# More about gradient boosting

Previously, you learned about gradient-boosting machines (GBMs). Gradient boosting is one of the most powerful supervised learning techniques. It's important to understand how gradient boosting works because, as a data professional, you'll likely encounter this type of model frequently. In this reading, you'll review how gradient boosting works and then explore it in greater depth through a worked example.

**Review**

Recall that gradient boosting is a supervised learning technique that uses model ensembling to predict on a target variable. Although GBMs do not have to be tree-based, tree ensembles are the most common implementation of this technique. There are two key features of tree-based gradient boosting that set it apart from other modeling techniques:

1. It works by building an ensemble of decision tree base learners wherein each base learner is trained successively, attempts to predict the error—also known as "residual"—of the previous tree, and therefore compensate for it.
2. Its base learner trees are known as "weak learners" or "decision stumps." They are generally very shallow.

Here is a review of the pseudo-code outline of gradient boosting for an ensemble of just three trees:

```
learner1.fit(X, y)              # Fit the data
$\hat{y}_1$ = learner1.predict(X)      # Predict on X
$error_1 = y - \hat{y}_1$               # Calculate the error → (actual – predicted)
learner2.fit(X, $error_1$)        # Fit tree 2, but target = error from tree 1
$\hat{y}_2$ = learner2.predict(X)      # Predict on X
$error_2 = \hat{y}_2 - error_1$         # Calculate the new error
learner3.fit(X, $error_2$)        # Fit tree 3, but target = error from tree 2
$\hat{y}_3$ = learner3.predict(X)      # Predict on X
$error_3 = \hat{y}_3 - error_2$         # Calculate the new error
```

For these observations and any new samples being predicted by this model:

**Final prediction** = learner1.predict(X) + learner2.predict(X) + learner3.predict(X)

**A worked example**

Here is a demonstration of this process in action. For this scenario, consider a vendor selling bottles of water along a bike path. Below is a table of how many bottles of water he sold on different days and the noontime temperature of each day. The model will try to predict how many water bottles the man sold based on the day's temperature in degrees Celsius.

| Temperature (℃) | Sales (bottles) |
|---|---|
| 14 | 72 |
| 17 | 80 |
| 20 | 98 |
| 23 | 137 |
| 26 | 192 |
| 29 | 225 |
| 32 | 290 |
| 35 | 201 |
| 38 | 95 |
| 41 | 81 |

Step one is to fit a model to the data. This example uses temperature in Celsius as a single X feature and sales as the target variable. The model is a regular decision tree that is only allowed to grow to a maximum depth of one (i.e., it only makes one split), to replicate the weak learners used by GBMs.

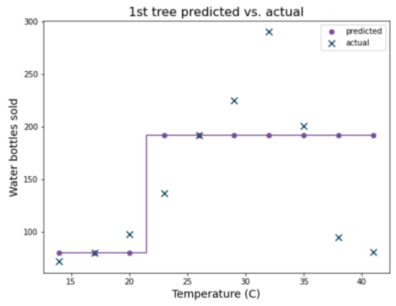
| index | temp (℃) | sales | tree 1 predictions | tree 1 error | tree 2 predictions | tree 2 error | tree 3 predictions | tree 3 error | final predictio |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 14 | 72 | 80 | -8 | 4.5 | -12.5 | -4.5 | -8 | 80 |
| 1 | 17 | 80 | 80 | 0 | 4.5 | -4.5 | -4.5 | 0 | 80 |
| 2 | 20 | 98 | 80 | 18 | 4.5 | 13.5 | -4.5 | 18 | 80 |
| 3 | 23 | 137 | 192 | -55 | 4.5 | -59.5 | -4.5 | -55 | 192 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 4 | 26 | 192 | 192 | 0 | 4.5 | -4.5 | -4.5 | 0 | 192 |
| 5 | 29 | 225 | 192 | 33 | 4.5 | 28.5 | 7 | 21.5 | 203.5 |
| 6 | 32 | 290 | 192 | 98 | 4.5 | 93.5 | 7 | 86.5 | 203.5 |
| 7 | 35 | 201 | 192 | 9 | 4.5 | 4.5 | 7 | -2.5 | 203.5 |
| 8 | 38 | 95 | 192 | -97 | -104 | 7 | 7 | 0 | 95 |
| 9 | 41 | 81 | 192 | -111 | -104 | -7 | 7 | -14 | 95 |

