Machine Learning: A brief introduction to Gradient Boosting

Overview: In the **Introduction to Random Forest** tutorial, we reviewed **decision trees**, which are the building blocks for a Random Forest. Decision trees are also the building blocks for Gradient Boosting, but in a very different way.

What is Gradient Boosting?

Like Random Forest, Gradient boosting is also an ensemble of decision trees. Unlike Random Forest however, which is focused on an ensemble of high variance (i.e.: fully-grown) trees, Gradient Boosting is focused on an ensemble of very shallow trees (sometimes called decision *stumps*).

Suppose we have a data set consisting of 100 patients with a tumor diagnosed, which can be further classified into one of two distinct subtypes. **Fig 1** is an example of a very shallow **decision stump** with a goal of predicting whether a patient with a particular tumor has subtype A or B. From this decision tree, we can assume that age has some association with whether a patient has tumor subtype A or B. Recall, the decision tree fitting process will select the best predictor and cut point from the set of available predictors using some criterion of node purity to define "best".

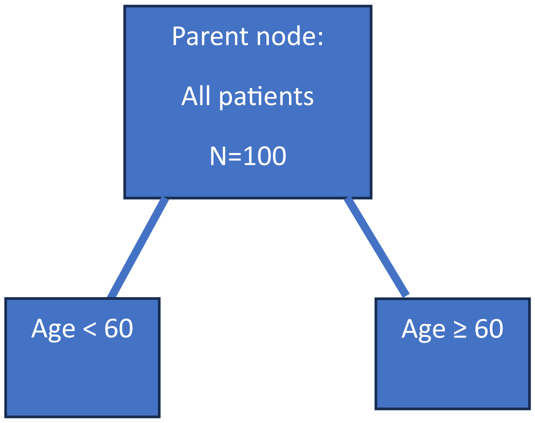

Fig 1. Example of a decision stump.

The decision stumps used in gradient boosting can be, and often are, deeper than this. Especially if we want to allow interactions between variables. For example, if we wanted to model the interaction between age and diabetes status, we could fit a decision tree with an additional level.

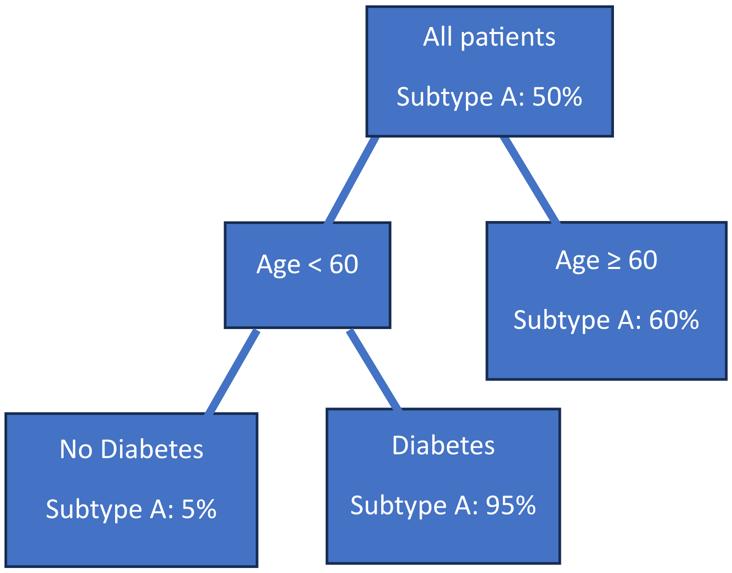

Fig 2. Example of a decision tree with an interaction effect. The presence of diabetes only impacts prediction when the patient is under 60 years old.

A shallow decision stump (like what we have above in **Fig 1 and 2**) is what is called a **low variance, high bias** classifier. This means that if we sampled repeatedly from the target population (theoretically that is; we mean that if we have many similar data sets consisting of a random sample of patients from the population of patients with the disease of interest) and fit a decision stump each time to predict subtype A vs B, we likely would not get much variation in our resulting predictions (low variance). Additionally, we likely would not predict tumor subtype very well (high bias) because we have made the prediction function too simplistic (age < 60 is probably not the only discriminator of whether someone has tumor subtype A or B).

An *extreme* example of a high bias/low variance classifier is if we used the sample proportion to predict subtype. In our sample, we have 40% of patients with subtype A. If we used this as a simple prediction tool, we would simply assign a probability of subtype A of 40% to every new patient. While this is indeed likely to be low variance (predictions based on subtype A proportion would not vary too wildly from 40% in similar samples), it is also high bias because it is likely much too simple of a prediction tool compared to the true function that gives rise to tumor subtypes.

