Boosting

Instructor: Junghye Lee

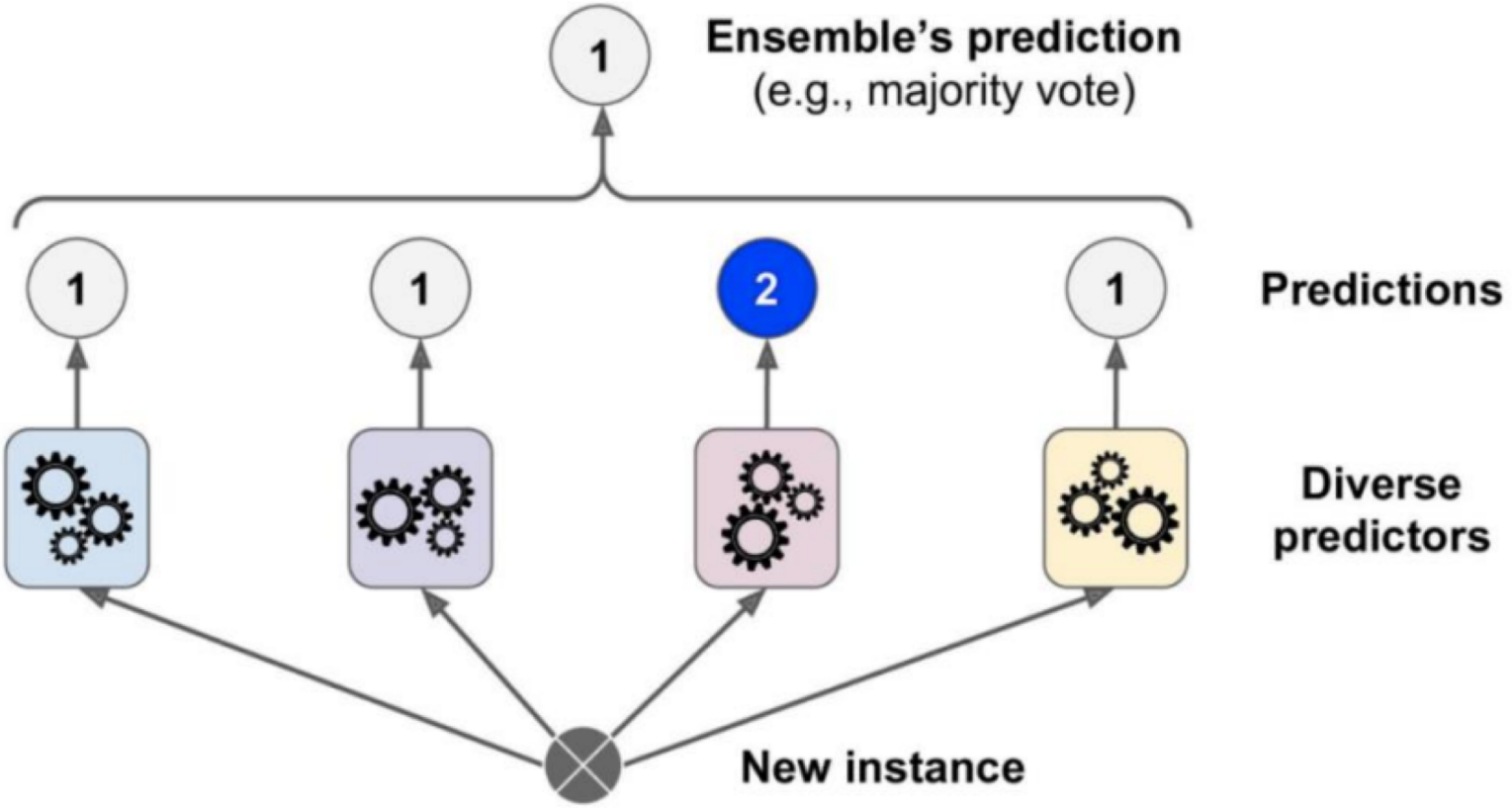
Department of Industrial Engineering
junghyelee@unist.ac.kr

Contents

Ensemble

- In statistics and machine learning, ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.
- Use “multiple models” to predict the output (both regression & classification)
  - Several weak learners become one strong learner.
  - Consider each weak learner as an expert that produces “advice” (a classification result, possibly with confidence).
  - We want to combine their predictions intelligently.
  - Call each weak learner *L* multiple times with a different distribution of the data (perhaps a subset of the whole training set).

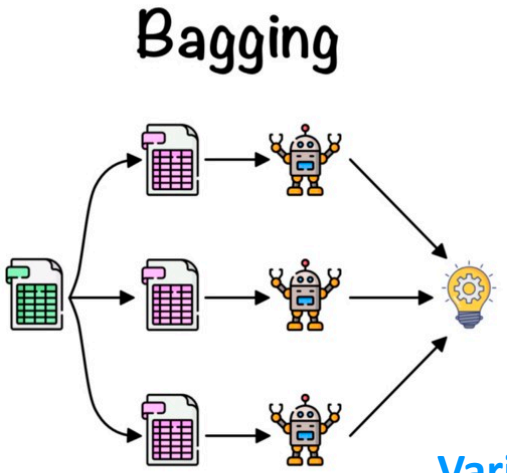
Ensemble



Two approaches: bagging & boosting

Bagging vs. Boosting

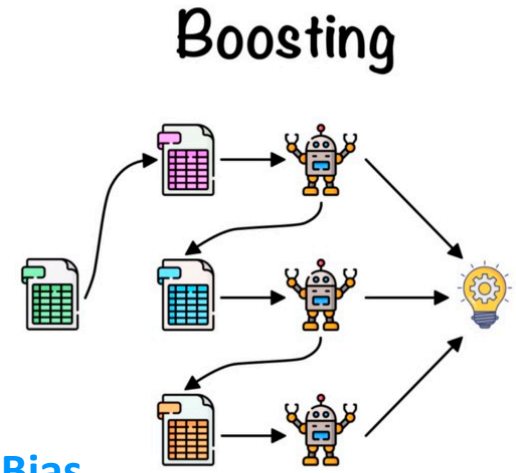
- Bagging: the construction of complementary base-learners is left to chance and to the unstability of the learning methods.
- Boosting: actively seek to generate complementary base-learner - training the next base-learner based on the mistakes of the previous learners.



