Comparison of Machine Learning Algorithms for Handwritten Digit Recognition

Shixiao Wu[1,2(✉)], Wanyun Wei[3], and Libing Zhang[2]

[1] School of Electronic Information, Wuhan University, Wuhan 430072, China
343564602@qq.com
[2] Department of Information Engineering, Wuhan Business University, Wuhan 430056, China
[3] Department of Electromechanical Engineering, Lanzhou Resource & Environment Voc-Tech College, Lanzhou 730000, China

Abstract. This paper adopts 10 machine learning algorithms to present the classification results of handwritten digit recognition on Minist dataset. These algorithms include k-nearest neighbors, support vector machine (SVM), decision trees (DT), random forest (RF), naive bayes, multilayer perception (MLP), logistic regression with neural network, artificial neural network (ANN), back-propagation (BP), convolutional neural network (CNN) and so on. We execute the experiments through matlab2015b and anaconda (python 3.6), and the result (accuracy and run-time) shows that SVM and RF achieve better performance. They has the accuracy of 98.08% and 97% separately, less running-time is taken compared with other methods. All the experiment are executed in CPU environment, without GPU. We also execute CNN algorithm for handwritten digit recognition in GPU (Nvidia GeForce GTX 1060), finally find that this algorithm achieves the best performance and the best classification result, the accuracy is up to 99%.

Keywords: Machine learning · Handwritten digit recognition Comparison

1 Introduction

Handwritten Digit recognition is well known in OCR and pattern recognition [1]. They can deal with problems like Zipcode recognition, bank check processing, form data entry, etc. For the Zipcode recognition, Wang and Srihari believe that acquisition, binarization, location, and preliminary segmentation should be performed [2]. Metric that judges a recognition system always include recognition accuracy and elapsed time, memory and so on. Feature extraction and classifier selection largely effect the performance of the recognition [3, 4].

Compared with on-line handwritten digit recognition [5], off-line recognition still plays the leader [6]. In this paper, we only focus on the classifier performance and off-line handwritten digit recognition. There are many existing techniques for handwritten digit recognition. LeCun et al., apply large BP networks to solve real image-recognition problems [7], Matan et al., adopt space displacement neural network (SDNN) to recognize

© Springer Nature Singapore Pte Ltd. 2018
K. Li et al. (Eds.): ISICA 2017, CCIS 874, pp. 532–542, 2018.
https://doi.org/10.1007/978-981-13-1651-7_47

handwritten multi-digit string [8], Hinton et al., use linear auto-encoders to recognize handwritten digit from grey-level images. UNIPEN database is also a famous testbed in isolated handwritten character recognition [9]. Statistical methods such as fisher discriminant analysis and PCA [10, 11], machine learning methods like MLP, RF, ANN, CNN, BP, NB, SVM, etc., are well-known solutions [3]. Lotfi and Benyettou apply probabilistic neural networks for handwritten digits [12]. Le Cun et al., make a comparison about various classifiers, such as Baseline Linear Classifier, Baseline Nearest Neighbor or Classifier, Pairwise Linear Classifier, Principal Component Analysis and Polynomial Classifier, Radial Basis Function Network, Multilayer Neural Network, LeNet network, Tangent Distance Classifier (TDC), Optimal Margin Classifier (OMC) and so on. Cheng-Lin Liu et al., has estimated the performance of different classifier, such as MLP, RBF classifier, PC, and LVQ classifier, DLQDF, SVM, etc. In this paper, we use the accuracy and elapsed time as metric to compare the performance of 10 machine learning solutions, also we will mention main parameter settings.

The next arrangement are as follows: we will make a short description for MNIST database in Sect. 2, we introduce 10 mentioned machine learning solutions in Sect. 3, we give the experiment figures and tables result in Sect. 4, we make a conclusion in Sects. 5 and 6 we give the project supports directions.

2 MNIST

Generally, the MINIST Database are composed of 60,000 training images and 10,000 testing images. For a larger set available from NIST, MNIST is a subset [13]. NIST's Special Database 3 (SD3) and Special Database 1 (SD1) help to construct the MINST. Among the complete set of samples, they believe that the result should be independent of the choice of training set and test, data from different sources (SD3 and SD1) are collected to mix the NIST database. For the 60,000 training images and 10,000 testing images, SD3 and SD1 take the half. This paper adopt all the training images and testing images to test the performance of the classifier. As the experiments we have made, different number of images, different configuration of the computer, even the running status of the CPU (like run a lot of procedures) will largely effect the performance.

