Model Combiners for Data Analysis

Rajendra Akerkar

Technomathematics Research Foundation
TMRF Report 08- 2011
Model Combiners for Data Analysis

Improving Accuracy Through Combining Predictions

TMRF-report-08-2011

TMRF White Paper

By Rajendra Akerkar

August 2011
Contents

Summary ......................................................................................................................... 3
Introduction .................................................................................................................. 4
Data Analysis ............................................................................................................... 6
Model Combiners ......................................................................................................... 8
Bagging ......................................................................................................................... 9
Boosting ....................................................................................................................... 11
Outlook ......................................................................................................................... 13
Conclusions .................................................................................................................. 14
Bibliography ................................................................................................................. 15
Summary

The Model Combiners are machine learning methods that leverage the power of multiple models to achieve better prediction accuracy than any of the individual models could on their own. The Model Combiners refers to the procedures employed to train multiple learning machines and combine their outputs, treating them as a “committee” of decision makers. The principle is that the committee decision, with individual predictions combined appropriately, should have better overall accuracy, on average, than any individual committee member. Numerous empirical and theoretical studies have demonstrated that ensemble models very often attain higher accuracy than single models.

The basic goal when designing model combiners is the same as when establishing a committee of people: each member of the committee should be competent as possible, but the members should be complementary to one another. If the members are not complementary, i.e., if they always agree, then the committee is unnecessary—any one member is sufficient. If the members are complementary, then when one or a few members make an error, the probability is high that the remaining members can correct this error. Research in model combiners has largely revolved around designing ensembles consisting of competent yet complementary models.
Introduction

A supervised machine learning task involves constructing a mapping from input data to the appropriate outputs. In a classification learning task, each output is one or more classes to which the input belongs. The goal of classification learning is to develop a model that separates the data into the different classes, with the aim of classifying new examples in the future.

Model combiners can provide a vital boost to industrial challenges -- from investment timing to drug discovery, and fraud detection to recommendation systems -- where predictive accuracy is more crucial than model interpretability.

For example, a credit card company may develop a model that separates people who defaulted on their credit cards from those who did not, based on other known information, such as annual income. The goal would be to predict whether a new credit card applicant is likely to default on his credit card and thereby decide whether to approve or deny this applicant a new card. In a regression learning task, each output is a continuous value to be predicted, e.g., the average balance that a creditcard holder carries over to the next month.

![Figure 1: An ensemble of linear classifiers. Each line is a linear classifier](image)

Many traditional machine learning algorithms generate a single model (e.g., a decision tree or neural network). Model combiners (ensemble) instead generate multiple models. For instance, the ensemble passes it to each of its multiple base models, obtains their predictions, and then combines them in suitable way (e.g., averaging or voting). However, it is important to have base models that are competent but also complementary. The figure 1 illustrates a classification problem in which the goal is to separate the points marked with plus signs from points marked with minus signs. None of the three individual linear classifiers (marked A, B, and C) is able to separate the two classes of points. But, a majority
vote over all three linear classifiers yields the piecewise-linear classifier given as a bold line. This classifier is able to separate the two classes perfectly. For instance, the plusses at the top of the figure are correctly classified by A and B, but are misclassified by C. The majority vote over these correctly classifies these points as plusses. This happens because A and B are very different from C. If our model combiners instead consisted of three copies of C, then all three classifiers would misclassify the plusses at the top of the figure, and so would a majority vote over these classifiers.

By and large the model combiners are used due to the following reasons:

- When the data volume is too large to be handled by a single classifier, an ensemble system can let each classifier process a partition of the data and then combine their results.
- It is also beneficial to construct an ensemble system when the data available is not adequate. A machine learning algorithm usually requires sufficient and representative data to generate an effective model. If the data size is small, there is still a way to satisfy this condition by constructing an ensemble of classifiers trained on overlapping random subsets drawn from resampling the original data.
- When the decision boundary that separates different classes is too complex for a single classifier to cover, this decision boundary can be approximated by an appropriate combination of different classifiers. For instance, although a linear classifier would not be capable of learning a complex non-linear boundary in a two-dimensional and two-class problem, an ensemble of such linear classifiers can solve this problem by learning smaller and easier-to-learn partitions of the data space.

