Boosting

Thijs Becker,[a] Melvin Geubbelmans,[b] Axel-Jan Rousseau,[b] Dirk Valkenborg,[b] and Tomasz Burzykowski[b,c]
Hasselt, Belgium, and Bialystok, Poland

Boosting is an *ensemble method* in which models are trained sequentially, and their predictions are combined. This approach differs from other ensemble methods such as random forests,[1] in which models are trained separately and their predictions are averaged.

We introduce the concept of boosting using an analogy mentioned in a previous article on supervised learning.[2] In this example, a person had to classify pictures of vehicles into the correct categories (trucks, cars, and motorbikes). After taking a test, it appears the person performs weakly in case of some pictures. Therefore, a smart person would devote additional study time to these pictures to improve an understanding of the content matter and hopefully obtain a better score for the next test. This mechanism of studying harder the concepts that are poorly understood is exactly what boosting is about.

We will use the same classification task discussed in the previous article to illustrate further boosting.[1] Although we use a classification example, note that regression tasks can also be performed with boosting.[3]

A graphical illustration of boosting is shown in the Figure. It consists of training models in subsequent steps.

In the first step, we train a model to predict the values of the dependent variable (target), as shown in Figure *A*. We use a decision tree with 1 node, which corresponds to creating a decision boundary that is a straight line. Why we use this model is discussed later. We need to calculate 2 things for this model. First, we calculate the classification accuracy. Second, we calculate which observations were misclassified.

In the second step, we train a new model (Fig *B*). The model is again a decision tree with 1 node. However, the first step has an important difference: the training data have been modified. Observations misclassified in the previous step are more important for training this model: errors on the observations deemed more important are more strongly penalized in the model-training procedure. The second model, therefore, depends on the first model because the training data are influenced by the (mis)classifications of the first model. This is illustrated in Figure *B*: points that correspond to the more important observations are marked with bigger symbols. After training the model, we calculate the classification accuracy and which points were misclassified. The accuracy is now dependent on how important each point is. Notice that there is a mild analogy with the support points or support vectors in support vector machines. These influential observations completely determine the placement of the decision boundary.

This process can be continued as long as necessary to reach a good predictive model. At each step, we reweigh the importance of the training observations, train a model, and calculate the (weighted) classification accuracy. In our example, we train 10 models, as shown in Figure *C*.

The final boosting model is obtained by adding the predictions of all models. More specifically, we add the predictions but weigh them differently. The higher the (weighted) accuracy of a model, the more it contributes to the final prediction. The final model is shown in Figure *D*. This type of model is called an *additive model* because models are added (eg, instead of averaged).

The performance of boosting is optimal if the model trained at each step is a *weak learner*. A weak learner cannot learn complex patterns in the data, and the performance of individual models is unlikely to be very good. A typical model that is used as a weak learner is a decision tree with a single node. Such models are called *(tree) stumps*. This learning process is an example of *slow learning*.

The algorithm which we described above is called AdaBoost. There are other boosting algorithms. The most popular one is *gradient boosting*. The differences between the various boosting algorithms lie in how the models are added and how the importance of each observation for training the new model in the sequence

[a]VITO, Mol, Belgium.
[b]Center for Statistics, Hasselt University, Belgium.
[c]Medical University of Białystok, Białystok, Poland.
This research received funding from the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program.
Address correspondence to: Tomasz Burzykowski, Hasselt University - Data Science Institute, Agoralaan 1, Building D, B-3590 Diepenbeek, Belgium; e-mail, tomasz.burzykowski@uhasselt.be.
Am J Orthod Dentofacial Orthop 2024;165:122-4
0889-5406/$36.00

