Ensemble Learning

Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China
zhouzh@nju.edu.cn

Synonyms

Committee-based learning; Multiple classifier systems; Classifier combination

Definition

Ensemble learning is a machine learning paradigm where multiple learners are trained to solve the same problem. In contrast to ordinary machine learning approaches which try to learn *one* hypothesis from training data, ensemble methods try to construct a *set* of hypotheses and combine them to use.

Main Body Text

Introduction

An ensemble contains a number of learners which are usually called *base learners*. The **generalization** ability of an ensemble is usually much stronger than that of base learners. Actually, ensemble learning is appealing because that it is able to boost *weak learners* which are slightly better than random guess to *strong learners* which can make very accurate predictions. So, "base learners" are also referred as "weak learners". It is noteworthy, however, that although most theoretical analyses work on weak learners, base learners used in practice are not necessarily weak since using not-so-weak base learners often results in better performance.

Base learners are usually generated from training data by a *base learning algorithm* which can be decision tree, neural network or other kinds of machine learning algorithms. Most ensemble methods use a single base learning algorithm to produce *homogeneous* base learners, but there are also some methods which use multiple learning algorithms to produce *heterogeneous* learners. In the latter case there is no single base learning algorithm and thus, some people prefer calling the learners *individual learners* or *component learners* to "base learners", while the names "individual learners" and "component learners" can also be used for homogeneous base learners.

It is difficult to trace the starting point of the history of ensemble methods since the basic idea of deploying multiple models has been in use for a long time, yet it is clear that the hot wave of research on ensemble learning since the 1990s owes much to two works. The first is an applied research conducted by Hansen and Salamon [1] at the end of 1980s, where they found that predictions made by the combination of a set of classifiers are often more accurate than predictions made by the best single classifier. The second is a theoretical research conducted in 1989, where Schapire [2] proved that *weak learners* can be boosted to *strong learners*, and the proof resulted in Boosting, one of the most influential ensemble methods.

Constructing Ensembles

Typically, an ensemble is constructed in two steps. First, a number of base learners are produced, which can be generated in a *parallel* style or in a *sequential* style where the generation of a base learner has influence on the generation of subsequent

learners. Then, the base learners are combined to use, where among the most popular combination schemes are *majority voting* for classification and *weighted averaging* for regression.

Generally, to get a good ensemble, the base learners should be as more accurate as possible, and as more diverse as possible. This has been formally shown by Krogh and Vedelsby [3], and emphasized by many other people. There are many effective processes for estimating the *accuracy* of learners, such as **cross-validation**, hold-out test, etc. However, there is no rigorous definition on what is intuitively perceived as *diversity*. Although a number of diversity measures have been designed, Kuncheva and Whitaker [4] disclosed that the usefulness of existing diversity measures in constructing ensembles is suspectable. In practice, the diversity of the base learners can be introduced from different channels, such as subsampling the training examples, manipulating the attributes, manipulating the outputs, injecting randomness into learning algorithms, or even using multiple mechanisms simultaneously. The employment of different base learner generation processes and/or different combination schemes leads to different ensemble methods.

There are many effective ensemble methods. The following will briefly introduce three representative methods, *Boosting* [2, 5], *Bagging* [6] and *Stacking* [7]. Here, binary classification is considered for simplicity. That is, let $\mathcal{X}$ and $\mathcal{Y}$ denote the instance space and the set of class labels, respectively, assuming $\mathcal{Y} = \{-1, +1\}$. A training data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$ is given, where $\boldsymbol{x}_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$ $(i = 1, \cdots, m)$.

Boosting is in fact a family of algorithms since there are many variants. Here, the most famous algorithm, AdaBoost [5], is considered as an example. First, it assigns equal weights to all the training examples. Denote the distribution of the weights at the t-th learning round as D_t. From the training data set and D_t the algorithm generates a base learner $h_t : \mathcal{X} \rightarrow \mathcal{Y}$ by calling the base learning algorithm. Then, it uses the training examples to test h_t, and the weights of the incorrectly classified examples will be increased. Thus, an updated weight distribution D_{t+1} is obtained. From the training data set and D_{t+1} AdaBoost generates another base learner by calling the base learning algorithm again. Such a process is repeated for T times, each of which is called a *round*, and the final learner is derived by weighted majority voting of the T base learners, where the weights of the learners are determined during the training process. In practice, the base learning algorithm may be a learning algorithm which can use weighted training examples directly; otherwise the weights can be exploited by sampling the training examples according to the weight distribution D_t. The pseudo-code of AdaBoost is shown in Fig.1.

Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathcal{L}$;
Number of learning rounds T.