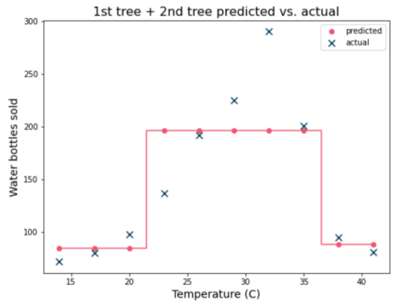
The table above contains information including the temperature, number of water bottles sold, predictions of three base learner trees, their error, and their final prediction for each day. The first tree tries to predict the number of sales. Each subsequent tree tries to predict the error of the tree that came before it.

Take the day at index 0 for example. The number of sales for that day was 72. Tree 1 predicted 80. To calculate the residual error, simply subtract actual minus predicted. In this case: 72 – 80 = -8. Notice that -8 is the value in the “tree 1 error” column. The next tree (tree 2) tries to predict tree 1’s error, and so on. The final prediction represents the sum of the predictions of all three trees. In the case of the day at index 0, this is: 80 + 4.5 – 4.5 = 80.

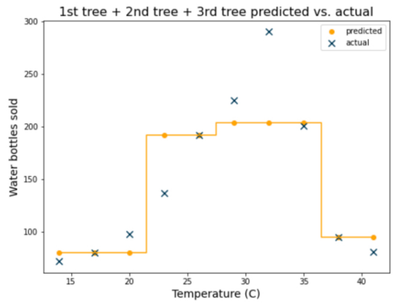
Here are some figures that depict the predicted vs. actual number of bottles sold for each step of the process.



In the first graph, the Xs indicate the actual number of water bottles sold and the purple dots indicate the predicted number. Each vertical part of the line that connects the dots represents a split, or decision boundary, of the model. Notice that for a single tree there’s only one vertical line, because it only makes one split.

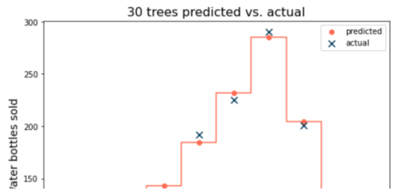


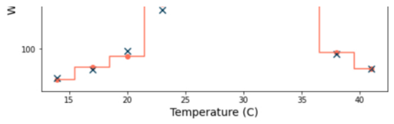
The next graph depicts the predictions of the first two trees vs. the actual values. Each predicted point represents the sum of the predictions of the first and second trees. The predictions match the actual values more closely, but the model still underfits the data.



The graph above depicts the next iteration of the model. Now there are three trees whose predictions are summed. Each additional base learner adds more nuance to the final prediction; it adds a decision boundary (represented here by the vertical segments of the yellow line), which allows the predictions to become more accurate. In this case, every sample is assigned to one of four different values, whereas in the previous example every sample was assigned to one of three different values.

Here is the prediction curve for 30 trees:





The predicted values of this model ensemble very closely match the actual values of the samples. Note, however, that they are not perfect predictions. Indeed, a single decision tree could fit this data perfectly if it were allowed to grow to a depth of five. However, a benefit of ensemble methods like gradient boosting is that they are less likely to overfit the data than a single decision tree.

## Key takeaways

Gradient boosting is a powerful and straightforward technique that uses an ensemble of weak learners to make a final prediction. GBM models are more resilient to high variance that results from overfitting the data due to being comprised of high-bias, low-variance weak learners. The bias of each weak learner in the final model is mitigated by the ensemble.

## Resources for more information

More detailed information about XGBoost can be found here:

- Gradient boosting with scikit-learn
- Gradient boosted decision trees in developers.google.com

Mark as completed

Like Dislike Report an issue