.48	98.40	98.34	96.71	91.51
Mnist	# SV	11554	5777	2889	1155	578
	Original SimpSVM	98.31	98.34	98.26	98.03	97.84
	SimpSVM-SVD	98.31	98.27	98.21	97.98	97.60
	SimpSVM-SGD	98.31	98.33	97.94	96.96	95.36

In the second experiment, we compare reducing time of improved SimpSVMs and original one on the prepared test datasets with *Nz* equal to half of *Ns*. This comparison is shown in Fig. 1. As can be seen in Fig. 1, on six datasets,

SimpSVM-SGD has a smaller reducing time than original SimpSVM while keeping the predictive performance of simplified SVMs has not been changed much and SimpSVM-SVD can reduce the time for simplifying SVM while the performance is almost unchanged. Special, on the "*letter*", *"shuttle"*, "*vowel*" and "*mnist*" datasets, SimpSVM-SVD has reducing time less than one-fifth of original SimpSVM, or otherwise it can run faster original one up to 5 times without any loss in predictive accuracy. Between improved SimpSVMs, SimpSVM-SVD runs faster than SimpSVM-SGD in most datasets.

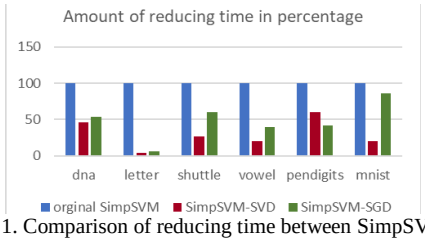

Fig. 1. Comparison of reducing time between SimpSVMs.

In the third experiment, we compare our improved SimpSVMs, original SimpSVM and the SVM simplification methods described in [4], [10] on the "*usps*" handwritten recognition dataset. In our experiment, SVM in LibSVM produces 45 one-vs-one SVMs with a total number of support vectors are 1459 (with Gaussian kernel), and the accuracy is 95.32%. The comparison in Fig. 2 shows that the SimpSVM-SVD produces competitive performance in term of speeding-up rate and preserving the predictive accuracy of simplified SVMs.

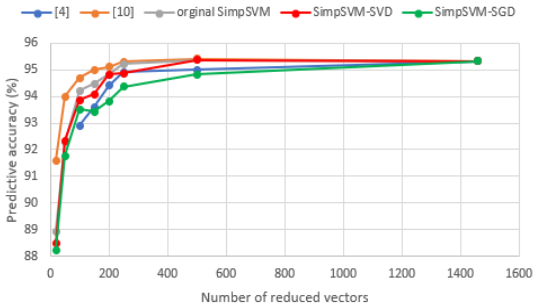

Fig. 2. Comparison of number of RVs and predictive accuracy between SimpSVMs.

B. Sign Language Recognition

There are many sign languages in the world, in which Auslan is the sign language used by the Non-vocal and Deaf community in Australia. In Auslan sign language, to express a meaning, each sign is expressed as a series of gestural patterns (Fig. 3). There are multiple signs in Auslan language so this is a multi-class classification problem.

Fig. 3. Some sample images of signs in Auslan sign language.

1) Data

We took the dataset from the UCI Machine Learning Repository [13], the dataset was captured from a native Auslan singer by using two Fifth Dimension Technologies (5DT) gloves with two high-quality position trackers over a period of nine weeks. In this data set, each data sample contains a series of 22-dimensional value vectors: x, y, z, yaw, pitch, roll, Little finger bend, Ring finger bend, Middle finger bend, Forefinger bend, Thumb bend... for both hands and their details are described in [14]. The dataset consists of a total of 2565 samples belonging to 95 distinctive signs with an average of 27 samples per sign.

2) Feature extraction

There have been many suggested methods to extract various features from the 22 channels of information. In this experiment, we used the method as described in [14] to extract the global features and meta-features.

First, we computed the global features: extracting various stream features as a whole including: minima/maxima/mean (of each channel). From 22 channels of information, we computed the above three information to achieve 66 global features [15].

Next, we computed the meta-features: extracting the original events, clustering them to generate the basis of synthetic event attributes. With each raw stream, the value of the synthetic event attribute is a confidence metric that original event as belonging to that cluster. For each channel, we used the following five events: localmin, localmax, flat (no perceptible change in gradient), decreasing/increasing (an extended period with considerable negative/positive gradient). For two decreasing and increasing events, we computed four parameters: average gradient, the average value of the channel, start time, duration. For the flat event, we computed three parameters: average value, the start time and duration. For two localmin/localmax events, we computed two parameters: time of the minimum/maximum and the value. The number of extracted meta-features is dependent on the results of clustering events [15].

Finally, the data from the global features and meta-features are recombined in a form suitable for classification.

3) Sign classification

For ease of comparison with the results of other studies that were previously published, we used 5-fold cross-validation procedure for experimentation.

In Fig. 4, we compared the predictive performance and reducing time of SimpSVM-SVD and original SimpSVM on the three types of features: global features, meta-features and both. The results show that both methods increase the classification accuracy when the number of features increases. If using a combination of both feature types, SimpSVM-SVD obtains the high classification accuracy (98,13%) as close to the original SimpSVM (98,17%), but it needs only 564 seconds for reducing time while original one needs to 2094 seconds. On all types of features, SimpSVM-SVD can run faster than original SimpSVM 4 times while the prediction accuracy has not been changed much.

With these above results, we believe that the improved SimpSVMs can reduce the time for simplifying SVM since they allow to improve overall training time of SimpSVM.