https://doi.org/10.1016/j.ajodo.2023.10.003

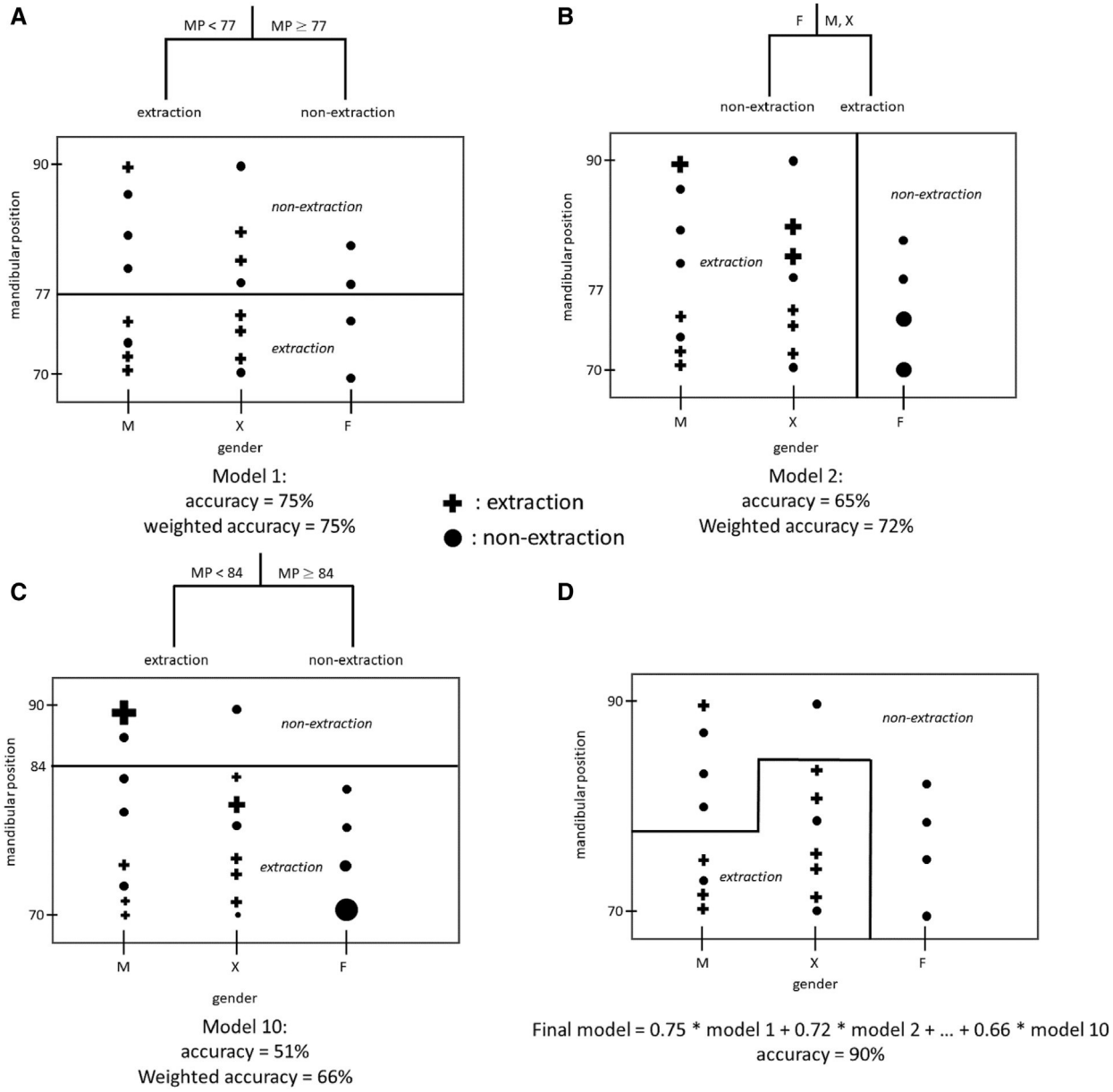

Fig. An example of boosting with decision trees. The mandibular position (MP) and gender (male [M], female [F], other [X]) are the features in the model: **A,** In the first step, a model is trained on the data using a decision tree with a single decision node. Its accuracy is 75%; **B,** In the second step, a second model is trained on the same data, but the importance of each observation is weighed: if it was misclassified by the first model, then it becomes more important and vice versa. Training observations with higher importance are shown as larger symbols. This procedure can be continued for an arbitrary number of models; **C,** There are 10 models in this case. A few observations are hard to classify, which are the most important observations for this final model in the sequence; **D,** The final model is a weighted sum of the models, and better models have a higher contribution.

is calculated. The general philosophy of all these boosting algorithms is the same: train a sequence of weak learners, each focusing on the mistakes made by the previous learners. Therefore, observations that are difficult to learn become more important for learners later in the sequence.