Parallel

Each model is independent!

Variance vs. Bias



Sequential

Next model is dependent on the previous model!

Steps of Bagging

1) Random sampling with replacement (= Bootstrapping)
2) Select subset of features randomly
3) Grow the tree to the largest
4) Repeat 1)-3) *T* times (total of *T* models)
5) Make prediction based on these *m* models

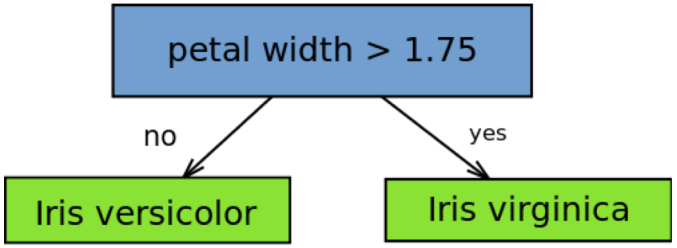
Steps of Boosting

1) First random sample (d1) without replacement
2) Train a weak learner C1
3) Second random sample (d2) without replacement, considering the misclassified sample in learner C1
4) Train a weak learner C2
5) Third random sample (d2) without replacement, considering the misclassified sample in learner C1,C2
6) Train a weak learner C3
7) Combine all the weak learners (C1-Cn) via weighted voting, weighted averaging or thresholding

Adaboost Algorithm for Classification

What is Adaboost?

- Adaboost = Adaptive Boosting
  - Adaptive? “more likely to choose the misclassified samples” from the previous learner
- Base learner: weak learners (not very accurate)
  - ex) Stump (= a tree with 1 node & 2 leaves)
- Weighted sum of the output of weak learners
  - unlike RF, learners get different vote to the final output!
- Weak learners are dependent to each other!
  - made sequentially



A decision stump makes a prediction based on the value of just a single input feature.

Explanation with a Toy Example

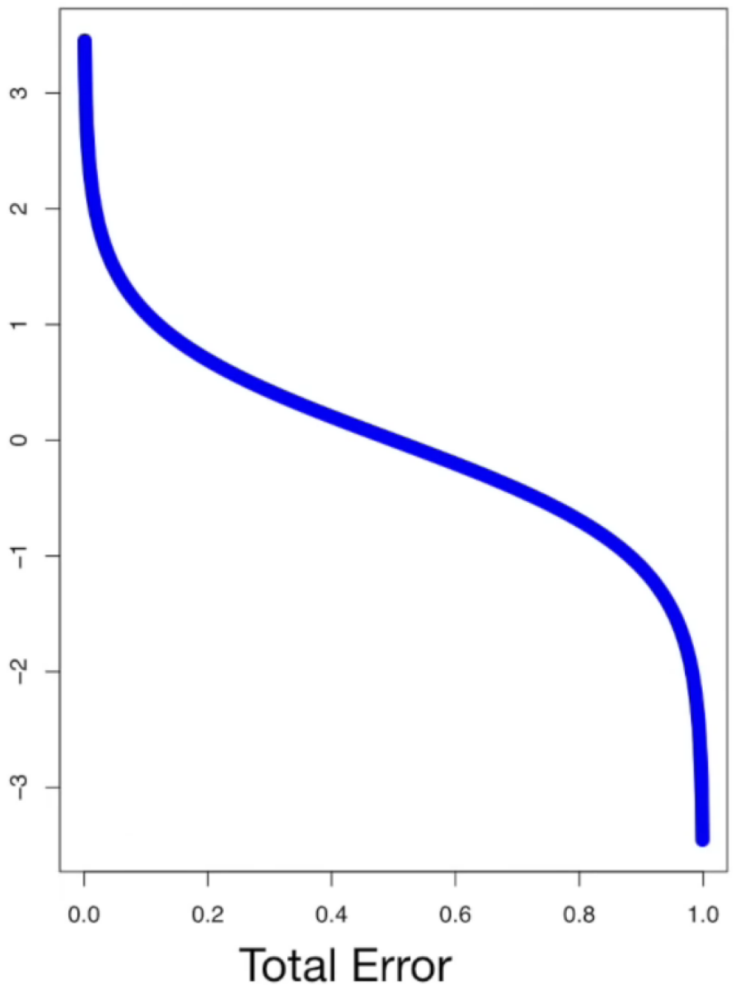
Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
Yes	Yes	205	Yes	1/8
No	Yes	180	Yes	1/8
Yes	No	210	Yes	1/8
Yes	Yes	167	Yes	1/8
No	Yes	156	No	1/8
No	Yes	125	No	1/8
Yes	No	168	No	1/8
Yes	Yes	172	No	1/8

- Start with the equal sample weight.
- For Chest Pain, Blocked Arteries, Patient Weight, make Stumps.
- Calculate the “Amount of Say” for each Stump.

