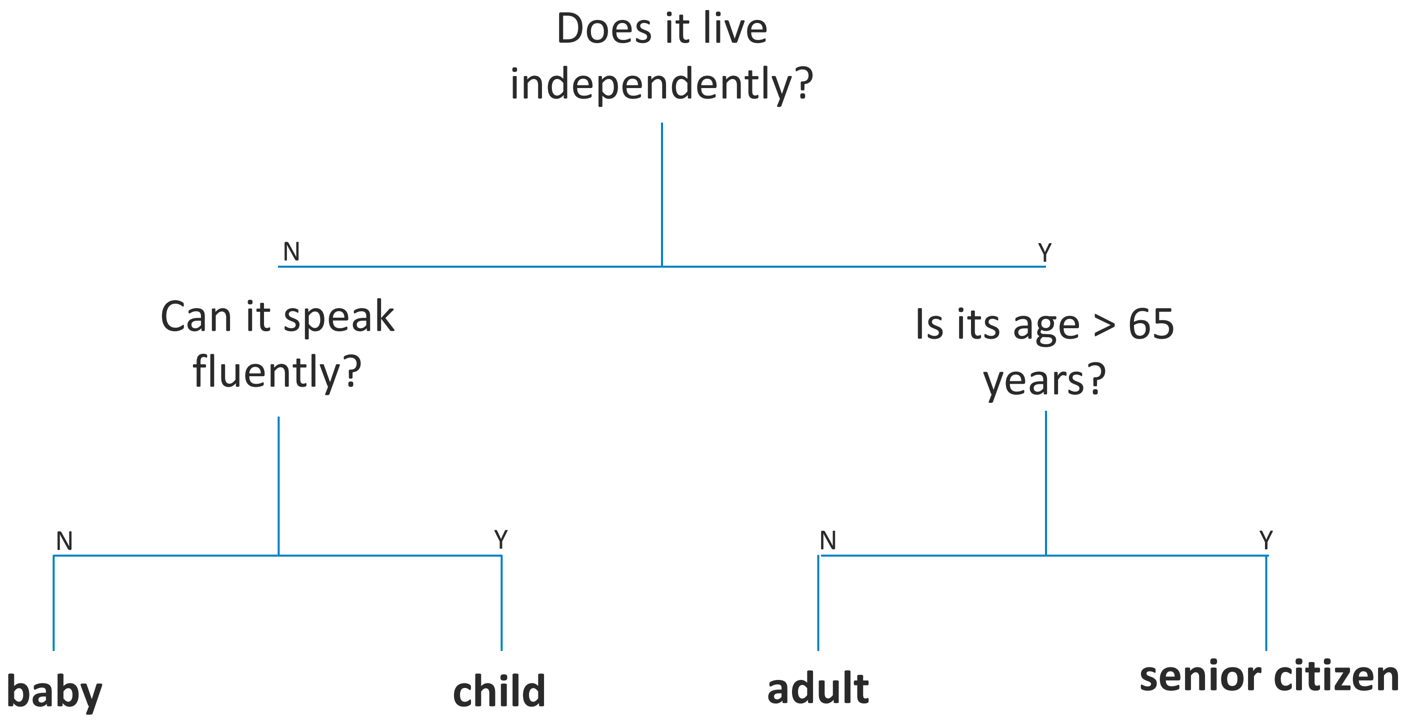
An Introduction to Gradient Boosting Decision Trees

Shrey Gupta, Duke University ’20
hello@shreygupta.me
Summer 2020

Decision Trees



Decision Trees

categorical features

continuous features

algorithm

Predictions are made at the **leaves** of the tree.

