Application of Deep Forest Method on Hand-writing Dataset

Ye Yushi (12207894)
Department of Mathematics
The Hong Kong University of Science and Technology
yyeaf@connect.ust.hk

April 10, 2017

Abstract

In this mini-project, we try to study the *deep forest* method [1] introduced by Professor Zhou Zhihua's team in February, 28th, 2017 and apply it on the hand-writing data supplied by Professor Yao Yuan. The result shows that this method do has the ability of increasing the prediction power layer by layer. Based on the idea of this new learning structure, we developed a modified model called "deep boosting", which can offer competitive prediction performance with deep forest but the training time is reduced significantly.

1 The Structure of Deep Forest Method

Deep forest is a novel decision tree ensemble method introduced by Professor Zhou Zhihua's team in February, 28th, 2017. The motivation is that people want to develop an ensemble learning structure which can go "deep" like neural networks (i.e. the model has multi-layer structure, and the number of layers can be very large). This new structure can fill some gaps of deep neural networks but also give competitive prediction accuracy. Specially, this method contains two major techniques: *cascade forest* and *multi-grained scanning*. We will try to describe these two techniques in a statistical way in the next two subsections. For simplicity, here we focus on classification problems with K classes of labels.

1.1 Cascade Forest

The idea of cascade forest is enlightened by representation learning in deep neural networks. In cascade forest model, each layer's output (i.e. prediction values or label probabilities) will turn to be extra features of next layer's input. In consequence, the model can go very deep if the prediction performance keeps increasing.

Specially, each layer of cascade forest is made up by several forests (the number of forests is a hyper-parameter m), these forest can be commonly used random forests, or some other decision tree ensembles. For example, in [1], the author introduced *complete random tree forest*, which contains n (another hyper-parameters) complete random trees, generated by randomly selecting a feature for split at each node of the tree, and growing tree until each leaf node contains only the same class of instances or no more than s_{min} (another hyper-parameter) instances. After fitting models with the training data, given an instance, each forest will output a class vector, with elements represent the estimated probabilities of different labels (for example, in the hand-writing data, the number of classes should be 10, i.e. 0 to 9). So suppose there are K classes, the whole layer will generate a output vector of $m \times k$ dimensions. Then this vector will be treated as augmented features for next layer.

Submitted to MATH 6380J Mini-project 2 .

In [1], the author also introduce k-folder cross validation to reduce the risk of overfitting. So each instance will be trained $k-1$ times in one forest, resulting $k-1$ class vectors. Then these vectors will be averaged for a final output.

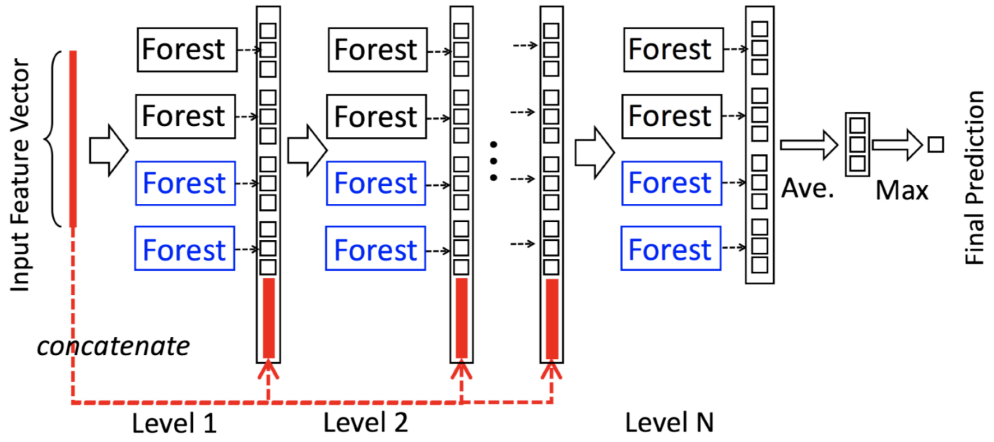

Figure 1: Illustration of Cascade Forest (copied from [1])

1.2 Multi-grained Scanning

The idea of multi-grained scanning is enlightened by convolutional neural networks (CNNs). This procedure is not necessary for all training datasets, but can improve prediction performance largely for sequential or spacial datasets such as images. Specially, for each instance in a sequence dataset, the m raw features will be sliced by a window with size d (a hyper-parameter) to generate $m-d+1$ new vectors with dimension d. Then we will train $m-d+1$ forest models to generate class vectors, these prediction vectors will be used as initial input features for later cascade forest models. This procedure can be illustrated by Figure [2].

2 Deep Boosting Method

Enlightened by the idea of deep forest model, here we developed a cousin model called deep boosting. The idea is for each layer of the cascade structure, instead of modeling several forests, we will build m boosting trees. For each tree, one can tune some regularization parameters such as learning rate γ, subsampling ratio η and maximum training round N. So this model will be more flexible than deep forest (if we set maximum training round $n_{max}=1$, then the boosting tree will be degenerated to a normal decision tree). Also, since nowadays there are some fast learning packages such as `xgboost`, which can model a boosting tree in a very short time period, we can take this advantage to accelerate the training speed of the whole model.

The algorithm of deep boosting is illustrated below:

3 Performance on hand-writing data

In this section, we compare the prediction performance for deep boosting method with and without multi-grained scanning. Specially, each model contains 12 layers. For each layer, we use one forest with 100 boosting trees (parameters include learning rate 0.3, subsampling ratio 0.5, maximum iteration 100, maximum depth 100, etc.). For multi-grained scanning case, we use 10*10 sliding window to generate new features. The results are listed below. They are calculated from the testing set, which contains 10000 instances. The models are trained on the training set, which contains 50000 different samples.