3 Machine Learning Methods

In this section, we will talk about 10 machine learning methods, which include CNN, RF, SVM (Poly kernel, rbf kernel, linear kernel), KNN (5NN, 9NN), ANN, BP, MLP, NB, Logistic Regression (LR) with NN, DT and so on. We will introduce them shortly as below:

3.1 CNN

Traditionally, CNN consists of input layer, convolutional layer, pooling/downsampling layer and fully-connected layer [14]. The convolutional layer detects local conjunctions of features from the previous layer, the pooling layer merge semantically similar

features into one. The main difference from other deep architectures is that CNN is designed to use minimal amounts of pre-processing [15]. The massive amount of convolution operations and large memory requirement are two bottlenecks common in CNN-based inferencing.

Among all the methods in CPU environments, CNN provides the highest accuracy, up to 99%, however, the elapsed time of CNN is about 7389 s, they take too much time to train. This test is executed in matlab deep learning toolbox, 60000 pictures are used to train and 10000 pictures are used to test, each of them has the size 28 * 28. For CNN layers, input layer, 2 convolution layers, 2 sub sampling layers are included, alpha is 1, batch-size is 50, numepochs is 1.

We also execute CNN algorithms in GPU environment, therein we install nvidia driver 384.90, cuda 9.0, cudnn v5, caffe. With the help of Nvidia GeForce GTX 1060 3G, we complete the test in 1 min and finally get the same classification result. Deep learning method CNN performs better.

3.2 RF

A random forest is composed of a collection of tree-structured classifiers, therein each of them is independent identically distributed random vector, and they casts a unit vote separately for the most popular class [16]. For RandomForestClassifier, parameters is set by n_estimators = 150, criterion = “gini”, max_depth = 32, max_features = “auto” (Table 1 and Fig. 1).

Table 1. SVM for MNIST (kernel = ‘poly’, degree = 2)

	Precision	Recall	F1-score	Support
0	0.9730	0.9918	0.9823	980
1	0.9920	0.9885	0.9903	1135
2	0.9561	0.9700	0.9630	1032
3	0.9598	0.9693	0.9645	1010
4	0.9774	0.9705	0.9739	982
5	0.9773	0.9641	0.9707	892
6	0.9740	0.9781	0.9760	958
7	0.9735	0.9660	0.9697	1028
8	0.9627	0.9528	0.9577	974
9	0.9601	0.9534	0.9567	1009
Avg/total	0.9707	0.9707	0.9707	10000

3.3 SVM

By projecting data into feature space and then searching the optimal separate hyper-plane, SVM can transform the non-linear problems into linear problems [3]. Basically, SVM solve binary classifier problems, but LIBSVM is developed to cover the multi-class problems [17]. Its kernel function includes poly kernel, rbf kernel, linear

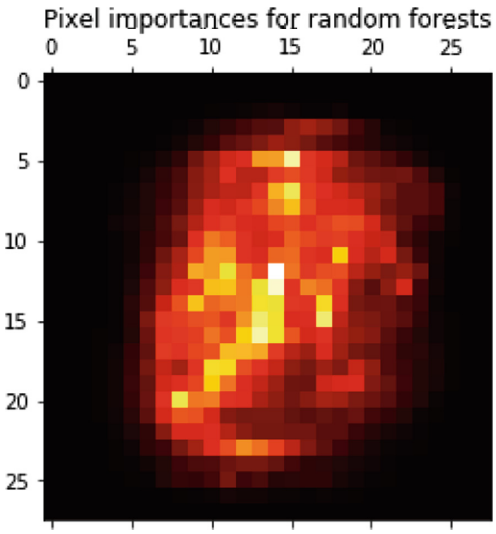

Fig. 1. Pixel importances for RF (dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.0741169 to fit)

kernel and sigmoid kernel. We compared the first three kernel function, finally find that poly kernel has the best performance and less time. For parameter setting, kernel = 'poly', degree = 2. Also we tried 3 kernel, two other kernel include linear and rbf, training images is 60000, and training labels is 60000. Running time: is 510.06473140107244 s. This code is provided by efe (Table 2).