With **Random Forest**, we combine low bias, high variance trees with lots of leaves (nodes) to yield a powerful low bias <u>and</u> low variance (through the averaging process) classifier. Gradient boosting is combining high bias, low variance trees. But unlike Random Forest, Gradient Boosting is <u>not</u> just taking a simple average across the trees in the ensemble to build a powerful classifier. So… what is it doing?

General steps in Gradient Boosting:

1. Predict outcome and compute the log odds of prediction based on the sample proportions in the input data set
2. Compute the residuals from this naïve prediction
3. Build decision tree # 1 using the residuals in the previous step as the outcome variable
4. Obtain the predictions from decision tree # 1 and compute the log odds
5. Use the new residuals to build decision tree # 2
6. Obtain the predictions from decision tree # 2 and compute the log odds
7. Repeat until prediction does not improve

Gradient Boosting is therefore carefully building an ensemble of shallow decision trees that are improving prediction (getting the predicted value closer and closer to the true value) with each new decision tree built from the residuals of the previous tree.

Let's go back to the tumor subtype example:

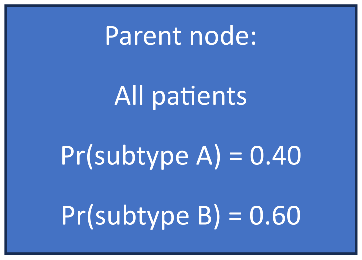

Fig 3. Tumor Subtype example

Let's create an indicator variable for subtype A. Y=1 when a patient has subtype A, 0 is subtype B.

Step 1: **Get the initial prediction and log odds**

Based on the overall proportion of subtype A in the data, our best (naïve) prediction for a new patient using no additional variables would be 0.40, and correspondingly the natural log of the odds would be ln(0.40/(1-0.40)) = ln(0.667) = -0.405.

Step 2: **Compute the residual *(Y_i – $prediction_i$)* for patient i in the sample (i=1,2,…100)**

Recall, patients with subtype A were assigned a value of 1 and subtype B assigned a value of 0. Therefore, the **residuals** of the initial prediction of 0.40 are (1-0.40) = 0.60 for people with tumor subtype A and (0-0.40) = -0.40 for people with tumor subtype B. The prediction of 0.40 is the same for everyone right now because it is based only on the observed proportion of subtype A tumors in the data. Later in the process, the predictions will vary across patients based on their specific covariate patterns.

Step 3: **Build a decision tree using the residuals as the new outcome variable**

It is important to note here that we are using a regression decision tree, not a classification decision tree, because we are modeling the residuals (which will be continuous) we found in step 2. Patients with subtype A have positive residuals while patients with subtype B have negative residuals. The goal here is find a variable that splits the data into two daughter nodes such that large positive residuals will all be in one node and the negative residuals will all be in the other node without much overlap.

We have collected many data points for each patient (demographics, tumor characteristics, comorbidities, etc.) that are potential predictors of the residual. A decision tree automatically finds the best splitting variable and cutpoint at each node to create daughter nodes with lowest mean square error. Recall, we use the Gini Index to measure purity in the classification tree.

Fig 3 shows the resulting decision stump after this process. As you can see, age contains some relevant information because node 1 contains patients < 60 years old and many of the positive residuals. Node 2 contains patients 60 and older and many of the negative residuals.

