What does “leaf weight” mean?

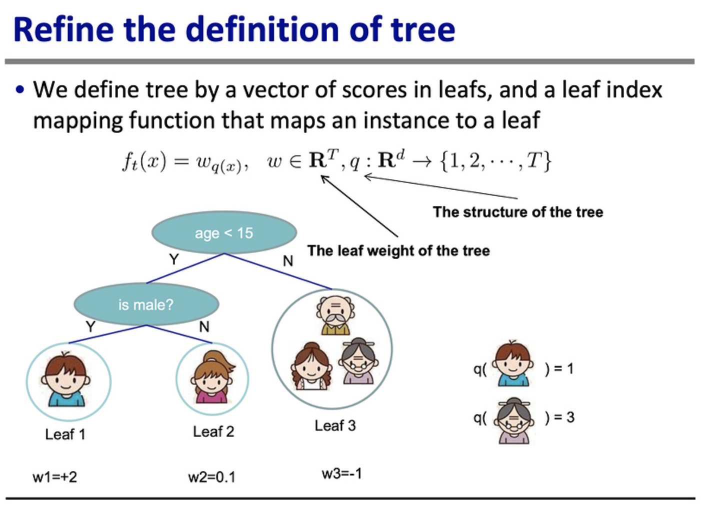
nscozzaro Apr '20

Apr 2020
1 / 7
Apr 2020
May 2020

I looked through Tianqi Chen’s **presentation** , but I’m struggling to understand the details of what the leaf weights are, and I would appreciate if someone could help confirm/clarify my understanding.

To put the equations into words on the slide “Put into context: Model and Parameters”, the predicted value/score (denoted as yhat) is equal to a sum of the K trees of the model, which each maps the attributes to scores. So far so good, I think.

Then in this slide shown below it gives a mock example of a decision tree that calculates the amount that someone likes a computer game X. (Aside: Isn’t this a weird example? Who likes a computer game X by an amount 2? What does that even mean? Why not choose an example that has a more concrete, relatable meaning?)



Refine the definition of tree

- We define tree by a vector of scores in leafs, and a leaf index mapping function that maps an instance to a leaf

$$f_t(x) = w_{q(x)}, \quad w \in \mathbf{R}^T, q : \mathbf{R}^d \to \{1, 2, \cdots, T\}$$

Now this is where I start to get lost. I can tabulate mock data for the mock example, assuming that the model is perfect so that the weights (w1, w2, w3) are equal to the true value. But even that seems weird: what’s the difference between the predicted value and the weights?

```
      x_i: "attributes"          y_i, the true score (not yhat_i, which is the predict
                 |                                 |
|-------------------------------|-----------------------------------|
| Age <15 (x_0) | is male (x_1) | Amount likes the computer game X  |
| 1             | 1             | 2                                 | (Young boy)
| 1             | 0             | 0.1                               | (Young girl)
| 0             | 1             | -1                                | (Old man)
| 0             | 0             | -1                                | (Old woman)
| 0             | 0             | -1                                | (Young woman, old
```

My question is: **Can someone please share what would be the ACTUAL function f?** I assume it’s a vector/matrix, but what are the actual numbers? Then my bonus follow up is how would you compute the f for this mock example? I feel it’s such a simple question, but I can’t figure out the answer. If someone could break this down for me in nitty gritty detail it would be a huge help. Thanks!

hcho3 May '20

The function f is a **step function** , i.e. a function that's constant piece-wise. In the example, you'd have the step function of two variables

$$f(\text{age}, \text{gender}) = \begin{cases} +2 & \text{if age} < 15 \text{ and gender} = \text{Male} \\ +0.1 & \text{if age} < 15 \text{ and gender} \neq \text{Male} \\ -1.0 & \text{if age} \geq 15 \end{cases}$$

The "leaf weight" is a fancy way of saying the real output associated with each leaf (exit) node.

Now how does this step function correspond to the decision tree? Let's say we have a test data point (age=10, gender=female). To get the prediction for the data point, we traverse the
tree from the top to bottom, performing a series of tests. At each intermediate node, we compare a
feature against a threshold. Depending on the result of the comparison, we proceed to either left or
right child node. For (10, female), we'd first perform the test "age < 15" and proceed to the left branch, because "age < 15" is true. Then we perform the second test "gender = male?", which evaluates to false, so we proceed to the right branch. We end up at the Leaf 2, whose output (leaf weight) is 0.1.

Now imagine doing this step for all values in 2D real space, and you get the step function

$$f(\text{age}, \text{gender}) = \begin{cases} +2 & \text{if age} < 15 \text{ and gender} = \text{Male} \\ +0.1 & \text{if age} < 15 \text{ and gender} \neq \text{Male} \\ -1.0 & \text{if age} \geq 15 \end{cases}$$

hcho3 **May '20**

I have put up my master's thesis, and it has pseudocode for the XGBoost algorithm: **https://drive.google.com/file/d/0B0c0MbnP6Nn-eUNRRkVOOGpkbFk/view?usp=sharing&resourcekey=0-nVw3WhovKW5FPvPM5GPHfg** . Hope it helps.

nscozzaro **May '20**

Thank you **@hcho3** .

I have a few followup questions:

1. Given the 5 records of input data above, how would you actually calculate the model for this example? It would be very useful to me to see a worked-through example rather than pseudocode, to help build my understanding.
2. Is there already an example of a simple worked-through xgboost example somewhere you could point me to?
3. I understand that the theoretical model is a decision tree/step function, but I don't think the model is saved as a bunch of if-statements; I assume it's saved as a matrix that transforms a record to the score, right?

nscozzaro **May '20**

> hcho3:
>
> the real output associated with each leaf (exit) node

By the way, is “real output” another way of saying the model’s predicted output, or model score?

---

hcho3 May '20

> nscozzaro:
>
> By the way, is “real output” another way of saying the model’s predicted output, or model score

I mean that each output is a real number.

---

hcho3 May '20

> nscozzaro:
>
> - Given the 5 records of input data above, how would you actually calculate the model for this example? It would be very useful to me to see a worked-through example rather than pseudocode, to help build my understanding.
> - Is there already an example of a simple worked-through xgboost example somewhere you could point me to?

Unfortunately, I don’t have a worked through example on my hand. For now you should look at the algorithm description in my master’s thesis.

Edit. I found **https://github.com/eriklindernoren/ML-From-Scratch/blob/master/mlfromscratch/supervised_learning/xgboost.py** , which is a simplified implementation of the XGBoost algorithm. You can add print() statements in the middle of the program and see what’s going on.

> nscozzaro:
>
> I don’t think the model is saved as a bunch of if-statements

The model is actually saved as a bunch of conditions. The decision tree is a tree graph, where some nodes contain a test condition. You can use either arrays or adjacency graph to represent tree graphs.