Algorithm: C4.5

splitting	information gain	$\max_{features} gain(X, features)$ $= original\ entropy - entropy\ after\ splitting$
stoppage	no more *examples* or *features* to split on	
pruning	compute *upper bounds* on *error* probabilities	1. *do nothing* 2. *collapse tree into leaf* 3. *replace tree with subtree*
missing values	treat as *additional feature*	

Algorithms like **C4.5** and **CART** are highly interpretable…

but prone to **overfitting** and **sensitive** to small perturbations in data (high **variance**).

underfitting

↑

error = bias + variance

↓

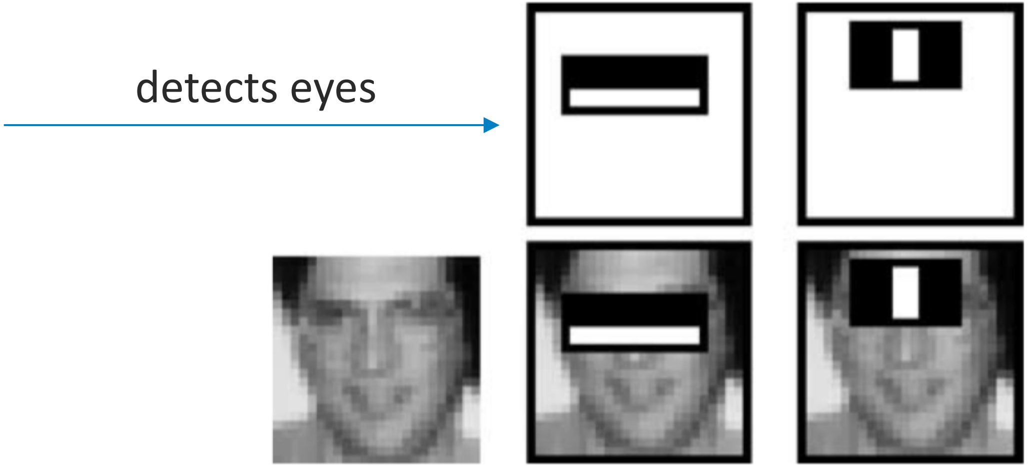
overfitting

high bias,
low variance +
slightly >50%
accuracy

↑

Can we turn several **weak** classifiers into a **strong** one?

Example



Each classifier is **weak** by itself— but several together become **strong**.

Source: Robust Real-Time Face Detection (Viola & Jones)

Algorithm

1. We'll construct m **weak** classifiers.

$$h_0(x), \ldots, h_m(x)$$

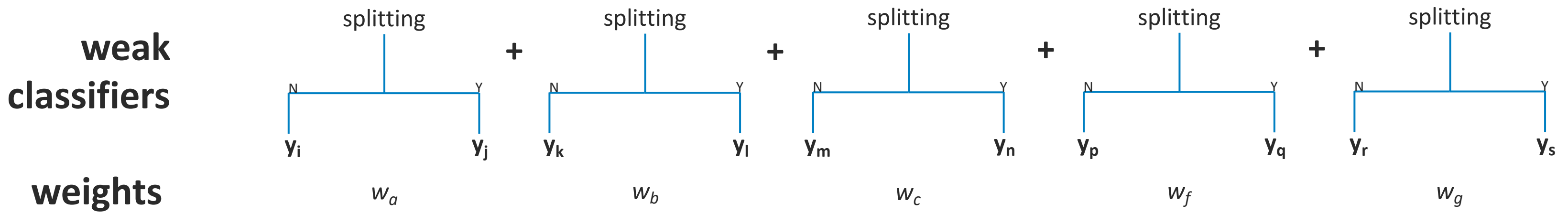
2. Each classifier will be **weighted**.

$$w_0, \ldots, w_m$$

3. The **strong** classifier will be composed of the weighted weak classifiers.

$$H(x) = w_0 \cdot h_0(x) + \ldots + w_m \cdot h_m(x)$$

Gradient Boosting Decision Trees



= **strong** classifier

How do we **create** these weak classifiers?