In conclusion, if there is a request for data fusion, model combiners is useful. Data fusion happens when we have data from various sources having features with different number and types that cannot be processed wholly by a single classifier. In this case, individual classifiers are applied on the data from the different sources and their outputs can be combined to make an ultimate decision.
Data Analysis

In data analysis and mechanized decision making applications, there are three fundamental reasons to construct model combiners: statistical, computational and representational. These reasons are derived by viewing machine learning as a search through a hypothesis space for the most accurate hypothesis.

- The statistical problem happens when the data available is too small. When the training data is scarce, the effective space of hypotheses searched by the learning algorithm will be smaller than when the training data is plentiful.
- The computational reason pertains to learning algorithms which get stuck in local optima when they perform a local search. For instance, neural network and decision tree algorithms both have this problem. In this case, even if the statistical problem is absent, it is still difficult for these algorithms to find the best hypothesis. Constructing an ensemble of individual classifiers generated from different starting points may produce a hypothesis which is closer to the true unknown function.
- The representational problem appears when there is no hypothesis in the hypotheses space to represent the true function. That is to say, the true function is outside of the hypotheses space.

Model combiners can be used to evaluate the relationships between explanatory variables and the response in conventional statistical models. Predictors or basis functions overlooked in a conventional model, may surface with an ensemble approach. Conversely, predictors thought be important in a conventional model, may prove to be worthless in output from an ensemble analysis. It does not necessarily follow that the ensemble results are superior. But the process of trying to understand why the two results differ will likely be instructive.

One could use ensemble methods to implement the covariance adjustments inherent in multiple regression and related procedures. One would "residualize" the response and the predictors of interest with ensemble methods. The desired regression coefficients would then be estimated from the sets of residuals.

Tools which support Model Combiners include:
**Open Source tools**
- Orange$^1$
- R$^2$
- WEKA$^3$
- RapidMiner$^4$ (Random Forest)

**Commercial tools**
- BigML$^5$ (Decision tree ensembles)
- wise.io$^6$ (very fast random forest)
- Salford Systems$^7$
- Statistica$^8$

---

$^1$ http://orange.biolab.si/
$^2$ http://www.r-project.org/
$^3$ http://www.cs.waikato.ac.nz/ml/weka/
$^4$ http://rapid-i.com/
$^5$ https://bigml.com/
$^6$ http://www.wise.io/
$^7$ http://www.salford-systems.com/
$^8$ http://www.statsoft.com/
Model Combiners

The term *model combiners* or *ensemble methods* are basically retained for bundled fits produced by a stochastic algorithm, the output of which is some combination of a large number of passes through the data. Such methods are loosely related to iterative procedures, on one hand, and to bootstrap procedures, on the other. This approach has been called by various commentators the most significant development in machine learning over the last decade.

Building predictive models has traditionally involved the training of many models from the entire training data set, and via various validation techniques, picking the best performing candidate. In reality however a single model may well be unstable and does not capture all the useful patterns that might be found in the data. Model combiners exploit some of the weaknesses of a single model approach, and model instability can be used to good effect. There are other surprises too, such as the inclusion of random modifications to the learning algorithm, which result in models that are surprisingly accurate.

A number of techniques have evolved that support ensemble learning, the best known being bagging and boosting. Bagging (Bootstrap Aggregating) is characterised by assigning a vote to each model (in classification problems) and choosing the class with the highest votes. Regression problems can be handled by taking an average of model values. Each model is built using a random sample of instances from the training set, so that each model gets a different set of instances to train on. This works best with unstable modelling techniques such as decision trees, whereas stable methods such as nearest neighbour usually do not benefit from bagging techniques.

