Confidence-based voting procedure for combining fuzzy systems and neural networks

V Stanovov¹, Sh Akhmedova¹ and Y Kamiya²

¹Reshetnev Siberian State University of Science and Technology, Krasnoyarsk, Russian Federation
²Aichi Prefectural University, Nakagute, Japan

E-mail: vladimirstanovov@yandex.ru

Abstract. In this study the confidence-based voting of neural net classifier and fuzzy logic based classifiers is proposed. In this method, for the cases when the fuzzy system is confident enough in its decision, i.e. when the membership value is large enough, fuzzy system makes the decision, otherwise, the neural net is applied. This allows classifying most of the objects by explainable interpretable fuzzy system, while using the more accurate neural network for the most difficult cases. The experiments are performed on a set of test datasets, and two problems of identifying the emotional state of a person using the data collected by non-contact vital Doppler sensors. The results show that this setup allows not only improving the classification quality, but also allows to explain the classification process the explanation of the classifier functioning.

1. Introduction
Nowadays the fuzzy systems and the neural networks have found a variety of applications in many areas of human life, starting from industry and financial sphere, up to medicine and social sciences. Fuzzy systems and neural networks are one of the most important computational intelligence technologies for machine learning and data mining. However, each of them has serious disadvantages; for example, neural nets are often criticized for their low interpretability. On the other hand, the fuzzy systems, while usually not so precise, represent the “white box”, i.e. their functioning could be easily understood by human experts.

Combining different machine learning techniques for solving classification, regression, clustering or other problems is usually based on some implementation of voting. The voting leads to increasing of generalization in most scenarios, thus leading to higher accuracy. While solving classification problems the easiest way of ensemble generation is the majority voting, however, other schemes exist, including bagging [1], boosting, [2, 3, 4], random forest [5], stacking [6]. Among boosting methods the following should be mentioned: Adaboost [7], gradient boosting [8] and random subspace method [9]. Most efficient ensemble techniques use some sort of classifier weighting.

In this study the novel weighting technique is proposed, which is based on the membership values, which are returned together with the class label in fuzzy classifiers. Based on these values, fuzzy systems should make decisions regarding classification; otherwise the neural networks could be used.

The experiments are performed on 10 problems from the KEEL machine learning repository, as well as on the real-world data, collected by using non-contact vital sensing with Doppler Effect. The labeling of the real-world data described the emotional state of a person while listening to music.
The rest of the paper is organized as follows: section two describes the hybrid evolutionary fuzzy classification algorithm used in this study, as well as the neural network architecture and training parameters. Section three contains description of the test and real-world datasets, section four contains the experimental setup and results, and section five concludes the paper.

2. Training fuzzy logic system and neural network

2.1. Fuzzy classification

The fuzzy set theory has found many applications in the area of machine learning, mostly for supervised and unsupervised problems solving. In most cases, these systems are referred to as Evolutionary Fuzzy Systems [10]. Among them, the fuzzy models consisting of a set of fuzzy rules are known as Fuzzy Rule-Based Systems (FRBS). In most of the studies FRBS are designed by using an implementation of a genetic algorithm (GA) [11] with specific crossover and mutation operators, this class of algorithms is usually referred to as Genetic Fuzzy Systems (GFS) [12]. The GAs are employed due to their ability to deal with large search spaces efficiently. Also, many algorithm versions incorporate a priori knowledge, or extract them from the data at hand.

The flexibility of fuzzy systems allows applying them in various areas; however, there are two main trends in fuzzy systems development. The first is usually called interpretable fuzzy systems, and it focuses on creating fuzzy systems capable of building fuzzy rules, which are easy to understand for human experts in the area of interest. These systems are usually simpler, but lead to larger error values. The second trend is called accurate fuzzy systems, and focuses on generating more complex fuzzy systems, which are not always interpretable, but capable of making precise predictions. Obviously, depending on the application area, one of these types of fuzzy systems should be considered, but a good tradeoff between them is often desirable.