Amount of Say

$$\text{Amount of Say} = \frac{1}{2}\log\left(\frac{1-\text{Total Error}}{\text{Total Error}}\right)$$

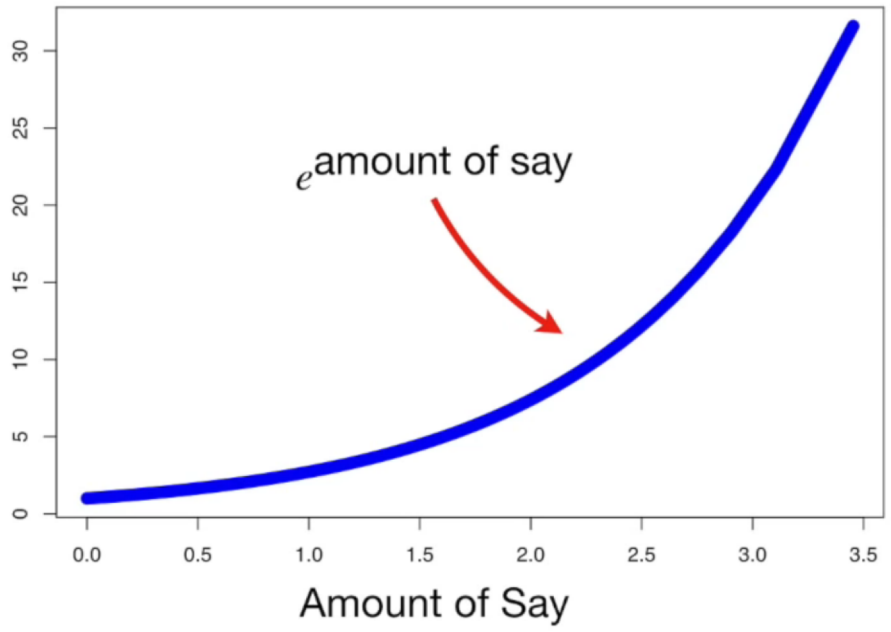
To prevent 0 or 1,
we put a small value ϵ.



- Total Error > 0.5
  - Negative value
- Total Error = 0.5
  - 0
- Total Error < 0.5
  - Positive value

Sample Weights (for Incorrectly Classified Ones)

- New sample weight = old sample weight x $e^{\text{amount of say}}$



  - When the Amount of Say is relatively large, (i.e., the last stump did a good job classifying samples), then we will scale the previous sample weight to a much larger one than the old one.

What about correctly classified samples?

Sample Weights (for Correctly Classified Ones)

- New sample weight = old sample weight x $e^{-\text{amount of say}}$
- Then

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	New Weight	Norm. Weight
Yes	Yes	205	Yes	0.05	0.07
No	Yes	180	Yes	0.05	0.07
Yes	No	210	Yes	0.05	0.07
Yes	Yes	167	Yes	0.33	0.49
No	Yes	156	No	0.05	0.07
No	Yes	125	No	0.05	0.07
Yes	No	168	No	0.05	0.07
Yes	Yes	172	No	0.05	0.07

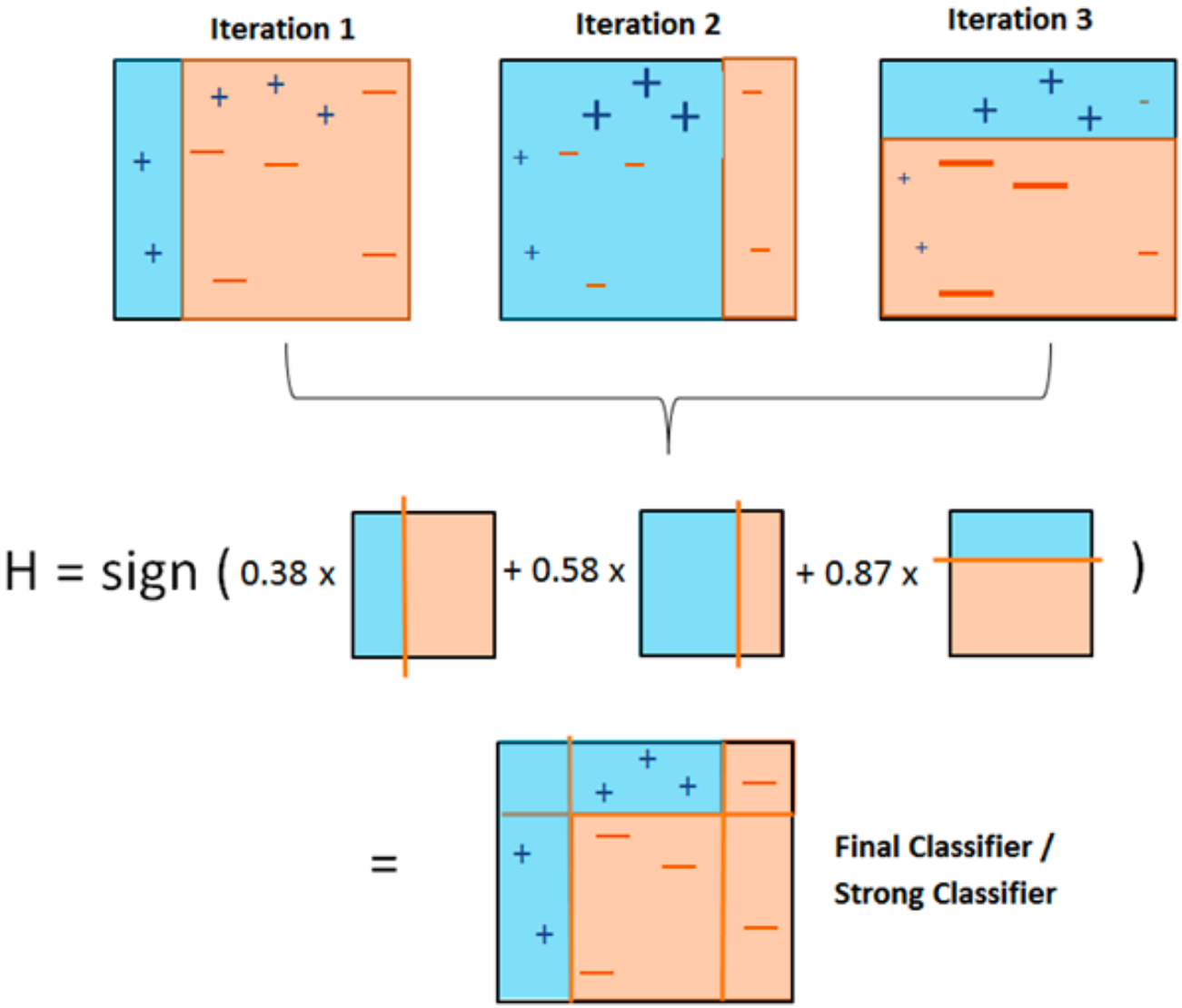
How We Can Use These Sample Weights?