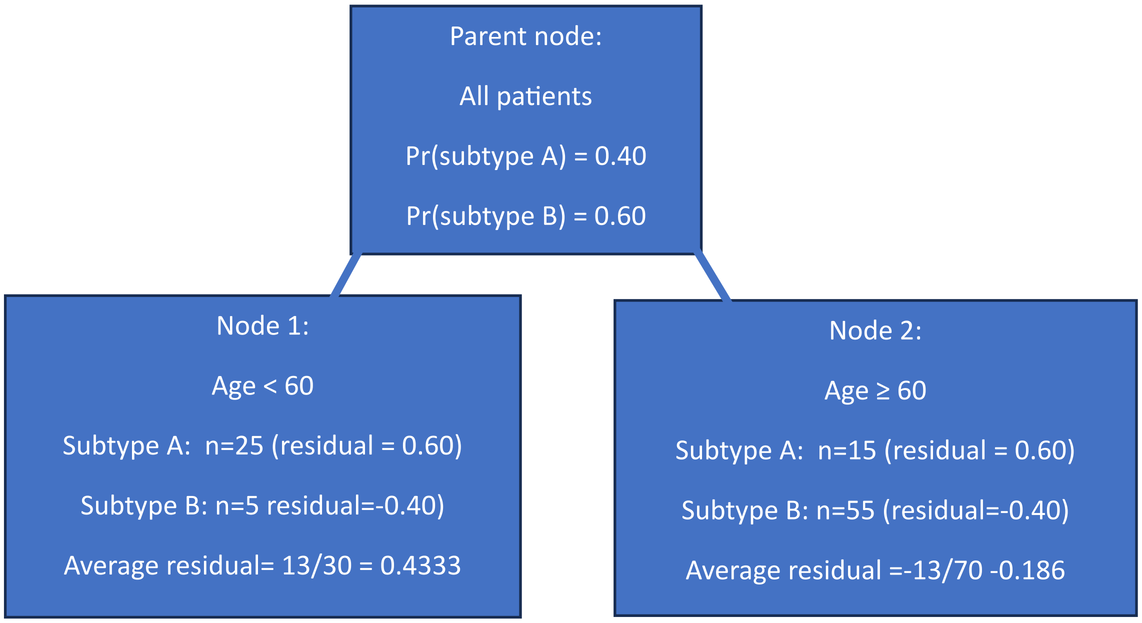
Parent node:
All patients
Pr(subtype A) = 0.40
Pr(subtype B) = 0.60
Node 1:
Age < 60
Subtype A: n=25 (residual = 0.60)
Subtype B: n=5 residual=-0.40)
Average residual= 13/30 = 0.4333
Node 2:
Age ≥ 60
Subtype A: n=15 (residual = 0.60)
Subtype B: n=55 (residual=-0.40)
Average residual =-13/70 -0.186

Fig. 3: The first decision stump for Gradient Boosting

Step 4: **Compute the updated log odds and probabilities**

For each daughter node we first compute the following over all k observations in the node:

$$\gamma = \frac{\sum_{i=1}^{k} residual_i}{\sum_{i=1}^{k} p_i\,(1-p_i)}$$

where p is the previous predicted probability for individual i in the node

In node 1 we get:

$$\gamma = \frac{\left((25\ x\ 0.60) + (5\ x\ {-0.40})\right)}{\left(30\ x\ 0.40\ x\ (1-0.40)\right)} = \frac{13}{7.2} = 1.806$$

And in the right daughter node (node 2), consisting of patients 60+ years old the log odds:

$$\gamma = \frac{(15\ x\ 0.60)\ +\ (55\ x\ -0.40)}{(70\ x\ 0.40\ x\ (1-0.40))} = -13/16.8\ =\ -0.774$$

Finally, we can compute the updated log odds in node k:

Updated log odds in node k = previous log odds + learning rate x γ

The learning rate tells us how much we are going to update those log odds due to the predictions from the new decision stump. We will set it to be 0.10 for now.

Updated log odds *= -0.405 + (0.10 x 1.806) = -0.225* **for node 1**

Updated log odds *= -0.405 + (0.10 x -0.774) = -0.483* **for node 2**

Converting back to probabilities (exp(log odds)/(1+exp(log odds))) we get our new prediction of 44.4% for patients in node 1 and 38.2% for patients in node 2.

For a patient who is 54 years old, the original (naïve) predicted probability for subtype A was 40%, which was updated after the first decision tree to be 44.4%.

While the predicted probability was raised, it was not raised nearly as much as it could have been given the new decision tree has 83% subtype A in node 1. This is where the **learning rate** comes in. Gradient Boosting does not allow each new prediction to change our previous prediction too much, especially when we specify a small learning rate. We therefore need more decision trees to really improve these predictions, especially with smaller learning rates.

Step 5: **Use these new probabilities to compute a new set of residuals and then fit a decision tree.**

For a patient who has subtype A and is 54 years old, the true status is 1.0 and the predicted status is 0.444. The residual therefore is 1-0.444 = 0.556. For a patient who is age 80 years old, the residual is 1-0.3823 = 0.618.

For a patient with subtype B who is 54 years old the residual is 0-0.444 = -0.444 and for a patient who is 80 years old the residual is 0-0.382 = -0.382.

