A homogeneous-heterogeneous ensemble of classifiers.

LUONG, A.V., VU, T.H., NGUYEN, P.M., VAN PHAM, N., MCCALL, J., LIEW, A.W.-C. and NGUYEN, T.T.

2020

A homogeneous-heterogeneous ensemble of classifiers

Anh Vu Luong[1], Trung Hieu Vu[2], Phuong Minh Nguyen[3], Nang Van Pham[4], John McCall[5], Alan Wee-Chung Liew[1], and Tien Thanh Nguyen[5] ✉

[1] School of Information and Communication Technology, Griffith University, Australia
[2] School of Electronics and Telecommunications, Hanoi University of Science and Technology, Hanoi, Vietnam
[3] Department of Environmental Technology, Faculty of Environmental Sciences, VNU University of Science, Vietnam National University, Hanoi, Vietnam
[4] School of Electrical Engineering, Hanoi University of Science and Technology, Hanoi, Vietnam
[5] School of Computing, Robert Gordon University, UK

Abstract. In this study, we introduce an ensemble system by combining homogeneous ensemble and heterogeneous ensemble into a single framework. Based on the observation that the projected data is significantly different from the original data as well as each other after using random projections, we construct the homogeneous module by applying random projections on the training data to obtain the new training sets. In the heterogeneous module, several learning algorithms will train on the new training sets to generate the base classifiers. We propose four combining algorithms based on Sum Rule and Majority Vote Rule for the proposed ensemble. Experiments on some popular datasets confirm that the proposed ensemble method is better than several well-known benchmark algorithms. proposed framework has great flexibility when applied to real-world applications. The proposed framework has great flexibility when applied to real-world applications by using any techniques that make rich training data for the homogeneous module, as well as using any set of learning algorithms for the heterogeneous module.

Keywords: Ensemble method · Multiple classifiers · Combining classifiers · Random Projection · Ensemble Learning · Combining Methods.

1 Introduction

Classification is one of the most studied machine learning problems. Given a set of labeled observations called training set, classification algorithms exploit the knowledge from the features-label relationship so as to assign a class label to an unlabeled sample. Although many learning algorithms have been proposed, no algorithm is known to perform the best for all problems. A popular solution is to combine multiple algorithms in an ensemble system in order to achieve better performance than using any single algorithm. In ensemble learning, training

different learning algorithms on the original training set to generate the base classifiers is known as heterogeneous ensemble, while training only one learning algorithm on many different training sets obtained from the original training data to generate the base classifiers is known as homogeneous ensemble [1].

In this study, we propose an ensemble system by combining homogeneous and heterogeneous ensembles into a single framework. Our work is based on the observation that random projection, a data transformation method, can create different projected data from the original data [2], thus making the new data available for many different learning algorithms to train base classifiers. In the proposed framework, the set of random projections is applied to the original training data to generate the new training sets (homogeneous module). Different learning algorithms then train on the new projected data to obtain base classifiers (heterogeneous module). The outputs of all the base classifiers are combined to get the final collaborated prediction for the sample. For the combining algorithm, we introduce four methods based on the two popular combining rules: Sum Rule and Majority Vote Rule [3]. In the first two combining algorithms, Sum Rule or Majority Vote Rule is directly applied to the outputs of all base classifiers. Meanwhile, in the remaining two combining algorithms, the two rules are combined by conducting the Sum Rule on the predictions associated with each random projection or associated with each learning algorithm first and then applying Majority Vote Rule on the outputs of the Sum Rule.

The structure of this paper is as follows. Section 2 presents background and related work in the ensemble system. Section 3 introduces a new ensemble framework consisted of a combination of homogeneous and heterogeneous ensemble and four combining methods to combine the outputs of the proposed ensemble. Experimental studies are presented in Section 4 in which we describe the settings for the experiments and the comparisons and discussions based on the experimental results. The conclusion is presented in Section 5.

