Assignment II: Evaluation on Random Forest and Extreme Leaning Machine Algorithm

Wang Maosen

School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore
Email: wang1496@e.ntu.edu.sg

Abstract: This programming assignment presents the evaluation and comparison of algorithms that are learnt in EE6227, Genetic Algorithm and Machine Learning. Random forest (RF) and Extreme Learning Machine (ELM) algorithm are used in this assignment. In this assignment, these algorithms are described combining with matlab code in the first part. Then, the important parameters that should be tune and the tuning method are introduced in the next part. In the experiment parts, ten datasets are chosen to evaluate. The parameter tuning process is executed for each dataset, and after tuning parameter, the final results can be exported. Signrank is used to do statistical testing. Finally, the conclusion can be drawn combing with output value and statistical results.

Keywords: ELM; Random Forest; Parameter Tuning; Signrank

---

1. Random Forest Description

For a machine learning model, we not only require it to be able to fully learn the training data set, but also require it to have good performance on the test data set, that is, to have a low generalization error. In machine learning models, we can often see that they fit very well on the training set. Once tested outside the sample, the effect is sometimes poor. This is overfitting. One of the most significant shortcomings of decision trees is that it is easy to fall into over-fitting, because we have no restrictions on its growth, so that it fully learns the characteristics of each data, including noise. In order to solve over-fitting, pruning is often used, but it also increases the complexity of the model. In order to solve these problems, Breiman proposed a random forest model [1].

Random forest is a supervised learning algorithm for both classification and regression. However, it is mainly used for classification problem. In the random forest generating process, data samples and data features are randomly selected to generate trees. One forest contains many trees, which can improve robust. Then the prediction results can be gained from each tree, and the optimal solution can be selected by voting. It is an ensemble method than a single decision tree because it reduces overfitting by averaging the results, thereby improving prediction performance [2].

The advantage of random forest is that it can prevent overfit. There are three reasons. Frist, random forest will choose mtry (set in maltab code) number features randomly. Second, in bagging process, random forest chooses samples randomly. Third, the fission direction of the decision tree is random. The steps of the random forest algorithm are summarized as follows:

- Through random bagging with replacement, k training sets are generated from the original data set, each training set contains 65% of distinct samples and 35% duplicates., and the ones that are not selected become out-of-bag data sets.
- Use the training set generated in step one to construct a decision tree. When constructing each tree, n features are randomly selected from all the features. Among these n features, the optimal feature is selected to generate child nodes according to the criterion of the smallest GINI coefficient. Each tree is fully grown and no pruning operation is taken.

➢ During the test, each tree has a classification. According to the simple majority voting principle, the random forest selects the classification with the most votes as the output.

```
% Generate bag indices
nTree = options.nTree;
B_indx = cell(nTree,1);

for i = 1:nTree
    B_indx{i} = randsample(size(trainX,1),size(trainX,1),true);
end
```

Figure 1. Random Bagging

```
for i=1:NO_temp_label
    NO_temp=length(find(temp==temp_label1(i)));
    c1=find(Labels_temp==temp_label1(i));
    diff_labels_l(c1)= diff_labels_l(c1)+NO_temp;
    diff_labels_r(c1)=diff_labels_r(c1)-NO_temp;
end
gr=0;
gl=0;

for nl=1:num_labels
    gl=gl+diff_labels_l(nl)*diff_labels_l(nl);
    gr=gr+diff_labels_r(nl)*diff_labels_r(nl);
end
j=j+length(index_temp);
   if j<=M-minleaf
  gl=1-gl/((j-1)*(j-1));
  gr=1-gr/((M-j+1)*(M-j+1));
   post_gini=(j-1)*gl/M+(M-j+1)*gr/M;
```

Figure 2. Calculating Geni impurity

When random forest generates node, the objective function is called Gini impurity that should be maximized shown in formula (1). This process will be repeated for many times, which is related to the number of trees and features. When the features and trees are large, it will cost much time to generate the forest. The matalb code in calculating Gini impurty is presented in Figure 2.