We use these new residuals to compute **decision tree #2**. Suppose we get the tree below, which now separates the data based on diabetes status:

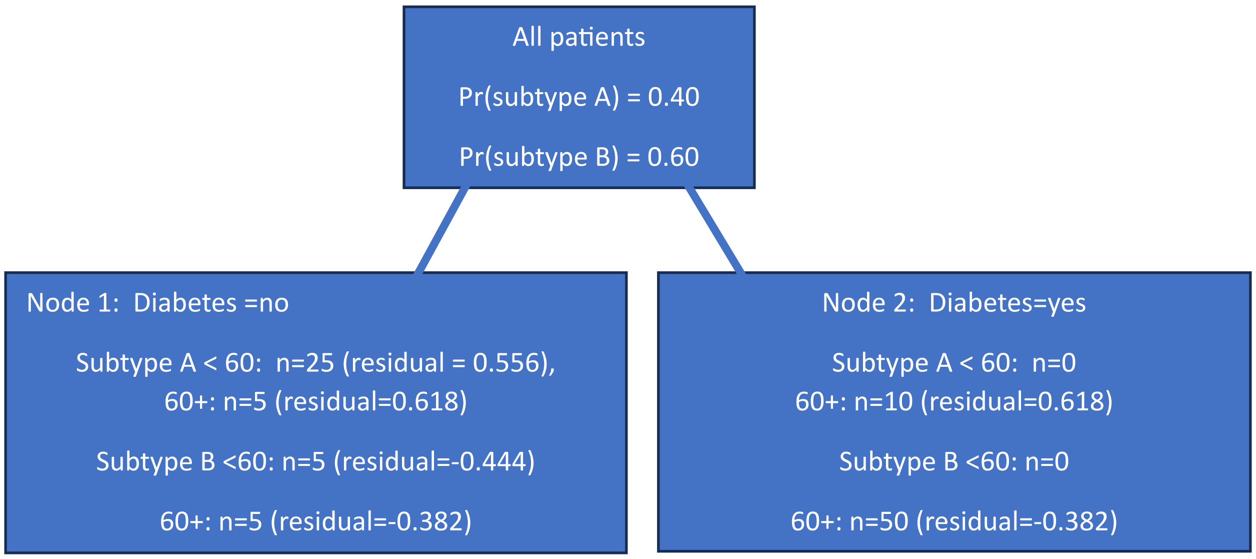
All patients
Pr(subtype A) = 0.40
Pr(subtype B) = 0.60
Node 1: Diabetes =no
Subtype A < 60: n=25 (residual = 0.556), 60+: n=5 (residual=0.618)
Subtype B <60: n=5 (residual=-0.444)
60+: n=5 (residual=-0.382)
Node 2: Diabetes=yes
Subtype A < 60: n=0
60+: n=10 (residual=0.618)
Subtype B <60: n=0
60+: n=50 (residual=-0.382)

Fig. 4: The second decision tree for Gradient Boosting

Step 6: **Obtain the predictions from decision tree # 2 and compute the log odds**

Just as before, we need to compute the γ value for the new nodes. For brevity, we will compute the value for node 1 and leave the computation for node 2 up to you:

For node 1 (diabetes=no):

$$\gamma = \frac{((25\ x\ 0.556) + (5\ x\ 0.618) + (5x(-0.382)) + (5x(-0.444)))}{(30\ x\ 0.444\ x\ (1-0.444) + 10\ x\ 0.382\ x\ (1-0.382))} = 1.317$$

Updated log odds = *-0.225 + (0.10 x 1.317) = -0.093* **for node 1**

Converting back to a probability, we see that for a 54 year-old patient without diabetes, the probability of tumor subtype A is now predicted to be exp(-0.093)/(1+exp(-0.093)) = 0.477

To recap, for this patient, the probability of tumor subtype A was originally 40%, updated to 43.3% after the first decision stump was taken into account, and is now 47.8% after the second decision tree. So moving in the right direction! Additional trees, using other combinations of informative predictors, will likely refine this prediction even more (Step 7 – **repeat until prediction does not improve**).

In summary, Gradient Boosting is a powerful algorithm for classification that uses weak decision trees (sometimes called decision stumps) *successively,* reducing previous errors. **Even if you do not care about the math behind Gradient Boosting, knowing the general algorithm will give you a good sense of what is going on and why a statistician or data scientist may suggest building a prediction model using Gradient Boosting.** Some **hyperparameters** you need to think about for Gradient Boosting are the depth of the trees, how many trees you want to use, and the learning rate.

Where can I read more about these different classifiers?

There are many **excellent online resources** that will allow you to more fully understand machine learning and the various algorithms used.