2   Background and Related work

In this section, we briefly introduce research approaches related to the ensemble system. First, there are approaches focusing on designing new architectures for the ensemble system. For example in [4], Zhang et al. used both random subspace and bootstrap sampling technique on the original training data to obtain the new training sets. The k Nearest Neighbor (kNN) algorithm is trained on these new training sets to obtain the EoC. In [5], an ensemble learning-based deep model was proposed in which learning model includes several layers of ensemble of classifiers. One layer receives input training data created by previous layer and then generates input training data for its next layer.

Besides, several combining algorithms have been introduced for classifiers' output aggregation as the better replacement for traditional combiners e.g. Sum Rule and Majority Vote. Nguyen et al. [6] used information granules to model predictions of the base classifiers in the form of vectors of intervals called granule prototypes. In this method, the combining algorithms were constructed by con-

sidering the distance between the predictions for a test sample and the granule prototypes. Kuncheva et al. [7] associated each class by a representation called decision template which is the average of the meta-data of training instances that belong to a class. The class that minimizes the distance between the corresponding decision template and the meta-data of the test sample is the final prediction. Nguyen et al. [8] proposed a Bayesian-based combining method in which the posterior probability that a sample belongs to a class label is computed by using the likelihood and the prior distribution. The likelihood distribution is approximated by the multivariate Gaussian.

Several modifications meanwhile focused on improving the performance of existing ensemble systems. Some approaches search for the weights of the base classifiers in the aggregation [2]. Several improvements for Boosting-based ensemble approach are RotBoost by combining Rotation Forest and AdaBoost in a single framework [9] and TotalBoost by adapting the constraints on the edges of all past hypotheses [10].

3   Proposed Ensemble System

In this study, we construct a new ensemble system by combining the homogeneous ensemble and the heterogeneous ensemble in a single framework. Briefly, we generate the new training sets from the original training data (the homogeneous module) and then train several different learning algorithms on these new training sets to obtain the base classifiers (the heterogeneous module). A class label is assigned to a test sample by combining the outputs of these base classifiers. By doing this, we can get rich diversity from the two types of ensembles: diversity from using different training sets and diversity from using different learning algorithms. Therefore, this is expected to perform better than either of the heterogeneous and homogeneous ensemble methods. There are two questions concerning the proposed ensemble (i) How to generate the new training sets used in the training of different learning algorithms to obtain the base classifiers? (ii) How to aggregate the base classifiers' outputs?

3.1   The homogeneous-heterogeneous ensemble system

We use random projection [11, 12] to generate the new training sets since the projected data is significantly different compared to the original data [2]. Random projection is a projection from a p-dimensional space $\mathbb{R}^p$ (up-space) to a q-dimensional space $\mathbb{R}^q$ (down-space): $\boldsymbol{T} : \mathbb{R}^p \rightarrow \mathbb{R}^q : \boldsymbol{\mathcal{D}}_j = \boldsymbol{T}[\boldsymbol{\mathcal{D}}] \subset \mathbb{R}^q$. The projection $\boldsymbol{T}$ can be represented in the form of matrix $\boldsymbol{R}$ in which each element of the matrix is generated according to a specified random distribution.

During the training phase, K random matrices of size $(p \times q)$ denoted by $\boldsymbol{R}_j(j = 1, ..., K)$ are generated. A random matrix is simply obtained by $\boldsymbol{R}_j = \{r\}$ of size $(p \times q)$, where r are random variables such that $E(r) = 0$ and $Var(r) = 1$. After that, K new training sets $\boldsymbol{\mathcal{D}}_j$ in the down-space (of size $(|\boldsymbol{\mathcal{D}}| \times q)$ are

generated from the original training set $\mathcal{D}$ (of size $|\mathcal{D}| \times p$) through the projection $\boldsymbol{R}_j$ from $\mathcal{D}$ to $\mathcal{D}_j$ which is given by:

$$\mathcal{D}_j = (\mathcal{D}\boldsymbol{R}_j)/\sqrt{q} \tag{1}$$

The T different learning algorithms $\{\mathcal{K}_i\} i = 1, ..., T$ are then trained on each $\mathcal{D}_j$ to obtain the *base classifiers* $h_{i,j}(i = 1, ..., T; j = 1, ..., K)$.