- In theory, we could use the sample weights to calculate weighted Gini Indices to determine which variable should split the next stump.

- The weighted Gini index would put more emphasis on correctly classifying this sample (the one that was misclassified by the last stump), since this sample has the largest sample weight.

- Alternatively, instead of using a weighted Gini index, we can make a new collection of samples that contains duplicate copies of the samples with the largest sample weights.

Adaboost Illustration with Weighted Gini Index



A New Collection of Samples

- Bagging with sample weights
  - We use the weights to determine the interval between [0, 1] corresponding to the sample to be sampled.

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weights
Yes	Yes	205	Yes	0.07
No	Yes	180	Yes	0.07
Yes	No	210	Yes	0.07
Yes	Yes	167	Yes	0.49
No	Yes	156	No	0.07
No	Yes	125	No	0.07
Yes	No	168	No	0.07
Yes	Yes	172	No	0.07

A New Collection of Samples

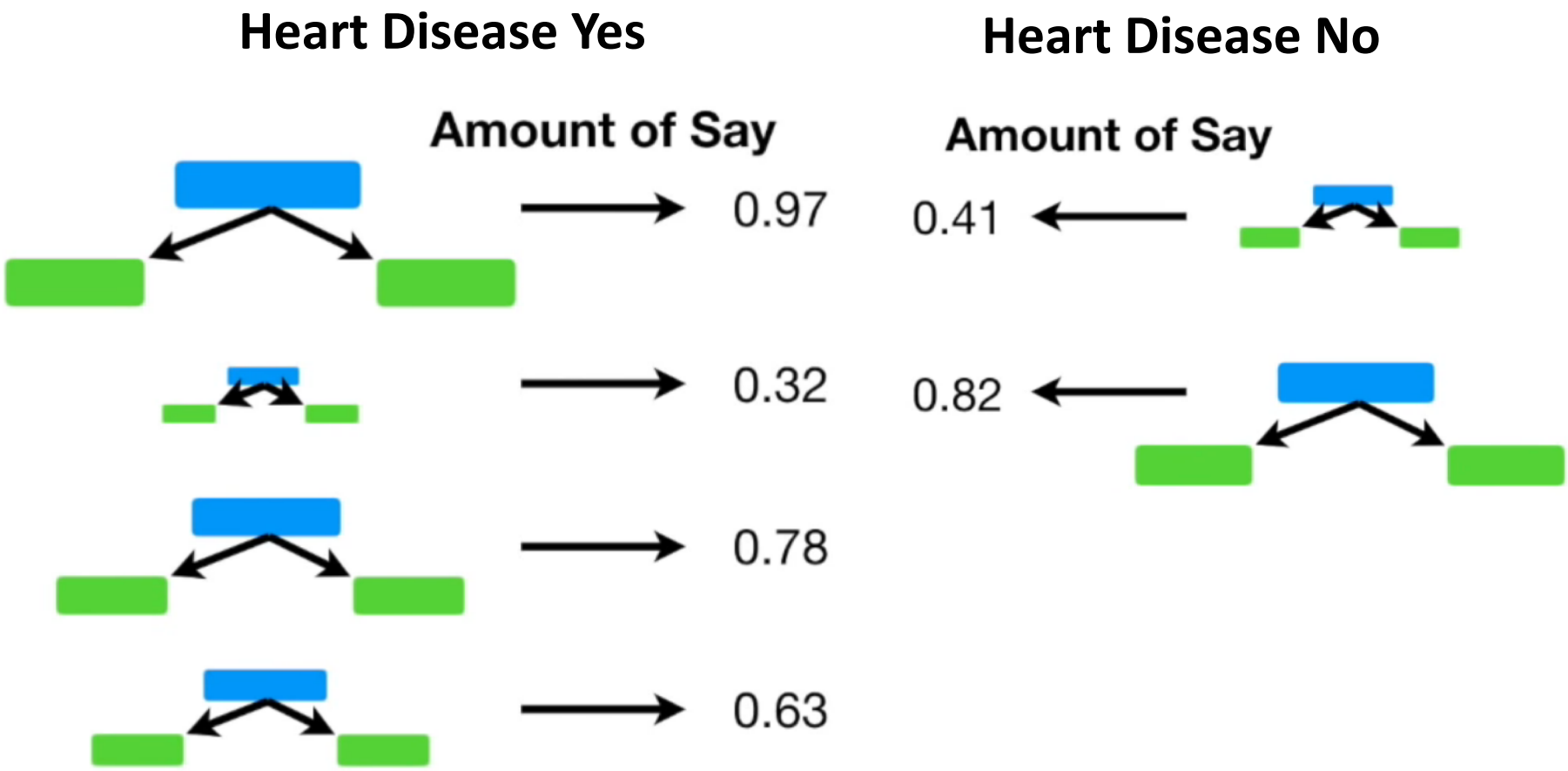
- Start with the same-sized blanks.
- Generate a random number from the uniform distribution.
- Fill in the blank in a row-wise manner.

reset

Chest Pain	Blocked Arteries	Patient Weight	Heart Disease	Sample Weight
No	Yes	156	No	1/8
Yes	Yes	167	Yes	1/8
No	Yes	125	No	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	167	Yes	1/8
Yes	Yes	172	No	1/8
Yes	Yes	205	Yes	1/8
Yes	Yes	167	Yes	1/8

You can repeat this *N* times!