Table 2. SVM for MNIST (kernel = 'poly', degree = 2)

	Precision	Recall	F1-score	Support
0	0.9789	0.9929	0.9858	980
1	0.9886	0.9938	0.9912	1135
2	0.9777	0.9767	0.9772	1032
3	0.9782	0.9772	0.9777	1010
4	0.9827	0.9837	0.9832	982
5	0.9798	0.9776	0.9787	892
6	0.9874	0.9812	0.9843	958
7	0.9795	0.9747	0.9771	1028
8	0.9774	0.9764	0.9769	974
9	0.9751	0.9703	0.9727	1009
Avg/total	0.9806	0.9806	0.9806	10000

3.4 KNN

KNN is an unsupervised machine learning methods. It is also the most simple method for machine learning, they determine the final classification through the affiliation of the great majority among the nearest neighbor. They have the fatal disadvantage, large quantity of computation takes more time. The value of K influence the performance of the classifier (Fig. 2).

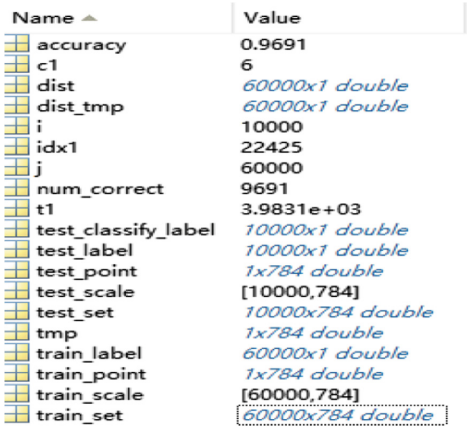

Fig. 2. KNN (k = 1), result and parameter setting (DT)

3.5 ANN

Bajpai et al., believes that ANN is one type of network which treat the node as "artificial neurons" [18]. Inspired in the natural neurons, this artificial neuron is a computational model, which highly abstract the complexity of real neurons. The neuron is activated when natural neurons receive strong signals, and then inputs and outputs of the natural neurons are computed through some mathematical function. This kind of network always include input layer, hidden layer, and output layer. For handwritten digit recognition, the input and output layer has 784 and 10 nodes separately, the number of hidden layer units is 300. For parameter settings, alpha = 0.1; (learning rate), beta = 0.01 (scaling factor for sigmoid function).

3.6 BP

Based on Deepest-Descent technique, Buscema et al., believe that BP is one kind of ANN [19]. If the hidden units has an appropriate number, they can simulate complex computation and minimize the error of nonlinear functions under this situation. Although they have flexible structure, the learning speed is slow and local minimum is easy to come out (Fig. 3).

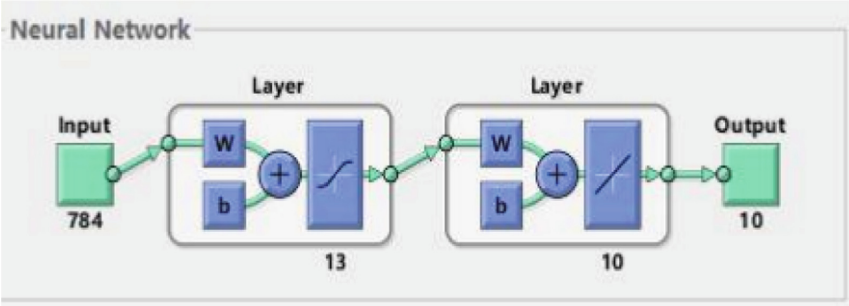

Fig. 3. BP network settings (DT)

3.7 MLP

For performing a wide variety of estimation tasks, MLP is a non-parametric technique [20]. The most widely used algorithm for training MLP is Error back propagation (EBP). MLP is one kind of ANN. For parameter setting, learning_rate is 0.5, weight_decay is 0, momentum is 0, minibatch sample size is 1, the number of iterations between displaying info is 100, the maximum number of iterations is 100000, the number of iterations between testing is 10. The final result is when testing iterations reaches 100000, mean loss is 0.06542, mean accuracy is 96.22%, the running time is 8680.518088.

3.8 NB

Given the value of the class variable, all attributes are independent in NB network [21]. The conditional independence assumption in the real word is feasible, so it employs competitive performance. Train set Accuracy: 83.545%, test set Accuracy: 84.26% (Fig. 4).