We consider the prediction of the base classifiers $h_{i,j}; i = 1, ..., T; j = 1, ..., K$ on a set $\mathcal{V} = \{\boldsymbol{x}_n, n = 1, ..., N\}$. Each instance $\boldsymbol{x}_n$ in $\mathcal{V}$ is first projected to the down-space by:

$$\tilde{\boldsymbol{x}}_{n,j} = (\boldsymbol{x}_n \boldsymbol{R}_j)/\sqrt{q} \tag{2}$$

The projected data $\tilde{\boldsymbol{x}}_{n,j}$ is fed into classifier $h_{i,j}$ to obtain the prediction. The predictions for the instances in $\mathcal{V}$ are given by:

$$\boldsymbol{L} = \begin{bmatrix} P_{1,1}(y_1|\boldsymbol{x}_1) \ ... \ P_{1,1}(y_M|\boldsymbol{x}_1) \ ... \ P_{1,K}(y_1|\boldsymbol{x}_1) \ ... \ P_{T,K}(y_M|\boldsymbol{x}_1) \\ \vdots \qquad\qquad \ddots \qquad\qquad \vdots \\ P_{1,1}(y_1|\boldsymbol{x}_N) \ ... \ P_{1,1}(y_M|\boldsymbol{x}_N) \ ... \ P_{1,K}(y_1|\boldsymbol{x}_N) \ ... \ P_{T,K}(y_M|\boldsymbol{x}_N) \end{bmatrix} \tag{3}$$

in which $P_{i,j}(y_m|\boldsymbol{x}_n)$ is the prediction that observation $\boldsymbol{x}_n$ belongs to class label y_m given by base classifier $h_{i,j}$. Each row of $\boldsymbol{L}$ is the concatenation of the predictions of all classifiers for one observation. The prediction matrix $\boldsymbol{L}$ (size of $N \times TKM$) is called the meta-data of $\mathcal{V}$.

3.2   Combining Methods

For the homogeneous ensemble, several hundred or thousand of classifiers are generated on the new training sets. Majority Vote rule on a large number of inputs, therefore, is effective for the combining purpose. This makes Majority Vote rule the most popular combining algorithm for the homogeneous ensemble. However, the Majority Vote rule is less effective on the heterogeneous ensemble as the majority on a small set of predictions is unreliable to obtain the final decision. In this study, we introduce four combining methods based on the Sum and Majority Vote rules to combine the output of base classifiers in the proposed ensemble system.

Sum rule: We compute the average on the predictions of all base classifiers given in (3). The Sum rule for the proposed ensemble method is given by:

$$\boldsymbol{x} \in y_u \text{ if } y_u = \operatorname{argmax}_{y_m, m=1,...,M} \frac{1}{TK} \sum_{i=1}^{T} \sum_{j=1}^{K} P_{i,j}(y_m|\boldsymbol{x}) \tag{4}$$

Sum-Majority Vote rule 1: Sum rule is first applied to the predictions associated with each random projection. After this step, we obtain K predictions results for each class label. The Majority Vote rule then is used on these predictions to obtain the final decision.

$$\boldsymbol{x} \in y_u \text{ if } y_u = \operatorname{argmax}_{y_m, m=1,...,M} \sum_{j=1}^{K} \Delta_{j,m}; \Delta_{j,s} = \begin{cases} 1 & \text{if } s = \operatorname{argmax}_{m=1,...,M} \sum_{i=1}^{T} P_{i,j}(y_m|\boldsymbol{x}) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