Table. Error table for the data of Konstantonis et al,[7] for a gradient-boosting model

	Predicted class	
True class	*Nonextraction*	*Extraction*
Nonextraction	369	27
Extraction	72	73

Boosting is often the best method for datasets that are not very large and do not consist of images. A neural network (which will be discussed in more detail in one of the next articles in this series on ML) is typically the best model for large datasets or image data. Boosting can require a lot of computation power because it is a slow learning algorithm that requires many models to be trained on the data. The typical number of models is in the hundreds or thousands.

Unfortunately, sampling observations from the data on the basis of whether they are difficult to learn will render the out-of-bag predictions[1] inadequate for estimating the prediction error. For boosting, we need to perform cross-validation to assess the prediction error.[4] The dataset is modified in each step, but the model is still trained on the full training data. This differs from bagging and random forests, as the previous article explains.[1]

Overfitting[4] can be controlled by limiting the number of models trained in the sequence. Another solution is *shrinkage*: when adding the models, the prediction of each new model in the sequence is multiplied by values between 0 and 1. This contributes to models trained later in the sequence as less influential because the repeated multiplication with a number between 0 and 1 leads to increasingly smaller contributions. In practice, the best strategy is to take a small value for shrinkage <0.1[5]; this requires many models to be trained to obtain a good performance because the contribution of an individual model is relatively small.

The use of decision trees as classifiers allows their interesting properties, as discussed in the previous article,[1] to be inherited. The situation is comparable to random forests but less interpretable than a single tree miming human decision-making. Luckily, some methods assign feature importance values to the features used in the boosting algorithm. For instance, partial dependence plots[6] can be used to visualize the behavior of the classifier as a function of 1 or 2 variables, in which the rest of the variables are averaged over. Although these methods provide insight into how the model works, they do not fully remove its "black box" nature.

The performance of a gradient-boosting model trained on the data of Konstantonis et al.[7] is shown in the Table. The performance of the random forest is superior to that of the decision tree. The evaluation was performed using 5-fold cross-validation. In this case, the performance is similar to a random forest model.[1]

To conclude, boosting is a powerful method for classification and regression tasks. It is usually the best method available if the dataset is not large and lacks images. Good implementations are available in Python and R. A popular software library is XGBoost, which provides a highly efficient and portable implementation of gradient boosting. The disadvantages of boosting are a lack of interpretability and higher requirements regarding computation time for training and deploying the model.

REFERENCES

1. Becker T, Rousseau AJ, Geubbelmans M, Burzykowski T, Valkenborg D. Decision trees, bagging, and random forests. Am J Orthod Dentofacial Orthop 2023;164:894-7.
2. Valkenborg D, Geubbelmans M, Rousseau AJ, Burzykowski T. Supervised learning. Am J Orthod Dentofacial Orthop 2023;164:146-9.
3. James G, Witten D, Hastie T, Tibshirani R. An introduction to statistical learning. New York: Springer; 2013.
4. Burzykowski T, Geubbelmans M, Rousseau AJ, Valkenborg D. Validation of machine learning algorithms. Am J Orthod Dentofacial Orthop 2023;164:295-7.
5. Friedman JH. Greedy function approximation: a gradient boosting machine. Ann Statist 2001;29.
6. Biecek P, Burzykowski T. Explanatory model analysis. Boca Raton: CRC Press; 2022.
7. Konstantonis D, Anthopoulou C, Makou M. Extraction decision and identification of treatment predictors in Class I malocclusions. Prog Orthod 2013;14:47.