$$
\begin{aligned}
OF_{\max} &= Gini_{beforesplit} - Gini_{aftersplit} \\
Gini_{beforesplit} &= 1 - \sum_{i=1}^{c} \left(\frac{n_{wi}}{n}\right)^2 \\
Gini_{aftersplit} &= \frac{n^l}{n}\left[1 - \sum_{i=1}^{c} \left(\frac{n_{wi}^{\ l}}{n^l}\right)^2\right] + \frac{n^r}{n}\left[1 - \sum_{i=1}^{c} \left(\frac{n_{wi}^{\ r}}{n^r}\right)^2\right]
\end{aligned}
\tag{1}
$$

2. ELM Description

Among most of the machine learning algorithms, neural networks and support vector machines (SVM) have always played a major role. However, as we all know, both neural networks and SVM face some challenging problems. For example: (1) slow learning speed, (2) need human intervention, (3) poor computational scalability. With the emergence of Extreme Learning Machine (ELM) proposed by professor Huang guangbin in our school [3], some challenges have been solved, and it attracted more and more attention from researchers. ELM is a single hidden layer feedforward network (SLFN). The essence of ELM is that there is no need to adjust the hidden layer of SLFN. Compared with those traditional computation, ELM provides better generalization performance with faster learning speed and minimal manual intervention [4]. From my point of view, ELM imitates neurons in human brain. As shown in Figure 3, there are tens of thousands of neurons in human brain, and the connections between neurons are particular complex and chaotic. So, ELM generates many neurons in the hidden layer with random weights. Then, through mathematical calculation, the neurons needed by the model are selected. As shown in Figure 4, the input weights of ELM are random and fixed, and iterative solution is not required. Only the weights from the hidden layer to the output layer are required to be solved, as shown in Figure 5. So compared with BP algorithm, the training speed of ELM is greatly improved.

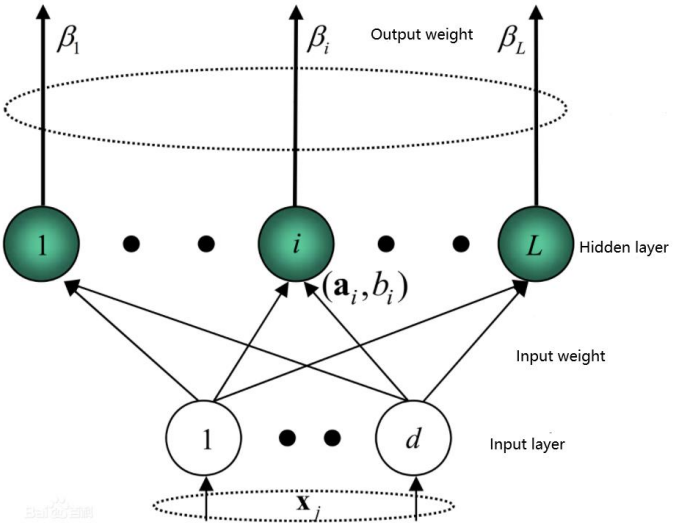


Figure 3. ELM network structure

```
InputWeight=rand(NumberofHiddenNeurons,NumberofInputNeurons)*2-1;
BiasofHiddenNeurons=rand(NumberofHiddenNeurons,1);
tempH=InputWeight*P;
clear P;
ind=ones(1,NumberofTrainingData);
BiasMatrix=BiasofHiddenNeurons(:,ind);
tempH=tempH+BiasMatrix;
```

Figure 4. Random generate input weights

```
%%%%%%%%%%% Calculate hidden neuron output matrix H
switch lower(ActivationFunction)
    case {'sig','sigmoid'}
        H = 1 ./ (1 + exp(-tempH));
    case {'sin','sine'}
        H = sin(tempH);
    case {'hardlim'}
        H = double(hardlim(tempH));
    case {'tribas'}
        H = tribas(tempH);
    case {'radbas'}
        H = radbas(tempH);
end
clear tempH;
%%%%%%%%%%% Calculate output weights OutputWeight (beta_i)
OutputWeight=pinv(H') * T';
```

Figure 5. Calculate output weights

The training process of ELM is as follows:

$$\sum_{i=1}^{N}\beta_i g_i(x_j)=\sum_{i=1}^{N}\beta_i g_i(w_i*x_j+b_i)=o_j,(j=1,...,N) \tag{2}$$