Inference



You can add up all the amount of says to get conclusion!

Adaboost Algorithm

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in \mathcal{X}$, $y_i \in \{-1, +1\}$.
Initialize: $D_1(i) = 1/m$ for $i = 1, \ldots, m$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution D_t.
- Get weak hypothesis $h_t : \mathcal{X} \to \{-1, +1\}$.
- Aim: select h_t with low weighted error:

$$\varepsilon_t = \Pr_{i \sim D_t} \left[h_t(x_i) \neq y_i \right].$$

- Choose $\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \varepsilon_t}{\varepsilon_t} \right)$.
- Update, for $i = 1, \ldots, m$:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

where Z_t is a normalization factor (chosen so that D_{t+1} will be a distribution).

Output the final hypothesis:

$$H(x) = \operatorname{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Practical issues

- For binary problems, weak learners have to have at least 50% accuracy (better than chance).
- Running time for training is O(mT), for m samples and T iterations.
- How do we choose T?
  - Trade-off between training time and accuracy
  - Best determined experimentally

Gradient Boost

Gradient Boost

- Gradient Boosting
  - Not so much different with Adaboost

    1. Combine weak learners to make classifications
    2. Some learners have more say than over learners
    3. Each learner is made by considering the result of the previous learner's mistake.

  - **Can use any cost function** (i.e., can be expressed in more general terms by using "gradient" to minimize the cost function)

  - **Using trees larger than stumps as base leaners**

  - **Scale factor**

Gradient Boosting for both Regression and Classification

A Toy Example for Regression

Height	Favorite Color	Gender	Weight
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

- We use the following differentiable loss function:

$$\frac{1}{2}(\mathbf{observed} - \mathbf{predicted})^2 = \frac{1}{2}(\boldsymbol{y} - \hat{\boldsymbol{y}})^2$$

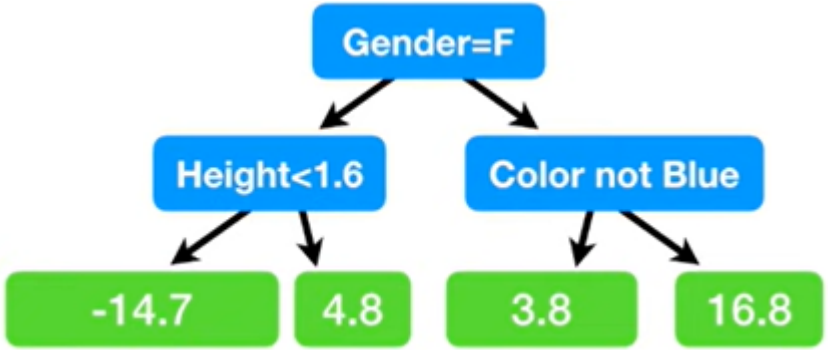
- Build the optimal prediction model without considering the independent variables.
  - Averaged weight: 71.2

Gradient Boost for Regression

- Calculate the residual based on the first prediction model.

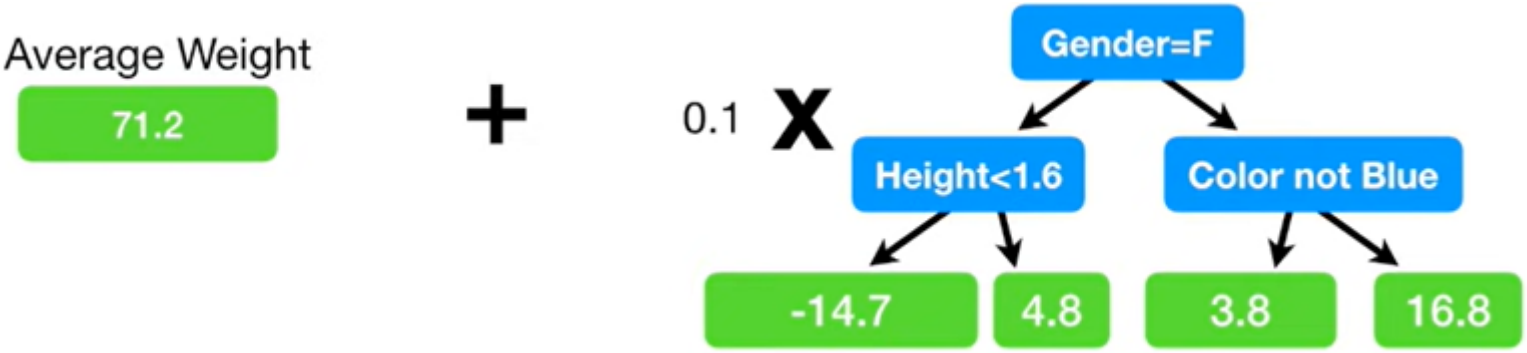
Height	Favorite Color	Gender	Weight	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

- We build a decision tree for the residual using factors.



Gradient Boost for Regression

- We sequentially add the prediction models.
  - Here, we scale the effect of other decision trees except for the first prediction model using α (learning rate; user-defined parameter).
  - $\alpha = 0.1$ in this example
  - Why?



- Compute the residual again.