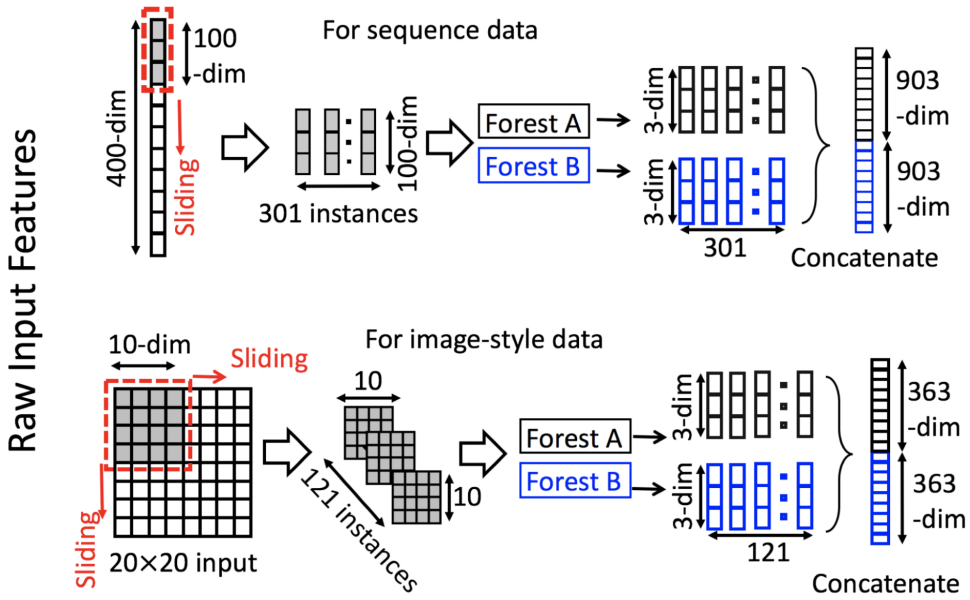

Figure 2: Illustration of Multi-grained Scanning (copied from [1])

Algorithm 1 Deep Boosting

Input: training data $\mathcal{D} = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_N}, y_N)\}$
initial input $\mathbf{x}_0 = \mathbf{x}$
for $n = 1$ to N **do**
  for $m = 1$ to M **do**
    for $k = 1$ to K **do**
      Train a boosting tree on fold $\mathcal{D}_{-k}$, named as $T_{m,k}$, with some tuning parameters such as learning rate γ and subsampling ratio η
    end for
    The prediction of tree T_m is given by $T_m(\mathbf{x}_{n-1}) = \frac{1}{k} T_{m,k}(\mathbf{x}_{n-1})$
  end for
  The prediction of layer n is given by $\mathbf{y}_n = F_n(\mathbf{x}_{n-1}) = \frac{1}{m} T_m(\mathbf{x}_{n-1})$
  The input for next layer will be $\mathbf{x}_n = \{\mathbf{x}_{n-1}, \mathbf{y}_n\}$
end for
Output: A cascade-structure model with N layers, each layer contains m boosting trees $T_1, T_2, \ldots, T_m$. One can use this model to make prediction on new data layer by layer.

Note that the red font shows best performance during iterations. We can see that the multi-grained scanning can give best performance at 8th layer, with mean accuracy 0.976960, a little high than the case that without multi-grained scanning, which give the best result at 12th layer, with mean accuracy 0.971280. This result shows that deep boosting algorithm has the ability of increasing its prediction power layer by layer, and multi-grained scanning can help to increase the prediction performance even further.

4 Comments on Deep Forest Method

From the previous study, we make the following comments on deep forest method.

Table 1: Deep Boosting Prediction Accuracy on the testing set

Number of layers	With multi-grained scanning		Without multi-grained scanning	
	mean	sd	mean	sd
1	0.971060	0.000493	0.953740	0.000611
2	0.974140	0.000654	0.961620	0.000572
3	0.975500	0.000300	0.965800	0.000458
4	0.976020	0.000522	0.968000	0.000442
5	0.976380	0.000239	0.969160	0.000635
6	0.976360	0.000055	0.970060	0.000720
7	0.976840	0.000503	0.970660	0.000270
8	0.976960	0.000404	0.970400	0.000200
9	0.976900	0.000187	0.970860	0.000321
10	0.976860	0.000297	0.970900	0.000418
11	0.976800	0.000308	0.971220	0.000205
12	0.976740	0.000261	0.971280	0.000164

The good:

- Similar to deep neural network, can go very "deep"
- The prediction performance is competitive with other cut-of-edge models
- For each layer, the sub-model can be very flexible. In the last section, we use boosting trees instead of random forest in the original setting, but the performance is even better.
- The tuning parameters are much less than deep neural networks. One can use early stopping regularization to stop the iteration as his wish
- This model can take the advantage of tree-based models, i.e. the model has better explanation ability than deep neural network. One can compare different features by some techniques such as importance sampling.

The bad:

- The training time is much longer than other competitive methods. Since latter layer contains more input variables than previous layer, the training time will also increase.
- As the results shows, the performance improvement actually reduces layer than layer. In the case of deep boosting without multi-grained scanning, the last 6 layers only increase the prediction accuracy from 0.970060 to 0.971280. Considering the computation cost, it seems not so efficient.

References

[1] Zhi-Hua Zhou& Ji Feng (2017) Deep Forest: Towards An Alternative to Deep Neural Networks `arXiv:0706.1234 [math.FA]`