Sum-Majority Vote rule 2: This is similar to the Sum-Majority Vote rule 1 except the order they are applied. Here Sum rule is used on the predictions associated with each learning algorithm to acquire the T predictions results for each class label. The Majority Vote rule then is used on these predictions to obtain the final decision.

$$\boldsymbol{x} \in y_u \text{ if } y_u = \operatorname{argmax}_{y_m, m=1,\ldots,M} \sum_{i=1}^{T} \Delta_{i,m}; \Delta_{i,s} = \begin{cases} 1 & \text{if } s = \operatorname{argmax}_{m=1,\ldots,M} \sum_{j=1}^{K} P_{i,j}(y_m|\boldsymbol{x}) \\ 0 & \text{otherwise} \end{cases} \quad (6)$$

Majority Vote rule: The Majority Vote rule applied to the prediction of all $(T \times K)$ base classifiers on the sample $\boldsymbol{x}$ is given by:

$$\boldsymbol{x} \in y_u \text{ if } y_u = \operatorname{argmax}_{y_m, m=1,\ldots,M} \sum_{i=1}^{T} \sum_{j=1}^{K} \Delta_{i,j,m}; \Delta_{i,j,s} = \begin{cases} 1 & \text{if } s = \operatorname{argmax}_{m=1,\ldots,M} P_{i,j}(y_m|\boldsymbol{x}) \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

4 Experimental Studies

4.1 Experimental Settings

We selected 24 popular datasets from the UCI database for our experiments. To construct the homogeneous module, we used *Normal*-based random projections $\mathcal{N}(0,1)$ with $q = 2 \times [\log_2(p)]$ [13]. We used three different learning algorithms: Linear Discriminant Analysis, Naïve Bayes, and kNN (k is set to 5) to create the heterogeneous module.

We performed 10-fold Cross-Validation procedure in which each fold a data file is divided into the training data and testing data. The Cross-Validation procedure was run 3 times so that we obtained 30 test results on each dataset from which to calculate the mean and variance of classification error rate.

4.2 Influence of parameters

Different number of random projections We examine the influence of using a different number of random projections on the performance of the proposed ensemble and which are the most suitable combining algorithms for the proposed ensemble. In this study, we used 10, 50, and 100 random projections in the homogeneous module (see Fig 1). Some observations can be made:

- Sum-Majority Vote rule 2 is the poorest combining method in our experiment. Sum-Majority Vote rule 2 uses Sum rule on the outputs associated with random projections (up to hundred) and then uses Majority Vote rule on the outputs of Sum rule (only 3). Because of voting on a small set of results, the Majority Vote rule results in poor performance. In fact, the classification error rates of Sum-Majority Vote rule 2 are usually higher than those of the other combining methods.
- Majority Vote rule has average performance in the experiment. However, on some datasets like Iris and Fertility, this method obtains the lowest classification error rates among all four combining methods.

– Sum rule and Sum_Majority Vote rule 1 are the best combining methods for the proposed ensemble as their performance is usually better than those of the other combining methods.
– A common trend in this figure is the reduction of classification error rate when increasing the number of random projections in the homogeneous module. However, there are exceptional cases on datasets like Led7digit.

In the next section, we used Sum_Majority Vote rule 1 as the combining algorithm for the proposed ensemble (with 100 random projections and 3 learning algorithms to obtain 300 classifiers) when comparing to the benchmark algorithms.

4.3   Comparison to benchmark algorithms

We selected three homogeneous ensemble methods namely Random Subspace, TotalBoost, and RotBoost as the benchmark algorithms (using 300 base classifiers). For RotBoost, we used 10 Rotation Forest [9] and 30 classifiers in AdaBoost to create 300 classifiers. Besides, we chose one well-known combining algorithms for heterogeneous ensemble systems namely the Decision Template method [7] for the comparison. We used the Friedman test to compare the results of all methods on all experimental datasets. Since p-value of this test is smaller than the pre-selected significant level of 0.05, we rejected the null hypothesis that all methods perform equally. We then run the Nemenyi post-hoc test to perform pairwise comparison of all methods. Some observations can be made from the test results in Table 1 and Fig 2.