Process:
$D_1(i) = 1/m$. % Initialize the weight distribution
for $t = 1, \cdots, T$:
$h_t = \mathcal{L}(\mathcal{D}, D_t)$; % Train a base learner h_t from $\mathcal{D}$ using distribution D_t
$\epsilon_t = \Pr_{i \sim D_i}[h_t(\boldsymbol{x}_i \neq y_i)]$; % Measure the error of h_t
$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$; % Determine the weight of h_t

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$$

$$= \frac{D_t(i)\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$$

% Update the distribution, where Z_t is a normalization
% factor which enables D_{t+1} to be a distribution
end.

Output: $H(\boldsymbol{x}) = \text{sign}\left(f\left(\boldsymbol{x}\right)\right) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

Fig. 1. The AdaBoost algorithm

Bagging [6] trains a number of base learners each from a different *bootstrap sample* by calling a base learning algorithm. A bootstrap sample is obtained by subsampling the training data set with replacement, where the size of a sample is as the same as that of the training data set. Thus, for a bootstrap sample, some training examples may appear but some may not, where the probability that an example appears at least once is about 0.632. After obtaining the base learners, Bagging combines them by majority voting and the most-voted class is predicted. The pseudo-code of Bagging is shown in Fig.2. It is worth mentioning that a variant of Bagging, *Random Forests* [8], has been deemed as one of the most powerful ensemble methods up to date.

Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
Base learning algorithm $\mathcal{L}$;
Number of learning rounds T.

Process:
for $t = 1, \cdots, T$:
$\mathcal{D}_t = Bootstrap(\mathcal{D})$; % Generate a bootstrap sample from $\mathcal{D}$
$h_t = \mathcal{L}(\mathcal{D}_t)$ % Train a base learner h_t from the bootstrap sample
end.

Output: $H(\boldsymbol{x}) = \text{argmax}_{y \in \mathcal{Y}} \sum_{t=1}^{T} 1(y = h_t(\boldsymbol{x}))$ % the value of $1(a)$ is 1 if a is $true$ and 0 otherwise

Fig. 2. The Bagging algorithm

In a typical implementation of Stacking [7], a number of first-level individual learners are generated from the training data set by employing different learning algorithms. Those individual learners are then combined by a second-level learner which is called as *meta-learner*. The pseudo-code of Stacking is shown in Fig.3. It is evident that Stacking has close relation with information fusion methods.

Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
First-level learning algorithms $\mathcal{L}_1, \cdots, \mathcal{L}_T$;
Second-level learning algorithm $\mathcal{L}$.

Process:
for $t = 1, \cdots, T$:
$h_t = \mathcal{L}_t(\mathcal{D})$ % Train a first-level individual learner h_t by applying the first-level
end; % learning algorithm $\mathcal{L}_t$ to the original data set $\mathcal{D}$
$\mathcal{D}' = \emptyset$; % Generate a new data set
for $i = 1, \cdots, m$:
for $t = 1, \cdots, T$:
$z_{it} = h_t(\boldsymbol{x}_i)$ % Use h_t to classify the training example $\boldsymbol{x}_i$
end;
$\mathcal{D}' = \mathcal{D}' \cup \{((z_{i1}, z_{i2}, \cdots, z_{iT}), y_i)\}$
end;
$h' = \mathcal{L}(\mathcal{D}')$. % Train the second-level learner h' by applying the second-level
% learning algorithm $\mathcal{L}$ to the new data set $\mathcal{D}'$

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \cdots, h_T(\boldsymbol{x}))$

Fig. 3. The Stacking algorithm

Generally speaking, there is no ensemble method which outperforms other ensemble methods consistently. Empirical studies on popular ensemble methods can be found in many papers such as [9, 10, 11]. Previously, it was thought that using more base learners will lead to a better performance, yet Zhou et al. [12] proved the "many could be better than all" theorem which indicates that this may not be the fact. It was shown that after generating a set of base learners, selecting some base learners instead of using all of them to compose an ensemble is a better choice. Such ensembles are called *selective ensembles*.

It is worth mentioning that in addition to classification and regression, ensemble methods have also been designed for clustering [13] and other kinds of machine learning tasks.

Why Ensembles Superior to Singles

To understand that why the **generalization** ability of an ensemble is usually much stronger than that of a single learner, Dietterich [14] gave three reasons by viewing the nature of machine learning as searching a hypothesis space for the most accurate hypothesis. The first reason is that, the training data might not provide sufficient information for choosing a single best learner. For example, there may be many learners perform equally well on the training data set. Thus, combining these

learners may be a better choice. The second reason is that, the search processes of the learning algorithms might be imperfect. For example, even if there exists a unique best hypothesis, it might be difficult to achieve since running the algorithms result in sub-optimal hypotheses. Thus, ensembles can compensate for such imperfect search processes. The third reason is that, the hypothesis space being searched might not contain the true target function, while ensembles can give some good approximation. For example, it is well-known that the classification boundaries of decision trees are linear segments parallel to coordinate axes. If the target classification boundary is a smooth diagonal line, using a single decision tree cannot lead to a good result yet a good approximation can be achieved by combining a set of decision trees. Note that those are intuitive instead of rigorous theoretical explanations.