Gradient Boosting

1. Initialize the model by **choosing the most likely class**.

$$H(x) = h_0(x) = \underset{classes\ c_j}{\arg\min} \sum_{i=0}^{n} L(y_i, c_j)$$

Gradient Boosting

2. Compute the **pseudo-residual** across each data point (x_i, y_i).

$$\rho_i = -\frac{\partial L(y_i, H(x_i))}{\partial H(x_i)}$$

Gradient Boosting

3. Train a **weak classifier** using data points (x_i, ρ_i).

Gradient Boosting

4. Compute the **weight** of the weak classifier by minimizing the loss.

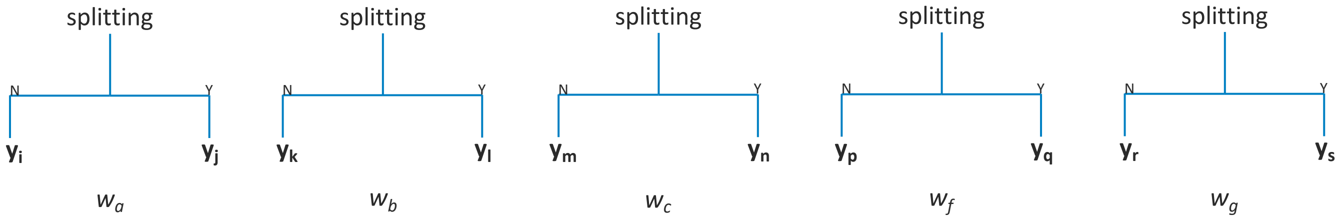
$$w_k = \arg\min_{w} \sum_{i=0}^{n} L(y_i, H(x_i) + w \cdot h_k(x_i))$$

Gradient Boosting

5. Update the **strong** classifier, and **repeat** steps 2-4 *m* times.

$$H(x) \leftarrow H(x) + w_k \cdot h_k(x)$$

In the case of decision trees, the weak classifiers trained will be **small** decision trees (e.g. **stumps**).



Different frameworks grow each weak decision tree **differently**…

two popular open-source frameworks are `XGBoost` and `LightGBM`.

`XGBoost` utilizes **regression** trees (CART)—predictions are the sum of the corresponding leaves. It specifies a **loss function**.

$$L(H) = \underbrace{\sum_{i=0}^{n} d(y_i, H(x_i))}_{\text{penalizes distance}} + \underbrace{\gamma \cdot \Gamma}_{\text{penalizes number of leaves}} + \underbrace{\frac{1}{2}\lambda |w|^2}_{\text{penalizes leaf weights}}$$

penalizes distance

penalizes number of leaves

penalizes leaf weights

`XGBoost`: Gradient Boosting

$$L = \sum_{i=0}^{n} d(y_i, H(x_i) + h_k(x_i)) + \gamma \cdot \Gamma + \frac{1}{2}\lambda|w|^2$$

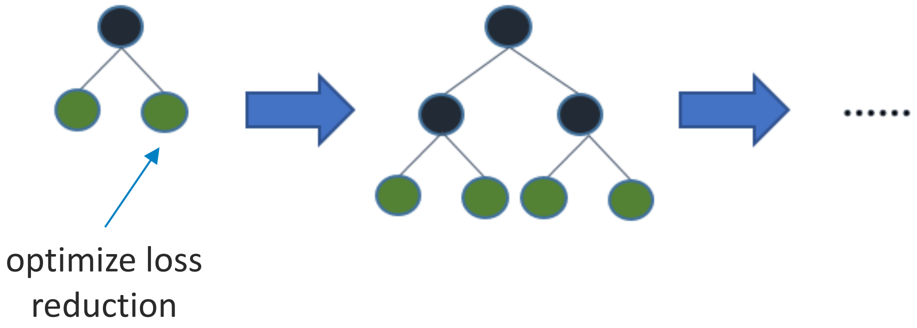
$$\approx \sum_{i=0}^{n}[d(y_i, H(x_i)) + g_i h_k(x_i) + \frac{1}{2}\Lambda_i h_k^2(x_i)] + \gamma \cdot \Gamma + \frac{1}{2}\lambda|w|^2$$

$$= \sum_{i=0}^{n}\underbrace{[g_i h_k(x_i) + \frac{1}{2}\Lambda_i h_k^2(x_i)]}_{\text{movement}} + \underbrace{\gamma \cdot \Gamma + \frac{1}{2}\lambda|w|^2}_{\text{regularization}} + C$$