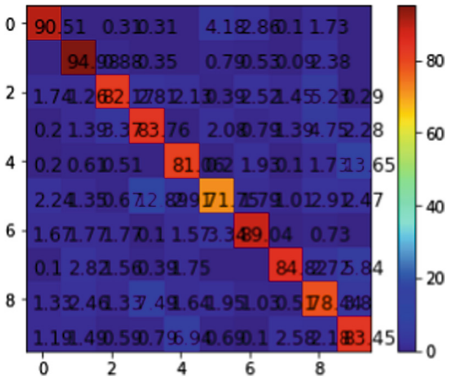

Fig. 4. Naive_Bayes_feature_confusion

3.9 LR

LR is the most simple binary-classification algorithm in the word. If the regression function is defined, you can classify the result into two kinds. For handwritten digit recognition, LR with NN (neural networks) can be applied. The parameters is set as follows: HidUnits = 400, learnRate = 0.1, batchSz = 100, miniBSz = 100, iterations is 20 (Fig. 5).

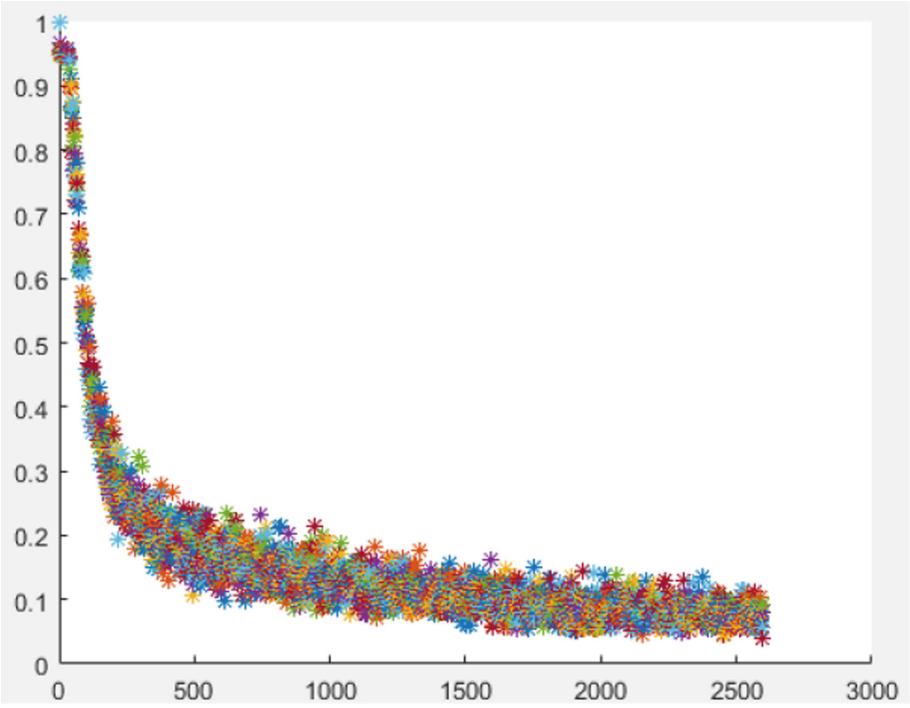

Fig. 5. Error is decreasing (LR with NN)

3.10 DT

Decision trees is composed of intermediate nodes and leaf nodes [22]. Conditions are adopted to label the outgoing edges from intermediate nodes, decisions or actions are used to label the leaf node. A leaf is reached by starting at the root then navigating down on true conditions. Running time in python (spider) is 249.32678459503586 s, Accuracy is 0.87 (±0.00). For parameter settings, criterion = "gini", max_depth = 32, max_features = 784. Training images is 60000 (Table 3 and Fig. 6).

Table 3. The running-time and accuracy for ten machine learning.

	Precision	Recall	F1-score	Support
0	0.9106	0.9357	0.9230	980
1	0.9546	0.9630	0.9588	1135
2	0.8708	0.8488	0.8957	1032
3	0.8259	0.8525	0.8408	1010
4	0.8749	0.8829	0.8789	982
5	0.8401	0.8307	0.8354	892
6	0.8912	0.8894	0.8903	958
7	0.9098	0.9027	0.9062	1028
8	0.8286	0.8039	0.8161	974
9	0.8564	0.8573	0.8569	1009
Avg/total	0.8781	0.8783	0.8781	10000

4 Experiment and Result

We introduce the algorithms in Sect. 3 shortly, this part we will give the final experiment result. We execute the code in python 3.5 and matlab 2015b. We execute CNN, the deep learning toolbox is needed. ANN, MLP, KNN, BP, CNN, LR with NN, SVM are executed in matlab 2015b, DT, RF, NB is executed in python 3.5. In Table 4, we will show the result for ten machine learning algorithms for handwritten digit recognition.