In this study, the Hybrid Evolutionary Fuzzy Classification Algorithm (HEFCA) is considered, as it has shown promising results for solving complex problems [13]. The HEFCA algorithm builds rules of the following form:

\[ R_q: \text{IF } x_1 \text{ is } A_{q1} \text{ and } \ldots \text{ and } x_n \text{ is } A_{qn} \text{ then Class } C_q \text{ with } CF_q, \]

where \( R_q \) is the \( q \)-th fuzzy rule, \( x = (x_1, \ldots, x_m) \) is the set of \( m \) training patterns in \( n \)-dimensional space, \( A_{qi} \) is a fuzzy set for the \( i \)-th variable, \( C_q \) is the class number, and \( CF_q \) is the class weight. The training sample has been previously normalized to \([0, 1]\). The product operator was used to calculate membership value for each pattern.

To improve accuracy of the system and keep the interpretability at a high level, four fuzzy granulations into 2, 3, 4 and 5 fuzzy terms were used, shown in figure 1. In addition, to make the rules simpler, the “Don’t Care” condition was used.

![Figure 1. Fuzzy granulation into 2, 3, 4 and 5 terms.](image)
The HEFCA algorithm starts by generating rules based on the heuristic rule generation procedure using the training sample. Next, the main loop proceeds as follows:

- sample-based initialization;
- selection (tournament or rank-based);
- crossover;
- mutation (3 levels);
- Michigan part (genetic or heuristic);
- operator probability adaptation;
- stopping criterion check, return to step 2 (the number of generations left).

The quality of each generated rule was estimated using the confidence value:

\[ Conf(A_q \rightarrow \text{Class } k) = \frac{\sum_{x_p \in \text{Class } k} \mu_{A_q}(x_p)}{\sum_{m=1}^{n} \mu_{A_q}(x_p)}, \]

where \( A_q \) is the \( q \)-th rule left part, \( k \) is the class number, \( \mu_{A_q}(x_p) \) is the membership value for the input value \( x_p \). The class number corresponding to the newly generated rule was determined as the class having the highest confidence. The weight of each rule was estimated as:

\[ CF_q = 2 \cdot Conf(A_q \rightarrow \text{Class } k) - 1, \]

so that the confidence of 1 is transformed to the weight equal to 1, and the confidence of 0.5 – to zero weight.

Unlike the previous works [14], in this version of HEFCA, the loss value was applied in fitness estimation. Here the fitness of an individual is calculated as follows:

\[ \text{Fitness}_i = \frac{5000 \cdot \text{Error}_i}{N} + \frac{5000 \cdot \text{Loss}_i}{N} + NR_i + \text{Len}_i, \]

where \( \text{Error}_i \) is the number of incorrectly classified individuals, \( NR_i \) is the number of rules. Here \( \text{Loss}_i \) is the total number of non-empty predicates in all rules, \( i = 1, \ldots, NP \), where \( NP \) is the population size. Besides, \( N \) is the sample size, and \( \text{Loss}_i \) is calculated as:

\[ \text{Loss}_i = \sum_{i=0}^{N} 1 - \mu_{A_w}(x_i) \cdot CF_w, \]

where \( w \) is the index of the winner rule, i.e. the rule having the largest weighted membership value. Adding the loss value to the fitness allows the algorithm to be sensible not only to the number of correctly or incorrectly classified instances, but also to how confident the classifier is about its final decision.

The training of the HEFCA algorithm was further improved by an instance selection method, using the balancing strategy, described in [14]. The instance selection creates a subsample of the original training sample, in which the number of instances belonging to each class are as balanced as possible, to prevent the negative effects of imbalanced datasets and speed up the search process. Each instance in the training set received probability to be chosen depending on how difficult it was to classify in previous adaptation periods when the subsample changed. More detailed description of the HEFCA algorithm could be found in [14].