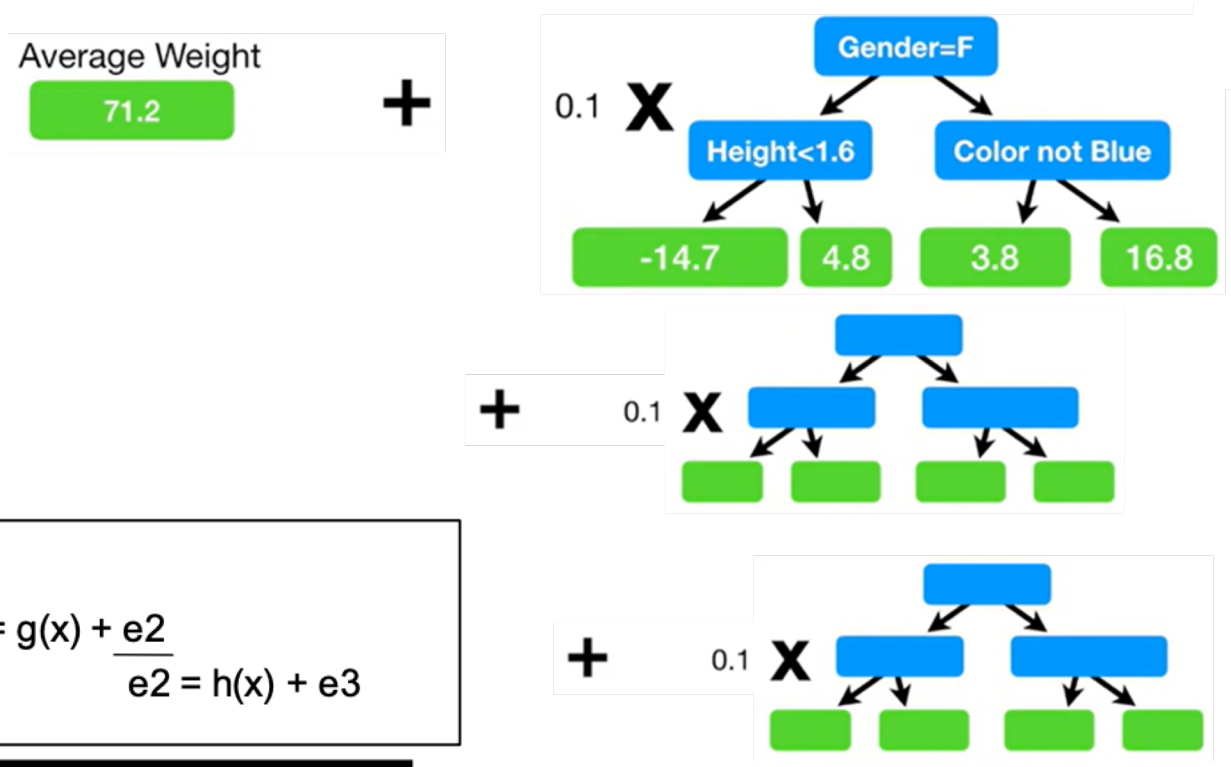
Gradient Boost for Regression

Height	Favorite Color	Gender	Weight	Residual 1	Residual 2
15.1	Blue	Male	88	16.8	15.1
1.6	Green	Female	76	4.8	4.3
1.5	Blue	Female	56	-15.2	-13.7
1.8	Red	Male	73	1.8	1.4
1.5	Green	Male	77	5.8	5.4
1.4	Blue	Female	57	-14.2	-12.7

Residuals decrease
which is the right direction

Gradient Boost for Regression

- We keep making trees until
  - we reach the maximum iterations specified, or
  - Adding additional trees does not significantly reduce the size of the residuals