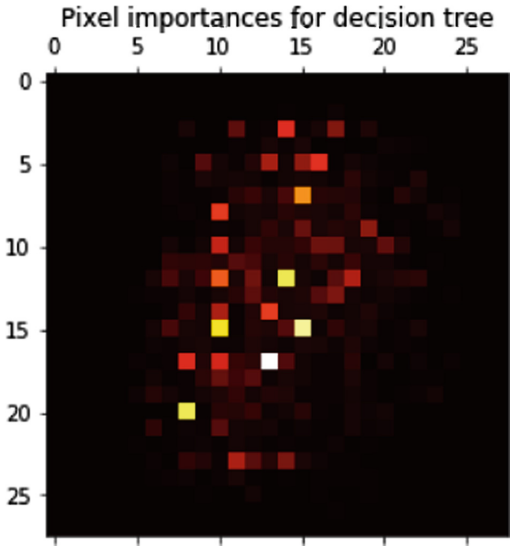

Fig. 6. Pixel importances for DT (dot: graph is too large for cairo-renderer bitmaps. Scaling by 0.0741169 to fit)

Table 4. The running-time and accuracy for ten machine learning.

Algorithm	Accuracy	Running time (seconds)
5NN	96.88%	3639.7
9NN	96.59%	3649.2
ANN	97.87%	7997.149
SVM (poly kernel)	98.08%	324.371
SVM (rbf kernel)	94.46%	839.468
SVM (linear kernel)	93.98%	482.513
MLP	96.44%	8680.518
CNN	98.85%	7389.859
LR with NN	96.81%	1187.660
NB	84.26%	1620
BP	92.04%	312.425
DT	87%	240.642
RF	97%	332.275

We find that the final running-time is highly related with the configuration and manipulation of the computer. If the same algorithm is executed in different software (python/matlab), the running time seems different. But through the accuracy and the running time, you can get the result that RF and SVM has the less time and higher accuracy (Figs. 7 and 8).

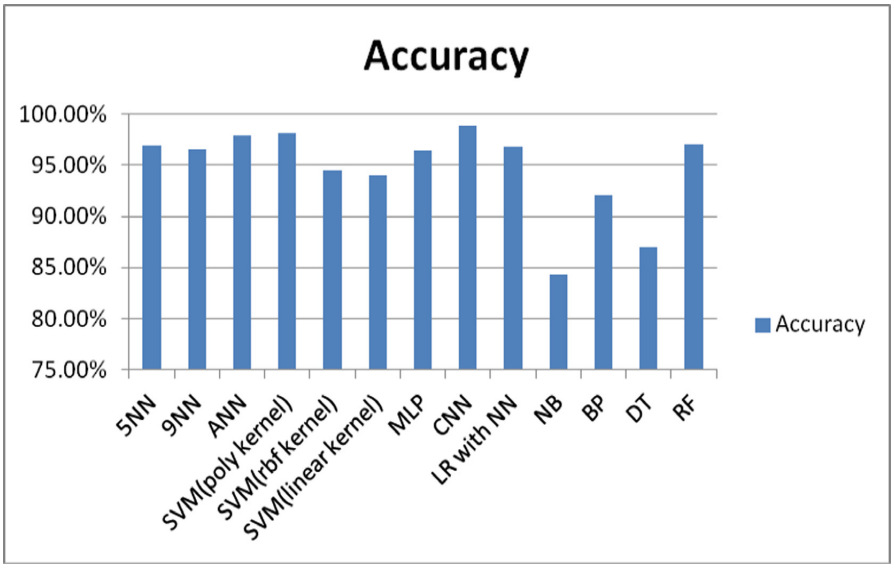

Fig. 7. The accuracy of 10 machine learning algorithms (CPU)