2.2. Classifying with Neural Net models

Nowadays, the research in the area of training neural networks has reached the level when most of the known architectures are implemented in well-developed libraries. In this study the Keras library was used with Tensorflow backend, to build a two-layered feed-forward neural net with sigmoid activation functions and a softmax layer. The categorical cross-entropy was used as the loss function, and the adaptive momentum (ADAM) method with Nesterov momentum was applied.
The parameters of the training algorithm for test datasets were as follows: learning rate was equal to 0.003, $beta_1$ and $beta_2$ were set to 0.95 and 0.999 respectively, and decay was equal to 0.005. The number of neurons in the first hidden layer was 100, in the second hidden layer – 50. The number of epochs was set to 25, batch size 100. For the real-world problems the learning rate was set to 0.03, number of neurons for the hidden layer – 20, only one hidden layer was used. The described setup represents a typical scenario of neural networks application in practice.

3. Proposed approach

The presented classification methods return as a result not only the class number, but also some measure of how sure the classifier is about its decision. The fuzzy logic classifier determines winner-rule by comparing the weighted membership values, so that the winner rule’s confidence $\mu_{A_w}(x_i) CF_w$ changes in range $(0,1]$, where 1 means that the rule fully describes an instance, while a value close to 0 means that among all rules present in the rule base, even the winner rule is not sure about the class number. In fuzzy classifiers the situation of rejected classification, when no rule gave non-zero membership value may also happen, however, the HEFCA algorithm generated such rule bases relatively rare.

On the other hand, the neural net with softmax layer returns a set of values, usually considered as probability of an instance belonging to each class. The softmax values are calculated as:

$$\sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}}$$

where the vector $z$ is the input to the softmax layer, $k$ is the number of classes, $i = 1, \ldots, k$. From this equation we may see that the case of complete unawareness of the classifier about an instance class represents the situation when all $\sigma(z)_i$ are equal to each other. Unlike fuzzy logic, the neural net will always return some probability values, even for instances which are very different from anything seen before, i.e. the classical neural net does consider the rejected classification situation.

To use both of these approaches, the following algorithm of combining FL and NN could be considered. If the confidence of a FL classifier is lower than a predefined level, the NN classifier is used. The main idea here is that in most cases the neural classifier has higher accuracy, but the fuzzy classifier has better interpretability.

For experiments in this study, the following rule was used: after estimating the class numbers and membership values of FL system on test sample, for 25% of instances, which have the smallest membership value, the result of the neural classifier was used. This method will be further referred to as the Confidence-Based Voting (CBV). In the next section the experimental setup and results are presented.

4. Experimental setup and results

To validate the proposed approach, the set of databases were used from the KEEL repository, with the characteristics shown in table 1.

| Dataset         | Number of instances | Number of features | Number of classes |
|-----------------|---------------------|--------------------|-------------------|
| Australian credit | 690                 | 14                 | 2                 |
| German credit   | 1000                | 24                 | 2                 |
| Magic           | 19020               | 10                 | 2                 |
| Page-blocks     | 5472                | 10                 | 5                 |
| Penbase         | 10992               | 16                 | 10                |
| Phoneme         | 5404                | 5                  | 2                 |
| Ring            | 7400                | 20                 | 2                 |
| Satimage        | 6435                | 36                 | 6                 |
The datasets in table 1 have different characteristics and are taken from different fields, some of them have large number of classes, others are characterized by class imbalance, large number of instances, which make these problems difficult to solve for most methods.

For the fuzzy rule base training with HEFCA algorithm, the following parameters were used: number of individuals 100, number of generations 5000, maximum number of rules 40, selection, mutation and Michigan part operators were self-configured; for instance, the adaptation period was set to 50 generations, the percent of instances used in the training subsample 30%.

For every dataset the 10-fold cross-validation was performed, and a set of classification quality measures were calculated, including accuracy (rate of correctly classified instances), $F$-measure for every class (for 2-class problems) and averaged confusion matrix. These measures were estimated for fuzzy logic and neural net classifiers separately, and the proposed combined approach.