Boosting is similar to bagging in that it uses a voting mechanism, although weighting is used so that the best models carry more influence. However the training sets for each model are derived in a wholly different manner. As models are built, emphasis is given to those instances in the training set that are incorrectly classified, and these are given priority in the training sets of new models. Gradient boosting is a particular variation of this approach used in regression problems. A technique known as randomisation supports the random variation of various parameters associated with a learning method. Neural networks for example require a random seed to set initial node weightings. This can be varied as each model is built, and almost all learning methods come with some form of parameterisation, and as such it has wide applicability.
Bagging

Every Bootstrap Aggregating (Bagging) creates multiple bootstrap training sets from the original training set and uses each of them to generate a classifier for inclusion in the ensemble. The algorithms for bagging and sampling with replacement are given in the following lines. In these algorithms, T is the original training set of N examples, M is the number of base models to be learned, Lb is the base model learning algorithm, the h's are the base models, random_integer(a,b) is a function that returns each of the integers from a to b with equal probability, and I(A) is the indicator function that returns 1 if A is true and 0 otherwise.

Bagging(T, M)
For each m = 1,2, K, M
Tm = Sample_With_Replacement(T, 1 T)
hm = Lb(Tm)
M
Return h_{	ext{inf}}(x) = \arg \max_y \in Y \sum I(h_m(x) = y),
m=1
Sample_With_Replacement(T, N)
S = {};
For i = 1,2, K, N
r = \text{random_integer}(1, N) \ Add \ T_r \ to \ S.
Return S.

To create a bootstrap training set from an original training set of size N, we perform N Multinomial trials, where in each trial, we draw one of the N examples. Each example has probability 1/N of being drawn in each trial. The second algorithm given above does exactly this—N times, the algorithm chooses a number r from 1 to N and adds the rth training example to the bootstrap training set S. Clearly, some of the original training examples will not be selected for inclusion in the bootstrap training set and others will be chosen one or more. In bagging, we create M such bootstrap training sets and then generate classifiers using each of them. Bagging returns a function h(x) that classifies new examples by returning the class y that gets the maximum number of votes from the base models h_1, h_2, ..., h_M. In bagging, the M bootstrap training sets that are created are likely to have some differences. If these differences are enough to induce noticeable differences among the M base models while leaving their performances reasonably good, then the ensemble will probably perform better than the base models individually. (Breiman, 1994) demonstrates that bagged ensembles tend to improve upon their base models more if the base model
learning algorithms are unstable—differences in their training sets tend to induce significant differences in the models. He notes that decision trees are unstable, which explains why bagged decision trees often outperform individual decision trees; however, decision stumps (decision trees with only one variable test) are stable, which explains why bagging with decision stumps tends not to improve upon individual decision stumps.
Section 6

Boosting

As CART, boosting is a forward stage-wise additive model. But CART works with smaller and smaller partitions of the data at each stage, whereas boosting uses the entire data set as each stage.

The AdaBoost algorithm trains models sequentially, with a new model trained at each round. At the end of each round, misclassified examples are identified and have their emphasis increased in a new training set which is then fed back into the start of the next round, and a new model is trained. The idea is that subsequent models should be able to compensate for errors made by earlier models.

AdaBoost generates a sequence of base models with different weight distributions over the training set. Its inputs are a set of $N$ training examples, a base model learning algorithm $L_0$, and the number $M$ of base models that we wish to combine. AdaBoost was originally designed for two-class classification problems; therefore, for this explanation we will assume that there are two possible classes. However, AdaBoost is regularly used with a larger number of classes. The first step in AdaBoost is to construct an initial distribution of weights $D_1$ over the training set. This distribution assigns equal weight to all $N$ training examples. We now enter the loop in the algorithm. The AdaBoost algorithm can be written as,

$$
\text{AdaBoost}(\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\},L_0,M)
$$

Initialize $D_1(n)=1/N$ for all $n \in \{1,\ldots,N\}$.