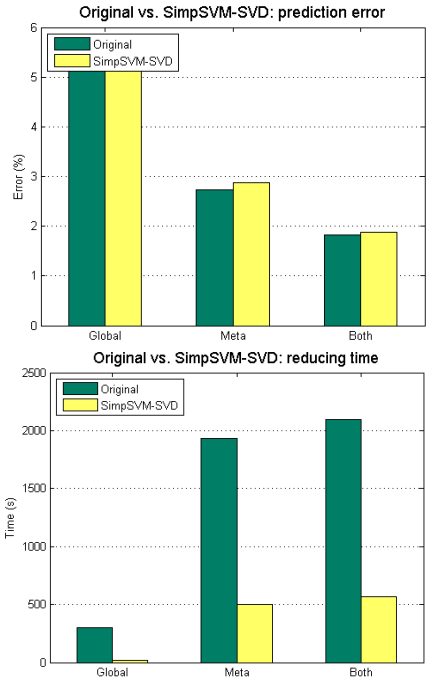

Fig. 4. The error rates and reducing times of models learned by SimpSVMs on Auslan datasets.

VI. Conclusion

In this paper, we have presented two improved SimpSVM algorithms for classification problems. The capability of these algorithms was investigated through the performance of several experiments on some datasets. Experimental results show the effectiveness of our approaches that they can reduce the time for simplifying SVM while keeping the predictive performance of simplified SVMs has not been changed much.

The future work will be to validate our approaches to other datasets and real-life problems.

References

[1] J. R. Quinlan, "Simplifying decision trees," *International Journal of Man-Machine Studies*, vol. 27, no. 3, p. 221.

[2] M. N. Nazri *et al.*, "An improved back propagation neural network algorithm on classification problems," *Database Theory and Application*, Springer Berlin Heidelberg, pp. 177-188, 2010.

[3] C. Cortes and V. Vapnik, "Support vector networks," *Machine Learning*, vol. 20, no. 3, pp. 273-297, 1995.

[4] C. J. C. Burges, "Simplified support vector decision rules," in *Proc. 13th International Conference on Machine Learning*, 1996, pp. 71–77.

[5] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discover*, vol. 2, pp. 121-167, 1998.

[6] C. J. C. Burges and B. Schoelkopf, "Improving the accuracy and speed of support vector learning machines," *Advances in Neural Information Processing Systems*, vol. 9, pp. 375-381, Cambridge, MA: MIT Press, 1997.

[7] Y. LeCun, L. Botou, L. Jackel, H. Drucker *et al.*, "Learning algorithms for classification: A comparison on handwritten digit recognition," *Neural Networks*, pp. 261-276, 1995.

[8] C. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: benchmarking of state-of-the-art techniques," *Pattern Recognition*, vol. 36, pp. 2271-2285, 2003.

[9] B. Sch ölkopf, S. Mika, C. J. C. Burges *et al*., "Input space vs. feature space in kernel-based methods," *IEEE Trans. Neural Networks*, vol. 10, pp. 1000-1017, 1999.

[10] B. Tang and D. Mazzoni, "Multiclass reduced-set support vector machines," in *Proc. Int'l Conf. Machine Learning*, ACM, New York, NY, USA, 2006, pp. 921-928.

[11] N. DungDuc and H. TuBao, "An efficient method for simplifying support vector machines," in *Proc. the 22nd International Conference on Machine Learning*, Bonn, Germany, August 07-11, 2005, vol. 119, pp. 617-624. ACM, New York, 2005.

[12] N. DucDung, K. Matsumoto, K. Hashimoto, Y. Takishima, D. Takatori, and M. Terabe, "Multi-class Support Vector Machine Simplification," in *Proc. the 10th Pacific Rim International Conference on Artificial Intelligence: Trends in Artificial Intelligence*, PRICAI 2008, Hanoi, Vietnam, December 15-19, 2008, pp. 799–808, 2008.

[13] Machine Learning Repository. [Online]. Available: https://archive.ics.uci.edu/ml/datasets.html

[14] M. W. Kadous, "Temporal classification: Extending the classification paradigm to multivariate time series," PhD Thesis, The University of New South Wales, 2002.

[15] Q. T. Pham, D. D. Nguyen, and T. T. Nguyen, "A comparison of simpsvm and rvm for sign language recognition," in *Proc. the 2017 International Conference on Machine Learning and Soft Computing*, Ho Chi Minh City, Vietnam, January 13-16, 2017, ACM, New York, USA, pp. 98–104, 2017.

Pham Quoc Thang is a Ph.D Student at VNU University of Engineering and Technology, Vietnam. He received the B.S. and M.S. degree in computer science from Hanoi National University of Education, Vietnam in 2001 and 2010. He is currently a lecturer at Tay Bac University, Vietnam. His current research interests include data mining, machine learning, data analysis, pattern recognition and artificial intelligence.

Hoang Thi Lam received the B.S. and M.S. degree in computer science from Hanoi National University of Education, Vietnam in 2001 and 2010. She has been a lecturer at Tay Bac University, Vietnam since 2001. Her research interests include information systems, databases and data mining.

Nguyen Thanh Thuy received B.S. degree in Mathematics, and Ph.D. degree in Computer Science from Hanoi University of Technology, Vietnam, in 1982 and 1987. He has been the professor of Vietnam since 2010. His primary research interests include uncertainty, fuzziness and knowledge base systems, decision support systems. He is also interested in soft-computing and hybrid intelligent systems, data mining and knowledge discovery from database, grid and parallel computing.