Table 2 presents the results for the test classification problems.

| Dataset       | HEFCA | NN (Keras) | CBV  |
|---------------|-------|------------|------|
| Australian credit | 0.8537 | 0.8695 | 0.8551 |
| German credit | 0.7280 | 0.7610 | 0.7310 |
| Magic | 0.8415 | 0.8284 | 0.8362 |
| Page-blocks | 0.9406 | 0.9589 | 0.9461 |
| Pen-based | 0.9395 | 0.9874 | 0.9790 |
| Phoneme | 0.8260 | 0.7972 | 0.8225 |
| Ring | 0.9236 | 0.8259 | 0.9008 |
| Satimage | 0.8636 | 0.7640 | 0.8244 |
| Segment | 0.9100 | 0.9498 | 0.9429 |
| Texture | 0.9122 | 0.9653 | 0.9356 |
| Twonorm  | 0.9043 | 0.9805 | 0.9305 |

The marked values in table 2 represent the cases when there was an accuracy improvement of CBV algorithm compared to the HEFCA results. This was the case for 7 datasets out of 11, for the rest 4 the neural net had significantly worse results, so the combination did not give any improvement.

Table 3 contains the $F$-measures values for 2-class datasets. In terms of $F$-measure, the NN classifier in most cases outperformed the fuzzy logic classifier on the first class. This is due to the fact that the first class in imbalanced datasets usually had larger number of instances.

This lead to the result when the $F$-measure for the first class improved for almost all cases, except for the artificial Ring problem, where the NN was much worse. At the same time, for the second class, the $F$-measure value was also improved compared to the results of NN, although still worse than the results of fuzzy logic alone.

| Dataset       | HEFCA | NN (Keras) | CBV  |
|---------------|-------|------------|------|
| Australian credit | 0.567  | 0.686 | 0.585 | 0.680 | 0.571 | 0.684 |
| German credit | 0.793  | 0.304 | 0.811 | 0.247 | 0.800 | 0.272 |
| Magic | 0.773  | 0.430 | 0.776 | 0.404 | 0.777 | 0.409 |
| Phoneme | 0.802  | 0.394 | 0.808 | 0.334 | 0.805 | 0.379 |
| Ring | 0.645  | 0.653 | 0.614 | 0.631 | 0.636 | 0.649 |
| Twonorm  | 0.646  | 0.642 | 0.663 | 0.662 | 0.652 | 0.649 |
In Table 4 the averaged confusion matrices of HEFCA, NN and CBV are presented for the Page-blocks dataset.

**Table 4.** Confusion matrices of HEFCA, NN and CBV for the Page-blocks dataset.

| HEFCA | Actual | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Reject |
|-------|--------|---------|---------|---------|---------|---------|---------|
| Predicted | | 488.0 | 2.9 | 0 | 0.1 | 0.3 | 0 |
|         | | 12.8 | 19.6 | 0 | 0.5 | 0 | 0 |
|         | | 2.2 | 0 | 0.6 | 0 | 0 | 0 |
|         | | 2.7 | 0.2 | 0 | 5.8 | 0 | 0 |
|         | | 10.4 | 0.1 | 0.3 | 0 | 0.7 | 0 |

| NN (Keras) | Actual | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Reject |
|------------|--------|---------|---------|---------|---------|---------|---------|
| Predicted | | 484.1 | 3.2 | 0.1 | 0.4 | 3.5 | 0 |
|           | | 3.5 | 28.1 | 0 | 0.8 | 0.5 | 0 |
|           | | 2.7 | 0 | 0 | 0 | 0.1 | 0 |
|           | | 0.9 | 0 | 0 | 7.3 | 0.5 | 0 |
|           | | 6.1 | 0.1 | 0.1 | 0 | 5.2 | 0 |

| CBV | Actual | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Reject |
|-----|--------|---------|---------|---------|---------|---------|---------|
| Predicted | | 486.9 | 2.7 | 0.1 | 0.2 | 1.4 | 0 |
|         | | 10.2 | 22.1 | 0 | 0.5 | 0.1 | 0 |
|         | | 2.8 | 0 | 0 | 0 | 0 | 0 |
|         | | 2.6 | 0.2 | 0 | 5.9 | 0 | 0 |
|         | | 8.4 | 0.2 | 0.1 | 0 | 2.8 | 0 |