In formula (2), w_i represents the weight between input layer and hidden layer. bi represents bias. β_i represents the weights between hidden layer and output layer. N represents training samples. O_j represents the output of classification. g_i represents activation function. In order to make the training data approximate to the true value, the formula (3) needs to be approached to zero, and we can gain formula (4).

$$\sum_{i=1}^{N}\|o_j-t_j\|=0 \tag{3}$$

$$\sum_{i=1}^{N}\beta_i g_i(w_i*x_j+b_i)=t_j,(t=1,...,N) \tag{4}$$

Formula (4) can be represented as: $H\beta=T$.

Where, $H=\begin{bmatrix} g(w_1*x_1+b_1),...,g(w_N*x_1+b_N) \\ ... \\ ... \\ ... \\ g(w_1*x_N+b_1),...,g(w_N*x_N+b_N) \end{bmatrix}$, $\beta=\begin{bmatrix}\beta_1^T \\ ... \\ \beta_N^T\end{bmatrix}_{N*m}$, $T=\begin{bmatrix}t_1^T \\ ... \\ t_N^T\end{bmatrix}_{N*m}$.

Then, the equation is solved by the least square method, seeking to minimize the training error:

$$\|H(w_1,...,W_N,b_1,...,b_N)\beta-T\|=\min_{\beta}\|H(w_1,...,W_N,b_1,...,b_N)\beta-T\| \tag{5}$$

3. Parameter Tuning

In this section, parameter tuning process is introduced in this part. In random forest, there are two parameters that should be tuned. They are the number of trees and the number of features in each tree. As for ELM, there are also two parameters to tune, including activation function and hidden neurons.

3.1. mtry tuning in random forest

In matlab codes, mtry represents number of features in each tree. The tuning method is to tune mtry from 1 to the total features. Then, updating the best mtry with the highest validation accuracy. The matlab code relating to the tuning process is shown below. The most advantage of this method is to increase accuracy, while training time will be increased.

```
for p1 = 1:numel(mtry_range)
    TestModelParameters = ModelParameters;
    TestModelParameters.mtry = mtry_range(p1);

    val_acc = zeros(4,1);   % Initialisation
    for k = 1:4...

    % Average the validation accuracy
    ValAcc = mean(val_acc);

    % Check if current configuration is the best
    if ValAcc > best_acc
        best_acc = ValAcc;
        best_mtry = mtry_range(p1);
    end
end
```

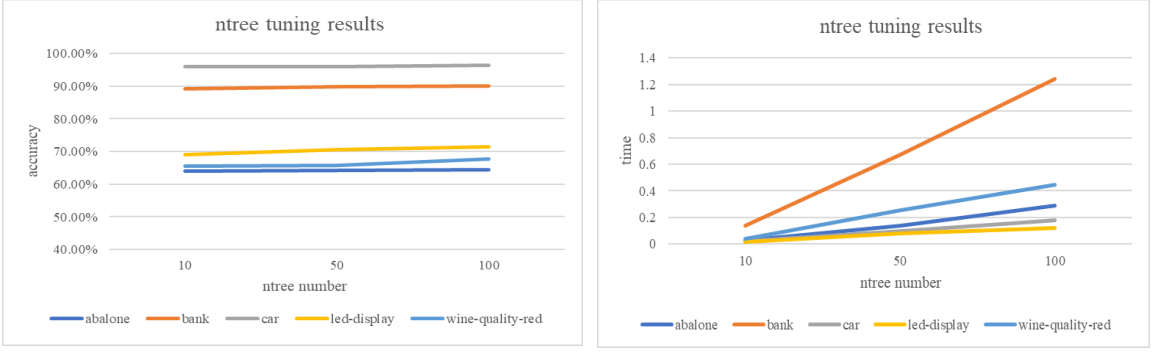
Figure 6. mtry tuning process

3.2. ntree tuning in random forest

In random forest, ntree means the number of trees in the forest. The tuning method is running five datasets with different number of trees. Then, the most suitable number can be chosen by analyzing tuning results. The tuning results are shown below. The experiment results are plotted in Figure 7. It can be seen from the table and the figure that the test accuracy can be slightly boosted with the increase of ntree. However, at the same time, the time cost will be greatly increased. User can tune this parameter according to their needs. If they need higher accuracy, they can increase ntree. If they can accept a relative low accuracy but they want to save training time, they can lower ntree.