For each $m = 1,2,\ldots,M$:

1. $h_m = L_0(\{(x_1,y_1),(x_2,y_2),\ldots,(x_N,y_N)\},D_m)$.
2. $\epsilon_m = \sum_{n,h_m(x_n)\neq y_n} D_m(n)$.
3. If $\epsilon_m \approx 1/2$ then,
   - set $M = m − 1$ and abort this loop.
4. Update distribution $D_m$:

$$
D_{m+1}(n) = D_m(n) \times \begin{cases} 
\frac{1}{2(1-\epsilon_m)} & \text{if } h_m(x_n) = y_n \\
\frac{1}{2\epsilon_m} & \text{otherwise.}
\end{cases}
$$

Return $h_M(x) = \arg\max_{y \in \mathcal{Y}} \sum_{m=1}^M I(h_m(x) = y) \log \left( \frac{1-\epsilon_m}{\epsilon_m} \right)$.
To construct the first base model, we call L with distribution D over the training set.

After getting back a model h1, we calculate its error e1 on the training set itself, which is just the sum of the weights of the training examples that h1 misclassifies. We require that e1 < 1/2 (this is the weak learning assumption—the error should be less than what we would achieve through randomly guessing the class). If this condition is not satisfied, then we stop and return the ensemble consisting of the previously-generated base models. If this condition is satisfied, then we calculate a new distribution D2 over the training examples as follows.

Examples that were correctly classified by h1 have their weights multiplied by 1/(2(1−e1)).

Examples that were misclassified by h1 have their weights multiplied by 1/(2e1).

Note that, because of our condition e1 < 1/2, correctly classified examples have their weights reduced and misclassified examples have their weights increased. Specifically, examples that h1 misclassified have their total weight increased to 1/2 under D2; and examples that h1 correctly classified have their total weight reduced to 1/2 under D2. We then go into the next iteration of the loop to construct base model h2 using the training set and the new distribution D2. The point is that the next base model will be generated by a weak learner; therefore, at least some of the examples misclassified by the previous base model will have to be correctly classified by the current base model. Boosting forces subsequent base models to correct the mistakes made by earlier models. We construct M base models in this way.

The ensemble returned by AdaBoost is a function that takes a new example as input and returns the class that gets the maximum weighted vote over the M base models, where each base model’s weight is log((1−εm)/εm), which is proportional to the base model’s accuracy on the weighted training set presented to it.

AdaBoost has performed very well in practice and is one of the few theoretically-motivated algorithms that has turned into a practical algorithm. However, AdaBoost can perform poorly when the training data is noisy (Dietterich, 2000), i.e., the inputs or outputs have been randomly contaminated. Noisy examples are normally difficult to learn. Because of this, the weights assigned to noisy examples often become much higher than for the other examples, often causing boosting to focus too much on those noisy examples at the expense of the remaining data. Some work has been done to mitigate the effect of noisy examples on boosting (Ratsch, et al., 2001).
Outlook

The fields of machine learning and data mining are increasingly moving away from working on small datasets in the form of flat files that are presumed to describe a single process. The fields are changing their locus toward the types of data increasingly being encountered today: very large datasets, possibly distributed over different locations, describing operations with multiple regimes of operation, time-series data, online applications (the data is not a time series but nevertheless arrives continually and must be processed as it arrives), partially-labeled data, and documents. Research in model combiners is beginning to explore these new types of data. For example, ensemble learning traditionally has required access to the entire dataset at once, i.e., it performs batch learning. However, this is unrealistic for very large datasets that cannot be loaded into memory all at once.
Conclusions

Model combiners began about decade ago as a new area within machine learning and were driven by the intent to leverage the power of multiple models and not just trust one model built on a small training set. Substantial theoretical and practical developments have happened over the past ten years and have led to several methods, especially bagging and boosting, being used to solve many real problems. However, ensemble methods also appear to be applicable to current and upcoming problems of distributed data mining and online applications. Hence, data analysts should stay tuned for further developments in the exciting field of model combiners.
Bibliography

Statistics, University of California, Berkeley.

Constructing Ensembles of Decision Trees: Bagging, Boosting, and
Randomization, Machine Learning 40, 139-158.

California, Berkeley.


through Combining Predictions. Synthesis Lectures on Data Mining and