The marked values in the last confusion matrix demonstrate the cases when there was an improvement in correct classification. It should be mentioned that FL and NN made different misclassification errors, for example, the NN performed better on classes 2, 4 and 5, while the FL was better on classes 1 and 3. The CBV model was able to find a better balance between the two used classifiers.

In table 5 the averaged confusion matrices of HEFCA, NN and CBV are presented for the Segment dataset. For the Segment dataset with 7 classes, due to high performance of the neural net, for CBV the improvement in terms of number of correctly classified instances was demonstrated on 4 classes, and a small deterioration of performance was only on the first class. This shows that in such scenarios the CBV model allows significantly improving the performance.

**Table 5.** Confusion matrices of HEFCA, NN and CBV for the Segment dataset.

| HEFCA | Actual | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Class 7 | Reject |
|-------|--------|---------|---------|---------|---------|---------|---------|---------|---------|
| Predicted | | 32.7 | 0 | 0.1 | 0.2 | 0 | 0 | 0 | 0 |
|         | | 0 | 33 | 0 | 0 | 0 | 0 | 0 | 0 |
|         | | 0.3 | 0.2 | 29.1 | 1.5 | 1.7 | 0.1 | 0.1 | 0 |
|         | | 1.6 | 0 | 0.7 | 27.4 | 1.2 | 2.1 | 0 | 0 |
|         | | 1.5 | 0 | 6.3 | 2.6 | 22.3 | 0.3 | 0 | 0 |
|         | | 0 | 0 | 0 | 0 | 33 | 0 | 0 | 0 |
|         | | 0.1 | 0 | 0.1 | 0 | 0.1 | 32.7 | 0 | 0 |

| NN (Keras) | Actual | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Class 7 | Reject |
|------------|--------|---------|---------|---------|---------|---------|---------|---------|---------|
| Predicted | | 32.6 | 0 | 0 | 0.4 | 0 | 0 | 0 | 0 |
|           | | 0 | 32.8 | 0 | 0.2 | 0 | 0 | 0 | 0 |
|           | | 0 | 0 | 29.9 | 1 | 2.1 | 0 | 0 | 0 |
|           | | 0.1 | 0 | 0 | 29.5 | 2.7 | 0.7 | 0 | 0 |
To test the CBV algorithm on real data, two datasets were collected. In the experiments with vital sensors based on Doppler Effect [16] human’s reaction while listening to music was measured. Ten people of different gender and age were asked to participate in experiments, and three states were tested:

- listening to music that a participant admitted to like in three different time periods;
- listening to music that a participant admitted to dislike in three different time periods;
- not listening to music at all in two different time periods.

The signal processing was performed using the ARS technique described in [17]. For each person 8 instances were obtained by pre-processing the raw data, resulting in 4 attributes, namely the deviation of the respiratory rate, volt, and the average value of the respiratory rate and the variance of the respiratory rate.

**Table 6.** Real-world datasets used in the experiments.

| Problem | Classes | Dataset sizes |
|---------|---------|---------------|
| “Listened” | 1 – listened to music | 1: 20 instances |
| | 2 – did not listen to music | 2: 60 instances |
| | | whole set: 80 instances |
| “Liked” | 1 – liked music | 1: 30 instances |
| | 2 – did not like music | 2: 30 instances |
| | | whole set: 60 instances |

The data was normalized and two classification problems were formulated, the properties are presented in table 6.