Table 1. ntree tuning experiments.

Dataset	ntree = 10		ntree = 50		ntree = 100	
	Test accuracy	Test time	Test accuracy	Test time	Test accuracy	Test time
abalone	64.00%	0.03	64.11%	0.1381	64.47%	0.2863
bank	89.17%	0.1371	89.83%	0.6726	90.06%	1.2397
car	95.95%	0.0206	95.95%	0.0986	96.53%	0.1817
led-display	69.00%	0.015	70.50%	0.0803	71.50%	0.1217
wine-quality-red	65.63%	0.0403	65.69%	0.2537	67.81%	0.4471



(**a**) (**b**)

Figure 8. ntree tuning experiments. (a) accuracy figure, (b) time figure.

3.3. activation function tuning in ELM

In ELM, the activation function is used to add non-linear factors, improve the neural network's ability to express the model, and solve problems that cannot be solved by the linear model. The activation function tuning method is to run dataset with the provided five activation function, including, sigmoidal function, sine function, hardlim function, triangular basis function, and radial basis function. Then, choose the highest validation accuracy.

3.4. neurons tuning in ELM

In ELM, the number of hidden layer's neuron should also be tune. This parameter affects training time and accuracy. The method to tune this parameter is similar to activation function. Two "for" loops are set in my matlab code. When training each dataset, the program will choose the best number of neurons and activation function. The range of neurons is from 10 to 1000, each loop will increase 10 neurons. The ELM tuning codes is presented in Figure 9. Two for loop are used in this process, j represents activation number, and i represents neurons in tuning process.

```
for j=1:5
    if j==1
        activation_function='sig';
    end
    if j==2
        activation_function='sin';
    end
    if j==3
        activation_function='hardlim';
    end
    if j==4
        activation_function='tribas';
    end
    if j==5
        activation_function='radbas';
    end
    for i=1:100
    Neurons = 10*i;
```

Figure 9. ELM tuning process

4. Experiments

In this part, ten datasets are chosen to test the two selected algorithm's performance. The parameter of the dataset is presented in table 2. The experimental results of the two algorithms are presented below. Signrank is used to do statistical testing between the results of ELM and Random Forest. The testing results are presented in table 5, the first value represents h, the second value represents p. p represents the probability that the medians of the two samples X and Y are equal. h=0 means that the difference between the medians of X and Y is not significant, and vice versa (h=1).

Table 2. Dataset characteristics

Dataset	Training Sampls	Training Features	Training Classes
Abalone	3341	8	3
Bank	3616	16	2
Car	1382	6	4
Led-display	800	7	10
Wine-quality-red	1279	11	6
Plant-margin	1280	64	100
Contrac	1178	9	3
Chess-krvkp	2556	36	2
Statlog-german-credit	800	24	2
yeast	1187	8	10

Table 3. Random Forest Experiment Results (ntree=2)

Dataset	**mtry**	**TestAcc**	**TestingTime**	**TrainingTime**	**ValidationAcc**
Abalone	8	57.42%	0.0216	2.4870	56.45%
Bank	13	89.50%	0.0252	1.8813	89.05%
Car	6	94.22%	0.0135	0.1362	95.37%
Led-display	7	70.50%	0.0061	0.1052	69.13%
Wine-quality-red	11	54.06%	0.0083	1.2032	55.67%
Plant-margin	48	45.00%	0.0136	4.4247	36.87%
Contrac	2	49.15%	0.0067	0.0877	49.49%
Chess-krvkp	34	98.28%	0.0110	0.5111	98.51%
Statlog-german-credit	15	70.00%	0.0045	0.3119	71.63%
yeast	6	50.17%	0.0131	0.4891	48.94%

Table 4. Random Forest Experiment Results (ntree=10)