There are many theoretical studies on famous ensemble methods such as Boosting and Bagging, yet it is far from a clear understanding of the underlying mechanism of these methods. For example, empirical observations show that Boosting often does *not* suffer from **overfitting** even after a large number of rounds, and sometimes it is even able to reduce the **generalization** error after the training error has already reached zero. Although many researchers have studied this phenomenon, theoretical explanations are still in arguing.

The **bias-variance decomposition** is often used in studying the performance of ensemble methods [9, 12]. It is known that Bagging can significantly reduce the variance, and therefore it is better to be applied to learners suffered from large variance, e.g., unstable learners such as decision trees or neural networks. Boosting can significantly reduce the bias in addition to reducing the variance, and therefore, on weak learners such as decision stumps, Boosting is usually more effective.

Applications

Ensemble learning has already been used in diverse applications such as optical character recognition, text categorization, face recognition, computer-aided medical diagnosis, gene expression analysis, etc. Actually, ensemble learning can be used wherever machine learning techniques can be used.

Summary

Ensemble learning is a powerful machine learning paradigm which has exhibited apparent advantages in many applications. By using multiple learners, the **generalization** ability of an ensemble can be much better than that of a single learner. A serious deficiency of current ensemble methods is the lack of comprehensibility, i.e., the knowledge learned by ensembles is not understandable to the user. Improving the comprehensibility of ensembles [15] is an important yet largely understudied direction. Another important issue is that currently no diversity measures is satisfying [4] although it is known that diversity plays an important role in ensembles. If those issues can be addressed well, ensemble learning will be able to contribute more to more applications.

Related Entries

Boosting, Classifier design, Machine learning, Multiple classifier systems, Multiple experts.

References

1. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Transactions on Pattern Analysis and Machine Intelligence **12**(10) (1990) 993–1001
2. Schapire, R.E.: The strength of weak learnability. Machine Learning **5**(2) (1990) 197–227
3. Krogh, A., Vedelsby, J.: Neural network ensembles, cross validation, and active learning. In Tesauro, G., Touretzky, D.S., Leen, T.K., eds.: Advances in Neural Information Processing Systems 7. MIT Press, Cambridge, MA (1995) 231–238
4. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Machine Learning **51**(2) (2003) 181–207
5. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to Boosting. Journal of Computer and System Sciences **55**(1) (1997) 119–139
6. Breiman, L.: Bagging predictors. Machine Learning **24**(2) (1996) 123–140
7. Wolpert, D.H.: Stacked generalization. Neural Networks **5**(2) (1992) 241–260
8. Breiman, L.: Random forests. Machine Learning **45**(1) (2001) 5–32

9. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, Boosting, and variants. Machine Learning **36**(1-2) (1999) 105–139
10. Ting, K.M., Witten, I.H.: Issues in stacked generalization. Journal of Artificial Intelligence Research **10** (1999) 271–289
11. Opitz, D., Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research **11** (1999) 169–198
12. Zhou, Z.H., Wu, J., Tang, W.: Ensembling neural networks: Many could be better than all. Artificial Intelligence **137**(1-2) (2002) 239–263
13. Strehl, A., Ghosh, J.: Cluster ensembles - a knowledge reuse framework for combining multiple partitionings. Journal of Machine Learning Research **3** (2002) 583–617
14. Dietterich, T.G.: Machine learning research: Four current directions. AI Magazine **18**(4) (1997) 97–136
15. Zhou, Z.H., Jiang, Y., Chen, S.F.: Extracting symbolic rules from trained neural network ensembles. AI Communications **16**(1) (2003) 3–15

Definitional Entries

Bias-Variance Decomposition

An important tool for analyzing machine learning approaches. Given a learning target and the size of training data set, it breaks the expected error of a learning approach into the sum of three non-negative quantities, i.e., the *intrinsic noise*, the *bias* and the *variance*. The intrinsic noise is a lower bound on the expected error of any learning approach on the target; the bias measures how closely the average estimate of the learning approach is able to approximate the target; the variance measures how much the estimate of the learning approach fluctuates for the different training sets of the same size.

Cross-Validation

A popular approach to estimating how well the result learned from a given training data set is going to generalize on unseen new data. It partitions the training data set into k subsets with equal size, and then uses the union of $k-1$ subsets for training while the remaining subset for performance evaluation. The final estimate is obtained by averaging after every subset has been used for evaluation once. A popular settings of k is 10 and in this case it is called as *10-fold cross-validation*; another popular setting of k is the number of training examples and in this case it is called as LOO (i.e., *Leave-One-Out*) test.

Generalization

The most central concept in machine learning, which characterizes how well the result learned from a given training data set can be applied to unseen new data.

Overfitting

The phenomenon that the learning result performs very good on training data but poorly on unseen new data, which is caused by that the learning approach has fit the training data too much such that some malign particularities that prevents a good generalization has also been captured by the learning result.