- “Listened” problem, each data sequence was labelled as “1” if the participant listened to the music and “0” otherwise;
- “Liked” problem, where each data sequence was labelled as “1” if the participant liked the music and “0” otherwise.

For the experiments on the real data, the parameters of HEFCA were changed, so that the maximum number of generations was set to 500 instead of 5000. For the neural net, only 20 nodes were used in a single hidden layer, and 5 epochs of training. The accuracy, F-measure and confusion matrix were calculated for both problems. table 7 contains the results for both datasets.

**Table 7.** Accuracy and F-measures of HEFCA, NN and CBV on real-world problems.

| Dataset | HEFCA | NN (Keras) | CBV |
|---------|-------|------------|-----|
| Liked   | 0.616 | 0.550      | 0.600 |
| Listened| 0.613 | 0.750      | 0.688 |

| Dataset | HEFCA | NN (Keras) | CBV |
|---------|-------|------------|-----|
|          | Class 1 | Class 2 | Class 1 | Class 2 | Class 1 | Class 2 |
| 0.1       | 0       | 2.7      | 0.9    | 28.9    | 0.4    | 0      |
| 0         | 0       | 0.2      | 0      | 32.8    | 0      | 0      |
| 0         | 0       | 0.1      | 0      | 32.9    | 0      | 0      |

| Predicted | Actual |
|-----------|--------|
| Class 1   | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | Class 7 | Reject |
| 32.6      | 0       | 0       | 0       | 0.4     | 0       | 0       | 0 |
| 0         | 33      | 0       | 0       | 0       | 0       | 0       | 0 |
| 0         | 0       | 0.1     | 29.8    | 1.8     | 1.3     | 0       | 0 |
| 0.5       | 0       | 0       | 29.1    | 2.5     | 0.9     | 0       | 0 |
| 0.8       | 0       | 3.4     | 0.9     | 27.4    | 0.5     | 0       | 0 |
| 0         | 0       | 0       | 0       | 33      | 0       | 0       | 0 |
| 0         | 0       | 0       | 0       | 0       | 0.1     | 32.9    | 0 |
For the “Liked” problem, in terms of accuracy, the CBV was not able to demonstrate any performance improvement, due to the low accuracy on this problem. The reason is probably due to the small amount of data available for training. For the Listened problem, the Neural Net had better overall classification; however, the F-measure value equal to 0 demonstrates that it did not recognize the first class in the test sample, while the FL did. Combining the methods with CBV allowed increasing the F-measure for the second class, however, at a cost of decreasing it on the first class.

Tables 8 and 9 contain the confusion matrices for the “Liked” and “Listened” problems.

**Table 8.** Confusion matrices of HEFCA, NN and CBV for “Liked” dataset.

| HEFCA   | Actual       |          |          |          |
|---------|--------------|----------|----------|----------|
|         | Class 1      | Class 2  | Reject   |
| Predicted | 2.1          | 0.9      | 0        |
|         | 1.4          | 1.6      | 0        |
| NN (Keras) | Actual      |          |          |          |
| Class 1  | 1.6          | 1.4      | 0        |
|         | 1.3          | 1.7      | 0        |
| CBV     | Actual      |          |          |          |
| Class 1  | 1.9          | 1.1      | 0        |
|         | 1.3          | 1.7      | 0        |

**Table 9.** Confusion matrices of HEFCA, NN and CBV for “Listened” dataset.

| HEFCA | Actual       |          |          |          |
|-------|--------------|----------|----------|----------|
|       | Class 1      | Class 2  | Reject   |
| Predicted | 1.0          | 1.0      | 0        |
|         | 2.1          | 3.9      | 0        |
| NN (Keras) | Actual      |          |          |          |
| Class 1  | 0.0          | 2.0      | 0        |
|         | 0.0          | 6.0      | 0        |
| CBV    | Actual      |          |          |          |
| Class 1  | 0.7          | 1.3      | 0        |
|         | 1.2          | 4.8      | 0        |

Compared to the HEFCA algorithm, the CBV allowed improving the classification quality, which could be seen in better recognition of the second class. For the case of “Listened” dataset, again, there was an improvement in classification accuracy for the second class, as it was easier for the neural net to classify all instances to the second class due to class imbalance. Still, the overall accuracy of the classifier has improved.