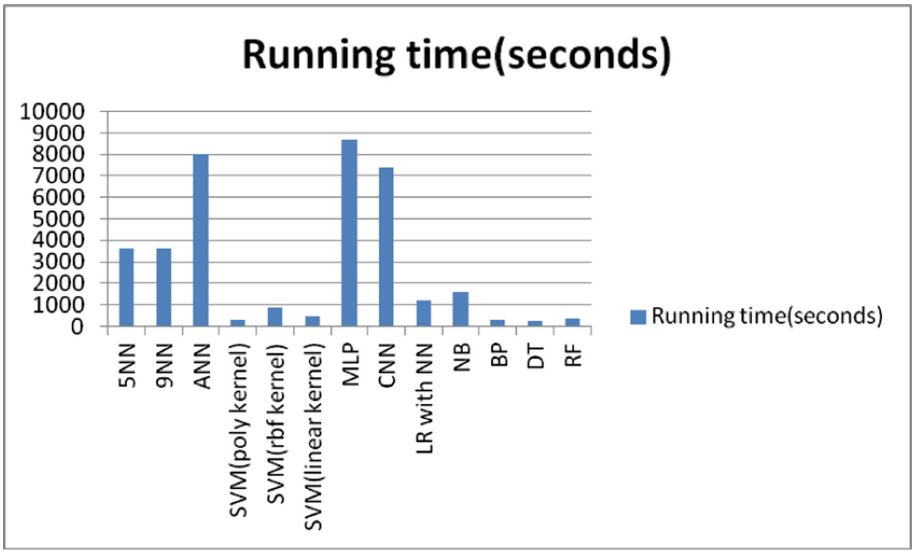

Fig. 8. Running time of 10 machine learning algorithms (CPU)

5 Conclusions

In this paper, we compare 10 machine learning algorithms for handwritten digit recognition in CPU environment. After the experiment, we find that RF and SVM take the less time and higher accuracy. The running time of the algorithms is largely effected by the status and the configuration of the computer. We also execute the experiment in GPU mode, we have Nvidia 384.90 driver, cuda 9.0 and cudnn v5, caffe, then get the same result in CPU but better speed, no more than 1 min.

Acknowledgments. This work was financially supported by Wuhan Teaching and Learning Research Programme (2017113).

References

1. Liu, C.-L., Nakashima, K., Sako, H., Fujisawa, H.: Pattern Recogn. **36**, 2271 (2003)
2. Wang, C.-H., Srihari, S.N.: Int. J. Comput. Vision **2**, 125 (1988)
3. Niu, X.-X., Suen, C.Y.: Pattern Recogn. **45**, 1318 (2012)
4. Lauer, F., Suen, C.Y., Bloch, G.: Pattern Recogn. **40**, 1816 (2007)
5. Kherallah, M., Haddad, L., Alimi, A.M., Mitiche, A.: Pattern Recogn. Lett. **29**, 580 (2008)
6. Arica, N., Yarman-Vural, F.T.: IEEE Trans. Syst. Man Cybern. Part C (Appl. Rev.) **31**, 216 (2001)
7. LeCun, Y., et al.: Advances in Neural Information Processing Systems, pp. 396–404 (1990)
8. Matan, O., Burges, C.J., LeCun, Y., Denker, J.S.: Advances in Neural Information Processing Systems, pp. 488–495 (1992)

9. Guyon, I., Schomaker, L., Plamondon, R., Liberman, M., Janet, S.: Proceedings of the 12th IAPR International Conference on Pattern recognition. Conference B: Computer Vision and Image Processing, vol. 2, pp. 29–33. IEEE (1994)
10. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification. Wiley, New York (2012)
11. Yu, N., Jiao, P.: 2012 IEEE Fifth International Conference on Advanced Computational Intelligence (ICACI), pp. 689–693. IEEE (2012)
12. Lotfi, A., Benyettou, A.: J. Artif. lntell. **4**, 288 (2011)
13. LeCun, Y.: MNIST OCR data (2013)
14. Bag, S.: Deep learning localization for self-driving cars, Ph.D. thesis. Rochester Institute of Technology (2017)
15. Qiu, X., Zhang, L., Ren, Y., Suganthan, P.N., Amaratunga, G.: 2014 IEEE Symposium on Computational Intelligence in Ensemble Learning (CIEL), pp. 1–6. IEEE (2014)
16. Breiman, L.: Mach. Learn. **45**, 5 (2001)
17. Chang, C.-C., Lin, C.-J.: ACM Trans. Intell. Syst. Technol. (TIST) **2**, 27 (2011)
18. Bajpai, S., Jain, K., Jain, N.: Int. J. Soft Comput. Eng. (I-JSCE) **1** (2011)
19. Buscema, M.: Subst. Use Misuse **33**, 233 (1998)
20. Verma, B.: IEEE Trans. Neural Netw. **8**, 1314 (1997)
21. Zhang, H.: AA **1**, 3 (2004)
22. Dobra, A.: Decision trees. In: Liu, L., Özsu, M. (eds.) Encyclopedia of Database Systems. Springer, New York (2016). https://doi.org/10.1007/978-1-4899-7993-3_553-2