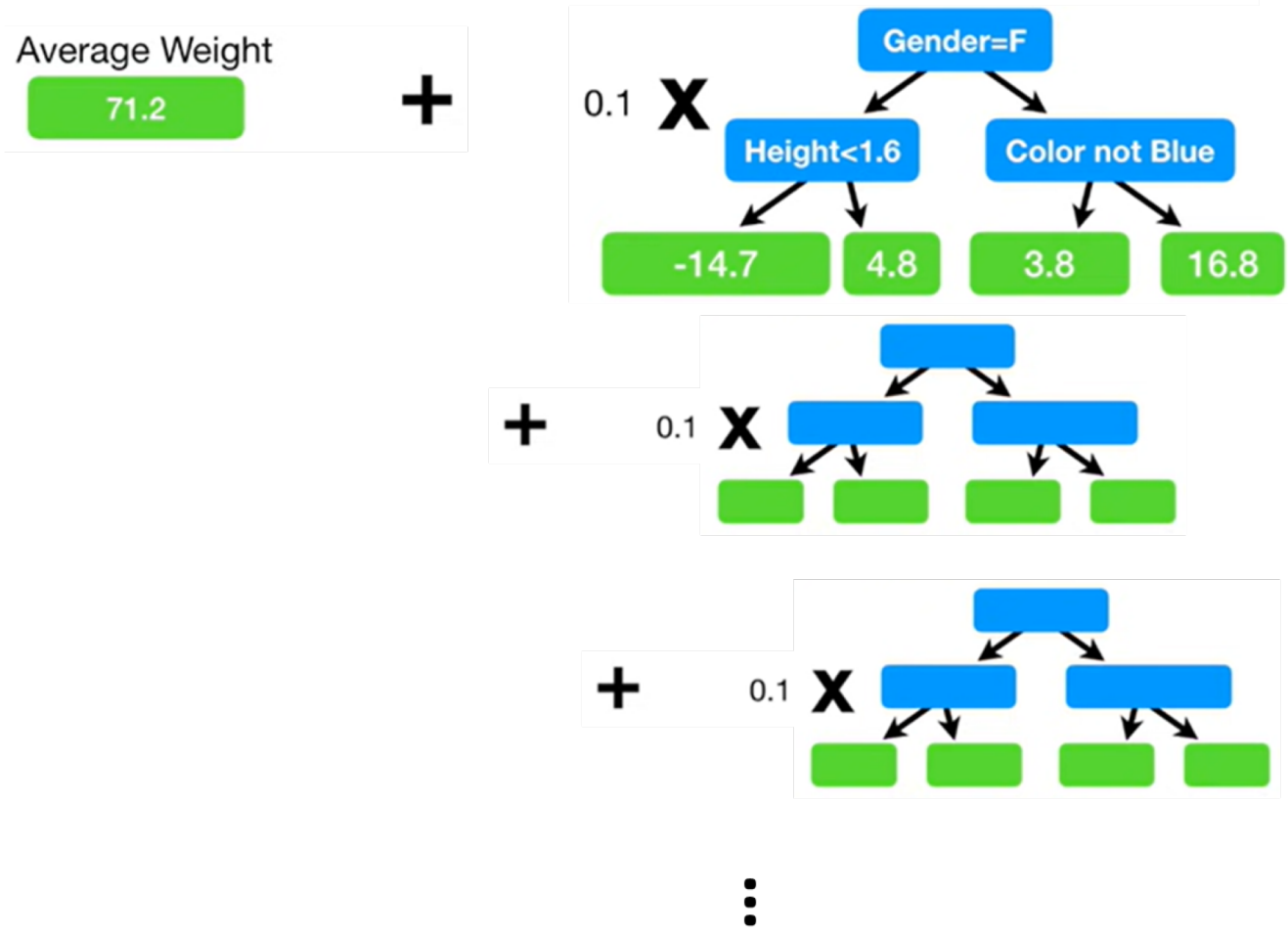


$$y = f(x) + e1$$
$$e1 = g(x) + e2$$
$$e2 = h(x) + e3$$

$$Y = f(x) + g(x) + h(x) + \ldots$$

Inference

- When a new sample comes $\boldsymbol{x}^*$
  - You just walk through the set of trees to get the outcome $\hat{\boldsymbol{y}}^*$



Gradient Boosting Algorithm (ver. 1)

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$. **How come?**
2. For $m = 1$ to M:
   (a) For $i = 1, 2, \ldots, N$ compute
   $$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$ **What is it?**
   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.
   (c) For $j = 1, 2, \ldots, J_m$ compute
   $$\gamma_{jm} = \arg\min_\gamma \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$
   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$. $\boldsymbol{\alpha = 1}$
3. Output $\hat{f}(x) = f_M(x)$.

Gradient Boosting Algorithm (ver. 2)

Gradient Boosted Regression Tree

1. **Input:** $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}, \theta, \gamma$
2. **Output:** $F(\mathrm{x}) = \sum_{i=0}^{M} F_i(x)$
3. Initialize $F_0(x) = \arg min_\beta \sum_{i=0}^{N} L(y_i, \beta)$
4. **While** $(m < M)$
5. $\quad d_i = -[\partial L(y_i, F(x_i))/\partial F(x_i)]_{F(x_i)=F_{m-1}(x_i)}$
6. $\quad \vartheta = \{(x_i, d_i)\}, i = 1, N$
7. $\quad g(\mathrm{x}) = \mathrm{FITREGRETREE}(\vartheta, \theta)$
8. $\quad \rho_m = \arg min_\rho \sum_{i=1}^{N} L(y_i, F_{m-1}(x) + \rho g(x))$
9. $\quad F_m(x) = F_{m-1}(x) + \gamma \rho_m g(x)$
13. **End while**

Practical Issues

- Accuracy vs. Complexity
- How many leaves?
  - In practice, the maximum number of leaves is often set to be between 8 to 32.
  - Depending on your data (size)!

A Toy Example for Classification

- (Different) We use the loss function for classification.
- (Same) But we use the regression tree considering “residual”.
  - **Residuals involve numeric meaning.**

Likes Popcorn	Age	Favorite Color	Weight
Yes	12	Blue	Yes (1)
Yes	87	Green	Yes (1)
No	44	Blue	No (0)
Yes	19	Red	No (0)
No	32	Green	Yes (1)
No	14	Blue	Yes (1)

- (Different) We should consider the probabilistic characteristics of predicted values to combine the trees.

A Toy Example for Classification

Likes Popcorn	Age	Favorite Color	Weight
Yes	12	Blue	Yes (1)
Yes	87	Green	Yes (1)
No	44	Blue	No (0)
Yes	19	Red	No (0)
No	32	Green	Yes (1)
No	14	Blue	Yes (1)

- The very first prediction model:

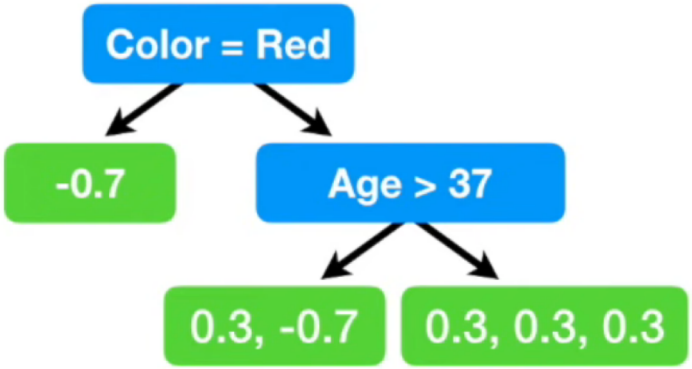
$$\log\left(\frac{P}{1-P}\right) = 0.7$$

- Calculate the residuals: observed − predicted

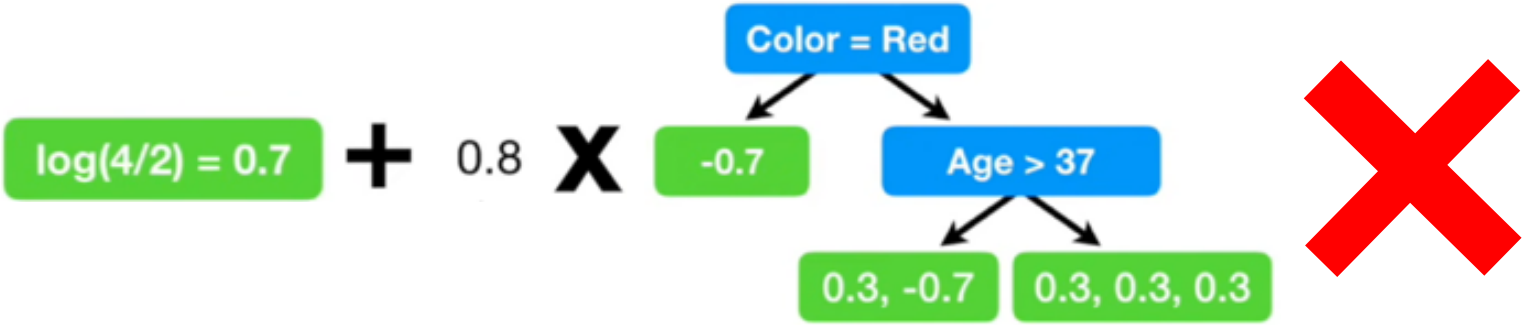
Residual 1
0.3
0.3
-0.7
-0.7
0.3
0.3

Gradient Boost for Classification

- We build a decision tree for the residual using factors.
  - Decision tree for regression