Dataset	**mtry**	**TestAcc**	**TestingTime**	**TrainingTime**	**ValidationAcc**
Abalone	1	64.00%	0.029	2.1114	62.74%
Bank	12	89.17%	0.1228	6.9219	89.77%
Car	4	95.95%	0.0189	0.2546	97.32%
Led-display	3	69.00%	0.0146	0.1433	71.88%
Wine-quality-red	7	65.63%	0.0384	3.1643	64.66%
Plant-margin	31	69.69%	0.0603	12.7525	63.91%
Contrac	4	55.25%	0.0377	0.8816	52.29%
Chess-krvkp	21	98.75%	0.0448	1.4131	99.18%
Statlog-german-credit	20	75.50%	0.0212	1.6546	74.50%
yeast	8	55.89%	0.0429	2.5296	58.30%

Table 5. Random Forest Experiment Results (ntree=100)

Dataset	mtry	TestAcc	TestingTime	TrainingTime	ValidationAcc
Abalone	1	64.47%	0.2937	43.6428	65.37%
Bank	9	90.06%	1.238	53.0228	90.04%
Car	6	96.53%	0.1796	3.2022	97.97%
Led-display	2	71.50%	0.1231	1.0426	72.25%
Wine-quality-red	3	67.81%	0.4139	16.2417	67.71%
Plant-margin	9	85.94%	0.563	43.9061	81.17%
Contrac	1	56.61%	0.0591	1.6229	52.37%
Chess-krvkp	21	98.28%	0.4796	14.2479	99.06%
Statlog-german-credit	17	77.00%	0.2284	15.383	75.50%
yeast	2	60.94%	0.3945	7.7287	62.26%

Table 6. ELM Experiment Results

Dataset	Activation function	Neur -ons	Test Acc	Train time	Train Acc	Validation Acc
Abalone	Sigmoidal	70	64.59%	0	68.66%	69.46%
Bank	Sigmoidal	100	89.16%	0	91.71%	90.27%
Car	sin	250	87.34%	0	100%	97.97%
Led-display	Sigmoidal	40	69.50%	0.125	75.75%	72.00%
Wine-quality-red	sin	330	59.69%	0.5156	78.19%	64.38%
Plant-margin	Sigmoidal	400	77.19	0.5469	98.98%	99.79%
Contrac	Sigmoidal	180	55.93%	0.125	64.01%	52.38%
Chess-krvkp	Sigmoidal	520	95.16%	0.6406	98.51%	99.06%
Statlog-german-credit	Sigmoidal	30	75.51%	0	76.63%	77.00%
yeast	Sigmoidal	70	63.30%	0.1563	63.52%	61.28%

Table 7. Statistical Testing (Signrank)

Dataset	RF (ntree=2)	RF (ntree=10)	RF (ntree=100)	ELM
RF (ntree=2)	0/1	1/0.0137	1/0.0039	0/0.1602
RF (ntree=10)	1/0.0137	0/1	1/0.0059	0/0.9102
RF (ntree=100)	1/0.0039	1/0.0059	0/1	1/0.0371
ELM	0/0.1602	0/0.9102	1/0.0371	0/1

5. Conclusion

In this assignment, two learning algorithms are evaluated. It can be seen from the experiment results that classification accuracy of random forest can be increased with the boosting of the number of trees, especially when the accuracy is not high. This conclusion can be drawn by comparing testing accuracy in table 3, table 4, and table 5. In statistical testing, the signrank results show that the increasing trees from 10 to 100 can have a significant increasing in terms of accuracy. However, the training process becomes very slow with trees increased. This is because the forest needs more time to generate trees. Meanwhile, when the accuracy is relatively high, increasing trees has a small increase in accuracy. As for ELM, the most advantage is that ELM training process is very fast. This is because the weights between input layer and hidden layer are fixed. At the same time, the accuracy is high. In the selected ten dataset, five test results in ELM are higher than Random Forest (ntree=10).

References

1. Breiman, Leo. "Random forests." Machine learning 45.1 (2001): 5-32.

2. Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." R news 2.3 (2002): 18-22.
3. Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: theory and applications." Neurocomputing 70.1-3 (2006): 489-501.
4. Huang, Guang-Bin, et al. "Extreme learning machine for regression and multiclass classification." IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 42.2 (2011): 513-529.