– Random Subspace ranks the second with rank value 2.94. Random Subspace randomly selects features from the feature space and then generate new training data that is associated with the selected features. In fact, this method normally is outstanding for high dimensional datasets like Libras (90 features) and Sonar (60 features) in the experiment.
– Decision Template method has an average performance and is worse than the proposed method based on the Nemenyi test result. The study in [6] showed that Decision Template method may not provide good representation for the meta-data, resulting in poorly performance on some datasets.
– RotBoost and TotalBoost are two poorest methods in our experiment and are worse than the proposed ensemble based on the Nemenyi test result.
– The proposed ensemble ranks the first with rank values 1.7. In detail, the proposed ensemble ranks first on 9 datasets (37.5%) and ranks second on 14 datasets (58.33%).

We note that some different approaches can be used to construct the homogeneous module for particular problems. For example, we can use random subspace technique to produce new training sets when working with high-dimension data. The heterogeneous module meanwhile can be customized by changing the learning algorithms that are used to produce the base classifiers. By this way, the proposed framework has great flexibility when applied to real-world applications.
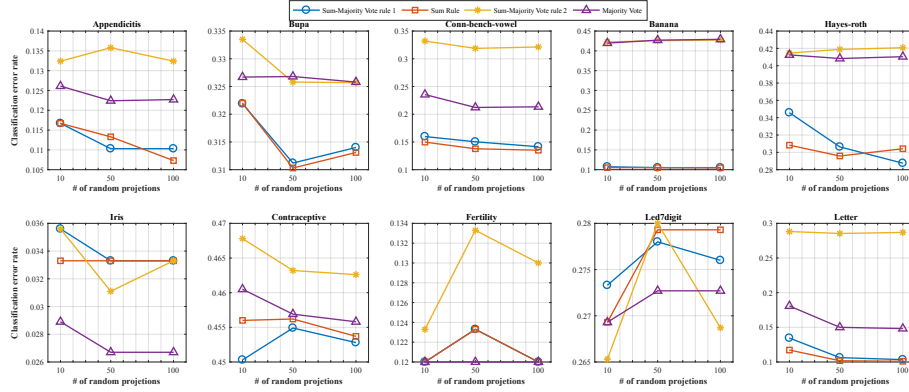
Fig. 1: The classification error rate of the proposed ensemble system with 4 combining algorithms on 10 datasets
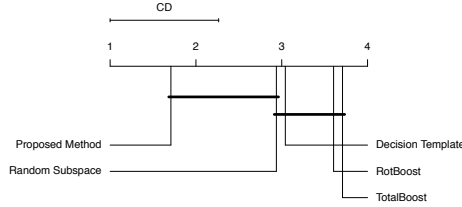

Fig. 2: The Nemenyi test result

Table 1: The mean and variance of classification error rates of the benchmark algorithms and the proposed ensemble