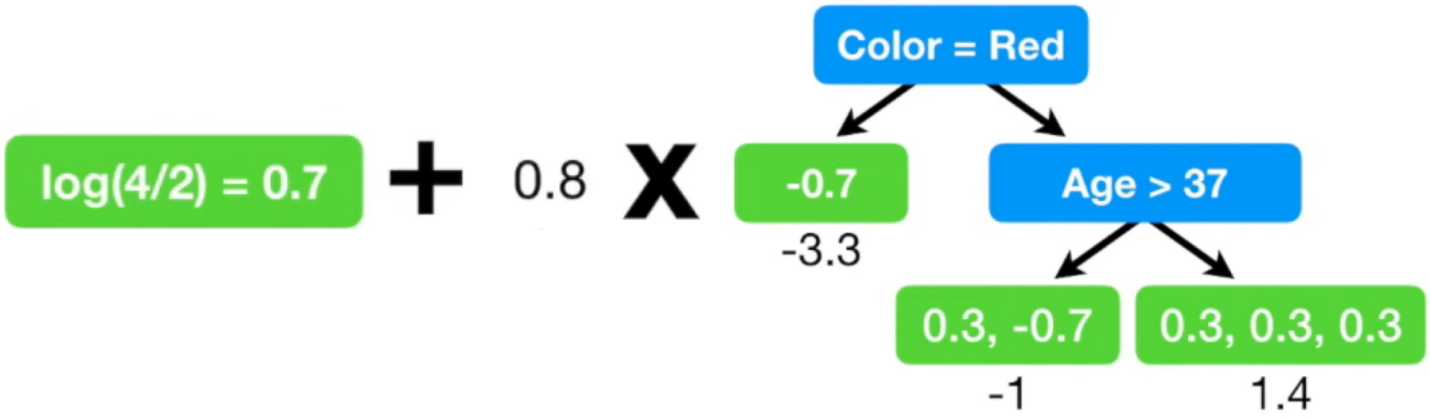
- Then, can we just add this to the first prediction like before?
  - The answer is No!



Gradient Boost for Classification

- To make the value to be log(odds), we have to adjust the value by using

$$\frac{\sum \text{Residual}_i}{\sum[\text{Previous Probability}_i(1 - \text{Previous Probability}_i)]} \quad (1)$$



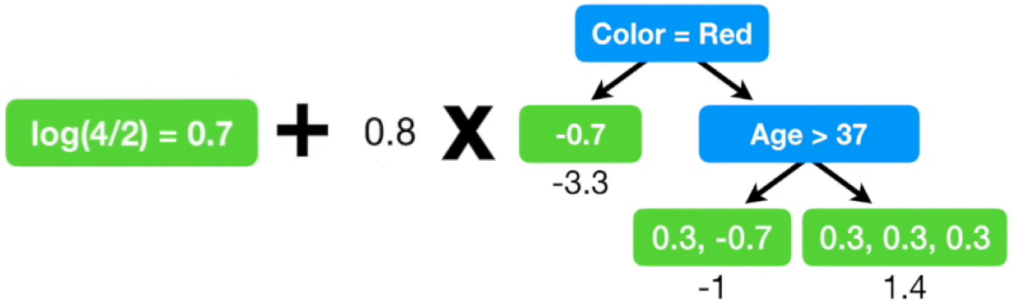
  - Why this equation? beyond our scope

Gradient Boost for Classification

- Then, we transform this value into the probability using the logistic function:

$$\frac{\exp(x)}{1+\exp(x)}$$

Likes Popcorn	Age	Favorite Color	Weight
Yes	12	Blue	Yes (1)



  - Prediction (before): 0.5
  - Prediction (after): 0.7 + (0.8 x 1.4) = 1.82
  - Convert 1.82 (log odds ratio) into 0.9 (probability)

Regression targets residual (continuous value)

Residual = label – probability (obtained from log(odds))

Gradient Boosting Algorithm

1. Initialize $f_0(x) = \arg\min_\gamma \sum_{i=1}^{N} L(y_i, \gamma)$.

2. For $m = 1$ to M:

   (a) For $i = 1, 2, \ldots, N$ compute

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}.$$

   (b) Fit a regression tree to the targets r_{im} giving terminal regions R_{jm}, $j = 1, 2, \ldots, J_m$.

   (c) For $j = 1, 2, \ldots, J_m$ compute

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \gamma\right).$$

   (d) Update $f_m(x) = f_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$.

3. Output $\hat{f}(x) = f_M(x)$.

What is the loss function?

What should be the residual in classification?

Then, what should be the F function?

It is not easy in classification (loss function) which however turns out to be (1).

Variants of Gradient Boosting

- XGBoost
  - Property: parallel computing firstly introduced
  - Cons: still slow
- LightGBM
  - Property: leaf-wise rather than level-wise (complexity ↓; memory and speed) + parallel computing
  - Cons: overfitting
- CatBoost
  - Property: partial residuals (resolving overfitting), handling categorical variables efficiently + parallel computing
  - Cons: could be degraded in datasets with many continuous variables

Questions?