movement

regularization

`XGBoost`: Gradient Boosting

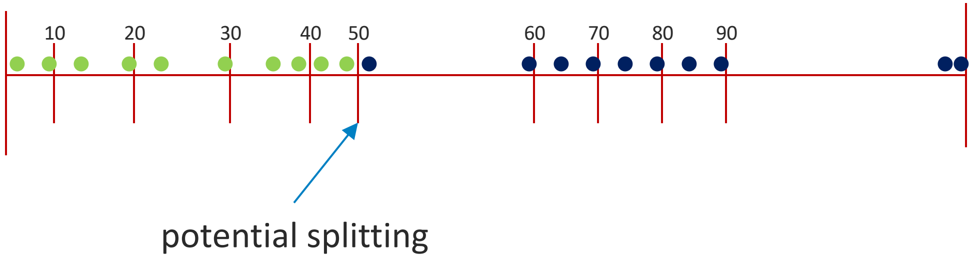


This yields a **closed-form** solution for the leaf weights w_j. Branching is done **level-wise**, **greedily**.

Source: Pushkar Mandot

XGBoost: Gradient Boosting

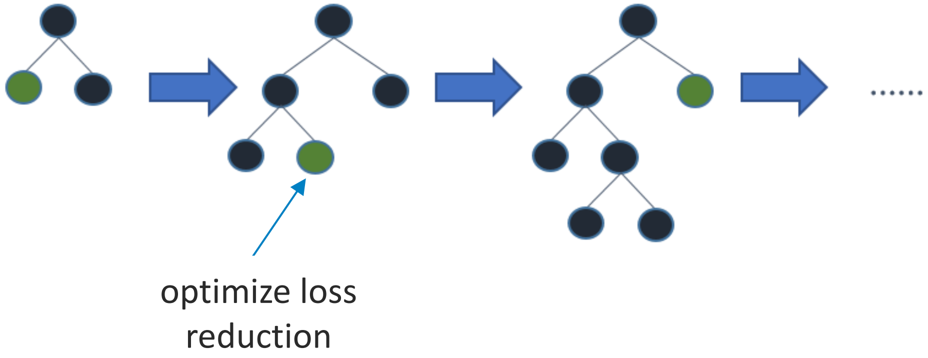
For large datasets, an **approximation algorithm** is used for **splitting**. Features are placed into **buckets** based on **percentile**.



`XGBoost`: Gradient Boosting

The algorithm **learns** the best direction to handle **missing values**—great for **sparse** data.

LightGBM: Gradient Boosting



Branching is instead done **leaf-wise**, **greedily**.

Source: Pushkar Mandot

`LightGBM`: Gradient Boosting

For large datasets, a new dataset is created with instances with **large gradients** and a **subsample** of those with **small gradients**.

`LightGBM`: Gradient Boosting

Again, the optimal direction for **missing values** is **learned**.

`LightGBM`: Gradient Boosting

Additionally, sparse **categorical features** are **bundled** via "exclusive feature bundling."

Important Parameters

	`XGBoost`	`LightGBM`
iterations	number of iterations (trees)	number of iterations (trees)
controls overfitting	learning rate	learning rate
	maximum depth	maximum depth
	minimum child weight	minimum data in leaf
		maximum number of leaves
controls speed	column sample (per iteration)	feature fraction (per iteration)
	subsample ratio (per iteration)	bagging fraction (per iteration)

References

1. *A Gentle Introduction to Gradient Boosting* (Li)

2. *CatBoost vs. Light GBM vs. XGBoost* (Swalin)

3. *Gradient boosting* (Wikipedia)

4. *LightGBM: A Highly Efficient Gradient Boosting Decision Tree* (Ke, et al.)

5. *What is LightGBM, How to implement it? How to fine tune the parameters?* (Mandot)

6. *XGBoost: A Scalable Tree Bosting System* (Chen & Guestrin)

shreygupta.me/phoenix-gbdt