	Proposed Ensemble		Decision Template		Random Subspace		TotalBoost		RotBoost	
	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance	Mean	Variance
Appendicitis	0.1103 (1)	7.79E-03	0.1297 (2)	8.24E-03	0.1418 (5)	8.14E-03	0.1358 (4)	6.40E-03	0.1324 (3)	6.41E-03
Artificial	0.2395 (2)	1.68E-03	0.2443 (3)	1.62E-03	0.2714 (4)	2.60E-03	0.2295 (1)	1.67E-03	0.3300 (5)	2.83E-03
Balance	0.1082 (2)	3.14E-04	0.0960 (1)	9.38E-04	0.2134 (5)	1.46E-03	0.1280 (3)	2.11E-03	0.1547 (4)	1.18E-03
Banana	0.1052 (1)	8.33E-05	0.1117 (2)	1.36E-04	0.3848 (5)	2.74E-03	0.1259 (4)	1.22E-04	0.1173 (3)	2.78E-04
Blood	0.2286 (2.5)	1.28E-03	0.2674 (4)	2.34E-03	0.2228 (1)	6.55E-04	0.3160 (5)	2.92E-03	0.2286 (2.5)	5.94E-04
Breast-Tissue	0.3600 (2)	1.20E-02	0.3858 (4)	1.07E-02	0.3333 (1)	6.75E-03	0.3752 (3)	1.34E-02	0.3955 (5)	1.76E-02
Bupa	0.3140 (1)	3.37E-03	0.3324 (3.5)	5.18E-03	0.3324 (3.5)	5.23E-03	0.3150 (2)	5.42E-03	0.3390 (5)	4.38E-03
Conn-bench-vowel	0.1414 (2)	2.43E-03	0.1938 (4)	2.50E-03	0.0594 (1)	8.83E-04	0.1655 (3)	2.40E-03	0.3611 (5)	6.28E-03
Contraceptive	0.4528 (1)	1.19E-03	0.4662 (3)	1.73E-03	0.4777 (4)	8.59E-04	0.5234 (5)	2.07E-03	0.4653 (2)	7.14E-04
Fertility	0.1200 (2)	1.60E-03	0.4200 (5)	2.29E-02	0.1200 (2)	1.60E-03	0.1767 (4)	9.79E-03	0.1200 (2)	1.60E-03
Haberman	0.2638 (1)	2.15E-03	0.3102 (4)	3.90E-03	0.2982 (3)	3.25E-03	0.3718 (5)	5.23E-03	0.2810 (2)	2.26E-03
Hayes-roth	0.2875 (2)	1.24E-02	0.4042 (5)	2.05E-02	0.3417 (3)	1.82E-02	0.1854 (1)	1.08E-02	0.3688 (4)	1.75E-02
Iris	0.0333 (1.5)	1.41E-03	0.0333 (1.5)	1.41E-03	0.0444 (4)	2.77E-03	0.0667 (5)	3.81E-03	0.0378 (3)	1.68E-03
Led7digit	0.2760 (2)	4.17E-03	0.2700 (1)	4.37E-03	0.4533 (5)	2.38E-03	0.2993 (4)	4.73E-03	0.2853 (3)	7.09E-03
Letter	0.1034 (2)	6.07E-05	0.1192 (3)	5.99E-05	0.0994 (1)	5.33E-05	0.3599 (5)	7.03E-04	0.1733 (4)	1.47E-04
Libras	0.2639 (2)	2.97E-03	0.2991 (4)	4.36E-03	0.2000 (1)	2.39E-03	0.2972 (3)	4.02E-03	0.4722 (5)	7.61E-03
Marketing	0.6662 (1)	1.29E-04	0.6978 (4)	2.00E-04	0.6705 (3)	1.16E-04	0.7083 (5)	4.39E-04	0.6701 (2)	2.14E-04
Sonar	0.1808 (2)	6.43E-03	0.2292 (4)	5.77E-03	0.1379 (1)	4.84E-03	0.1856 (3)	7.28E-03	0.2387 (5)	6.43E-03
Twonorm	0.0229 (2)	2.69E-05	0.0211 (1)	1.93E-05	0.0271 (3)	3.10E-05	0.0330 (4)	5.49E-05	0.0375 (5)	6.91E-05
Vertebral	0.1807 (1)	3.37E-03	0.1914 (4)	3.88E-03	0.2763 (5)	2.48E-03	0.1817 (2)	2.53E-03	0.1839 (3)	3.76E-03
Waveform_w_noise	0.1673 (2)	2.01E-04	0.1634 (1)	2.00E-04	0.1705 (3)	2.64E-04	0.1781 (5)	2.98E-04	0.1744 (4)	2.54E-04
Waveform_wo_noise	0.1568 (3)	3.98E-04	0.156 (2)	3.36E-04	0.1518 (1)	3.19E-04	0.1834 (5)	2.19E-04	0.1635 (4)	3.78E-04
Wine_red	0.4230 (2)	1.26E-04	0.5157 (5)	1.16E-03	0.3112 (1)	3.80E-04	0.4294 (4)	1.52E-04	0.4255 (3)	6.63E-04
Yeast	0.3971 (1)	8.78E-04	0.4182 (2)	7.85E-04	0.4926 (5)	1.96E-03	0.4771 (4)	1.44E-03	0.4499 (3)	2.67E-03
Average ranking	1.71		3.04		2.94		3.71		3.6	