**5. Conclusions**

In this paper the confidence-based voting procedure for two classifiers was proposed. The neural net and the fuzzy logic system have been chosen as basic learners, due to different properties: the first is often more accurate, while the second is capable of generating explainable classification rules. The CBV method is applied so that the most complicated instances, where the FL is not sure about the class number, are classified by the NN model, so that the overall accuracy could be improved. The experiments on several test datasets have shown that the application of the CBV method leads to significant performance boost in cases when the neural net performs better than the fuzzy logic-based
classifier. The threshold value, defining the percentage of the sample classified by NN could be tuned to get better performance. For the real-world problems, the CBV method allowed increasing the classification accuracy of the second class; however, due to low performance of the NN on these datasets because of their small size, the improvement was not as significant as on the larger test datasets. The presented CBV method represents a general scheme of combining several classifiers, so not only FL and NN could be applied. Further directions of work may include testing different classification methods in combination with NN, or introducing schemes of multiple classifiers voting based on their confidence.

References
[1] Breiman L 1996 Bagging predictors, Machine learning 24 2 123-40
[2] Freund Y 1997 A decision-theoretic generalization of on-line learning and an application to boosting, Journal of computer and system sciences 55 1 119-39
[3] Quinlan J R 1996 Bagging, boosting, and C4. 5, AAAI/IAAI 1 725-30
[4] Kearns M J 1988 Thoughts on hypothesis boosting ML class project 319-20
[5] Breiman L 2001 Random forests Machine learning 45 1 5-32
[6] Wolpert D H 1992 Stacked generalization Neural networks 5 2 241-59
[7] Schapire R E 2003 The boosting approach to machine learning: an overview Nonlinear estimation and classification (Springer NY) pp 149-71
[8] Friedman J H 2001 Greedy function approximation: a gradient boosting machine Annals of statistics 1189-232
[9] Skurichina M and Duin R P W 2002 Bagging, boosting and the random subspace method for linear classifiers Pattern Analysis & Applications 5 2 121-35
[10] Fazzolari M, Alcala R, Nojima Y, Ishibuchi H and Herrera F 2013 A review of the application of multiobjective evolutionary fuzzy systems: current status and further directions IEEE Transactions on Fuzzy Systems 21 1 45-65
[11] Holland J H 1992 Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence, MIT Press
[12] Herrera F 2008 Genetic fuzzy systems: taxonomy, current research trends and prospects, Evol. Intell 1 27-46
[13] Stanovov V, Semenkin E and Semenkina O 2015 Self-configuring hybrid evolutionary algorithm for fuzzy classification with active learning IEEE Congress on evolutionary computation CEC 2015 1823-30
[14] Stanovov V, Semenkin E and Semenkina O 2016 Self-configuring hybrid evolutionary algorithm for fuzzy imbalanced classification with adaptive instance selection Journal of Artificial Intelligence and Soft Computing Research 6 3 173-88
[15] Alcalá-Fdez J, Sánchez L, García S, del Jesús M J, Ventura S, Garrell J M, Otero J, Romero C, Bacardit J, Rivas V M, Fernández J C and Herrera F 2009 KEEL: a software tool to assess evolutionary algorithms for data mining problems Soft Comput 13 3 307–18
[16] Kamiya Y 2017 A new simple preprocessing method for MUSIC suitable for non-contact vital sensing using Doppler sensors Proc. of the Intelligent Interactive Multimedia Systems and Services (IIMSS 2017) pp 514–24
[17] Kamiya Y 2017 A simple parameter estimation method for periodic signals applicable to vital sensing using Doppler sensors, SICE Journal of Control, Measurement, and System Integration 10 5 378–84