MA477: Data Science

Lesson 23 Outline — 12 March 2026

United States Military Academy, West Point
Instructor: MAJ Patrick Kuiper

---

1 Administrative

- Calendar review
- Quiz 5
- Student review
- Boosted Tree Discussion
- Coding Exercise

2 Boosting Decision Trees Lesson Objectives

- Understand the algorithm for boosting regression trees and how it differs from the basic decision tree and bagging.
- Use functions in Python's sklearn module to fit boosted regression trees.

3 Boosted Trees: Intuition and Mathematical Framework

- Boosting is an ensemble method that constructs a strong predictor by combining many weak learners.
- In tree-based boosting, the weak learners are small decision trees trained **sequentially**.
- Each new tree attempts to correct the mistakes made by the current model.
- Contrast with bagging and random forests:
  - Bagging / Random Forests: trees trained independently and predictions averaged.
  - Boosting: trees trained sequentially, each improving the previous model.
- The boosted model takes the form

$$f(x) = \sum_{b=1}^{B} \lambda f_b(x)$$

- Variable definitions:
  - $f_b(x)$ = b-th regression tree
  - B = total number of trees
  - λ = learning rate (shrinkage parameter)
- Each tree contributes a small correction to the prediction.

3.1 Sequential Error Correction

- Goal: predict response y using predictors x.
- Start with a simple initial model:
$$f_0(x) = \bar{y}$$
- Variable definition:
  - $\bar{y}$ = mean of the response variable.
- This model predicts the same value for every observation.
- Compute residuals:
$$r_i^{(1)} = y_i - f_0(x_i)$$
- Variable definitions:
  - $r_i^{(1)}$ = residual for observation i
  - y_i = observed response
  - $f_0(x_i)$ = predicted response from the current model
- Interpretation of residuals:
  - $r_i^{(1)} > 0$: prediction too low
  - $r_i^{(1)} < 0$: prediction too high
- Fit a small regression tree to the residuals:
$$h_1(x)$$
- Variable definition:
  - $h_1(x)$ = tree trained to predict residuals
- Update the model:
$$f_1(x) = f_0(x) + \lambda h_1(x)$$
- Variable definitions:
  - $f_1(x)$ = updated prediction function
  - λ = learning rate controlling step size
- Interpretation: the model prediction becomes
  - old prediction + learned correction
- Repeat this process iteratively.
- At iteration b compute residuals:
$$r_i^{(b)} = y_i - f_{b-1}(x_i)$$
- Fit a new tree to residuals:
$$h_b(x)$$
- Update the model:
$$f_b(x) = f_{b-1}(x) + \lambda h_b(x)$$
- After B iterations the final model is
$$f(x) = f_0(x) + \lambda \sum_{b=1}^{B} h_b(x)$$
- Interpretation:
  - boosting gradually improves the model by learning patterns in the remaining errors.

3.2 Connection to Gradient Descent

- Boosting can be interpreted as gradient descent in function space.
- Suppose the loss function is squared error:

$$L = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

- Variable definitions:
  - L = loss function
  - n = number of observations
  - y_i = observed response
  - $f(x_i)$ = model prediction
- Gradient descent improves a model by moving in the direction that reduces the loss most rapidly.
- The derivative of the loss with respect to the model prediction is

$$\frac{\partial L}{\partial f(x_i)} = -2(y_i - f(x_i))$$

- The negative gradient becomes

$$-(\nabla L) = y_i - f(x_i)$$

- Observation:
  - $y_i - f(x_i)$ is exactly the residual.
- Interpretation:
  - fitting a model to the residuals approximates the direction that most rapidly reduces prediction error.
- Each new tree therefore moves the model slightly in the direction that reduces the loss.
- The learning rate λ controls the step size of this gradient descent process.

3.3 Small Numerical Example

Consider the following simple dataset.

x	y
1	3
2	6
4	12
5	15

Step 1: Initial Model

Start with the mean prediction:

$$f_0(x) = \bar{y}$$

Compute the mean

$$\bar{y} = \frac{3 + 6 + 12 + 15}{4} = 9$$

Thus the model predicts 9 for every observation.

Step 2: Compute Residuals

x	y	$f_0(x)$	$r_i = y_i - f_0(x_i)$
1	3	9	-6
2	6	9	-3
4	12	9	3
5	15	9	6

Notice that residuals are negative for small x and positive for large x. This suggests that predictions should increase as x increases.

Step 3: Fit a Simple Tree to Residuals

Suppose a small tree learns the rule

$$h_1(x) = \begin{cases} -4.5 & x < 3 \\ 4.5 & x \geq 3 \end{cases}$$

Step 4: Update the Model

Using learning rate $\lambda = 0.5$

$$f_1(x) = f_0(x) + 0.5h_1(x)$$

Predictions become

x	$f_0(x)$	$0.5h_1(x)$	$f_1(x)$
1	9	-2.25	6.75
2	9	-2.25	6.75
4	9	2.25	11.25
5	9	2.25	11.25

The predictions are now closer to the true values.

The process then continues by computing new residuals and fitting another small tree.

3.4 Key Insight

Residuals represent the direction in which predictions should change in order to reduce error. By repeatedly fitting models to residuals and updating predictions, boosting performs a form of gradient descent that gradually constructs a powerful predictive function.

4 Discussion Questions

Question 1

Why does predicting residuals help improve the model?

Answer

Residuals represent the difference between the true value and the predicted value:

$$r_i = y_i - f(x_i)$$

They therefore indicate how much the prediction should change. If we can predict residuals as a function of x, we can add those predicted residuals to the model to correct systematic errors.

Question 2

How does boosting differ from bagging or random forests?

Answer

Bagging and random forests train trees independently and average their predictions to reduce variance. Boosting trains trees sequentially. Each tree attempts to correct the errors made by the previous model, which primarily reduces bias.

Question 3

Describe the hyperparameters of boost9ing and how you will solve for them?

Answer

The learning rate λ controls the size of each update:

$$f_b(x) = f_{b-1}(x) + \lambda h_b(x)$$

A small learning rate prevents the model from overcorrecting and helps produce a more stable sequence of improvements. Smaller learning rates often require more trees but can improve predictive accuracy.

Question 4

Describe how boosting is similar to gradient descent?

Answer

For squared error loss

$$L = \sum (y_i - f(x_i))^2$$

the negative gradient of the loss is

$$y_i - f(x_i)$$

which is the residual. Thus fitting a model to residuals approximates the direction that most rapidly reduces the loss. Each new tree moves the model slightly in that direction, analogous to gradient descent.