*(,) indicates the ranking of a method on each dataset

5   Conclusions

In this work, we have introduced a design for ensemble systems by combining homogeneous module and heterogeneous module in a single framework by using random projections and different learning algorithms. The random projections are applied on the training set to generate many training set schemes. Several different learning algorithms then train classifiers on these new schemes. We proposed four combining algorithms to combine the outputs of these classifiers based on Sum Rule and Majority Vote Rule. Experiments on some well-known datasets show that the proposed ensemble system significantly outperformed several well-known benchmark algorithms. The proposed design is general, that means any techniques that make rich training data can be used for the homogeneous module, as well as any set of learning algorithms can be used for the heterogeneous module.

References

1. T.T. Nguyen, A.W.C. Liew, M.T. Tran, X.C. Pham, M.P. Nguyen, A novel genetic algorithm approach for simultaneous feature and classifier selection in multi classifier system, in IEEE CEC, 2014, pp. 1698-1705.
2. T.T. Nguyen, M.T. Dang, A.W.C. Liew, J.C. Bezdek, A weighted multiple classifier framework based on Random Projection, Information Sciences. 490 (2019), 36-58.
3. J. Kittler, M. Hatef, R.P.W. Duin, J. Matas, On Combining Classifiers, IEEE Transactions on Pattern Analysis and Machine Intelligence. 20(3) (1998), 226-239.
4. Y. Zhang, G. Cao, B. Wang, X. Li, A novel ensemble method for k-nearest neighbor, Pattern Recognition. 85 (2019), 13-25.
5. T.T. Nguyen, M.T. Dang, D. Pham, L.P. Dao et al., Deep Heterogeneous Ensemble, Aust. J. Intell. Inf. Process. Syst. 16(1) (2019), 1-9.
6. T.T. Nguyen, M.P. Nguyen, X.C. Pham, A.W.C. Liew, W. Pedrycz, Combining heterogeneous classifiers via granular prototypes, Applied Soft Computing. 73 (2018), 795-815.
7. L.I. Kuncheva, J.C. Bezdek, R.P.W. Duin, Decision templates for multiple classifier fusion: an experimental comparison, Pattern Recognition. 34 (2001), 299-314.
8. T.T. Nguyen, T.T.T. Nguyen, X.C. Pham, A.W.C Liew, A novel combining classifier method based on Variational Inference, Pattern Recognition. 49 (2016), 198-212
9. C.X. Zhang, J.S. Zhang, RotBoost: A technique for combining Rotation Forest and AdaBoost, Pattern Recognition Letters. 29(10) (2008), 1524-1536.
10. M.K. Warmuth, J. Liao, G. Ratsch. Totally corrective boosting algorithms that maximize the margin, in Proceeding of ICML, 2006, pp. 1001–1008, 2006.
11. M.T. Dang, A.V. Luong, T.-T. Vu et al., An Ensemble System with Random Projection and Dynamic Ensemble Selection, in Proceeding of ACIIDS, 2018, pp 576-586
12. W. Johnson, J. Lindenstrauss, Extensions of Lipschitz mapping into Hilbert space, Conference in modern analysis and probability, vol. 26 of Contemporary Mathematics, American Mathematical Society, 1984, pp. 189-206.
13. X.C. Pham, M.T. Dang, S.V. Dinh, et al., Learning from Data Stream Based on Random Projection and Hoeffding Tree Classifier, in Proceeding of DICTA, 2017.