GBRT를 소개합니다

Web & Image 임민섭

Why before What?

- 2010 Yahoo L2R Challenge
  - 36k 질의, 883k 문서, 700개의 feature를 사용

Why before What?

- 2010 Yahoo L2R Challenge
  - 36k 질의, 883k 문서, 700개의 feature를 사용
  - Challenge 내 몇가지 특징들:
    (1) 탑 랭커들은 decision tree를 주로 사용했다. (상위 5 팀이 모두 앙상블 트리를 사용함)

Why before What?

- 2010 Yahoo L2R Challenge
  - 36k 질의, 883k 문서, 700개의 feature를 사용
  - Challenge 내 몇가지 특징들:

(1) 탑 랭커들은 decision tree를 주로 사용했다. (상위 5 팀이 모두 앙상블 트리를 사용함)

(2) 앙상블 기법(boosting, bagging and random forests)들이 주로 사용됐다.

Why before What?

- 2010 Yahoo L2R Challenge
  - 36k 질의, 883k 문서, 700개의 feature를 사용
  - Challenge 내 몇가지 특징들:

(1) 탑 랭커들은 decision tree를 주로 사용했다. (상위 5 팀이 모두 앙상블 트리를 사용함)

(2) 앙상블 기법(boosting, bagging and random forests)들이 주로 사용됐다.

(3) 탑 랭커들 사이의 점수차는 굉장히 미미했다. (비슷한 방식을 사용했으니 당연한 것일수도…)

Gradient Boosted Regression Tree

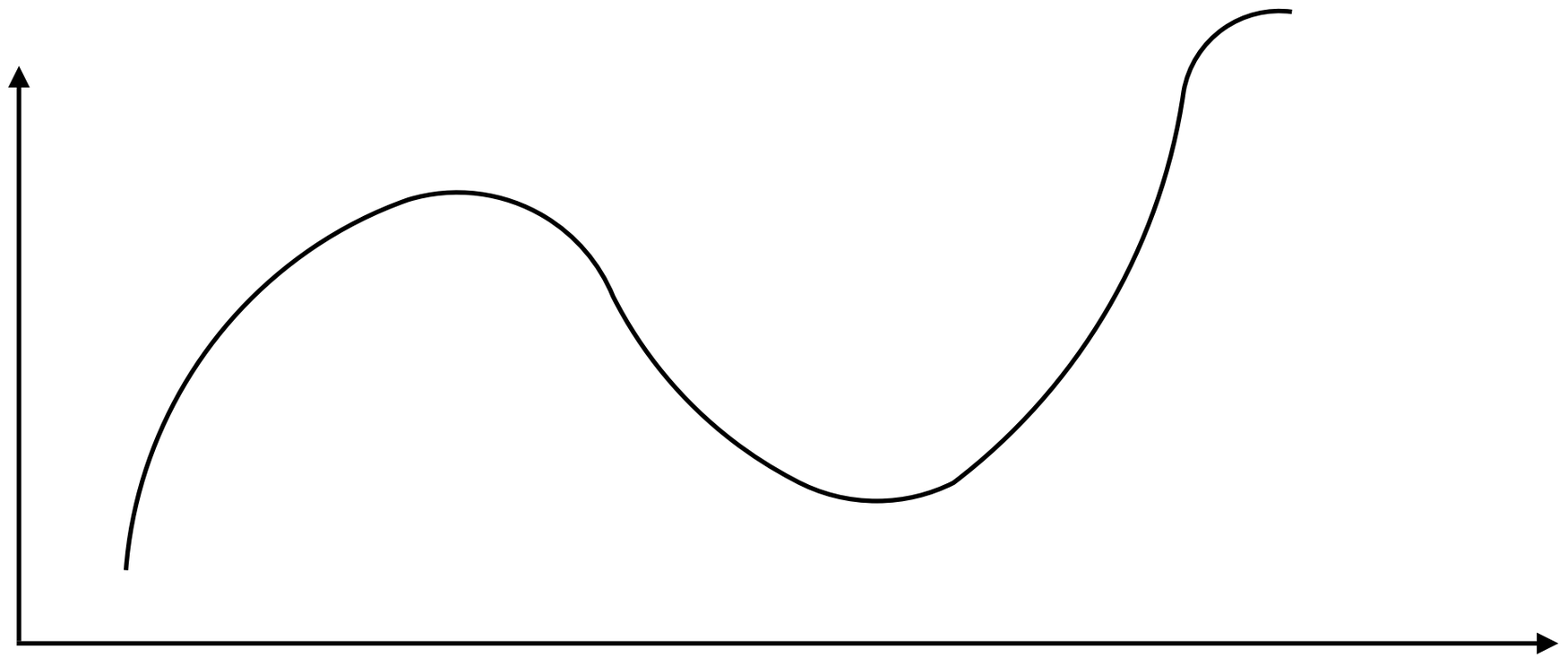
- Boosting 컨셉은?

- “The idea is to used the **weak learning method several times** to get a succession of hypotheses, each one refocused on the examples that the previous ones found difficult and misclassified. … Note, however, **it is not obvious at all how this can be done**” - Probably Approximately Correct: Nature’s Algorithms for Learning and Prospering in a Complex World, page 152, 2013

Gradient Boosted Regression Tree

- 더러운 수식에 들어가기 전에 그림으로 먼저 컨셉을 짚어보자

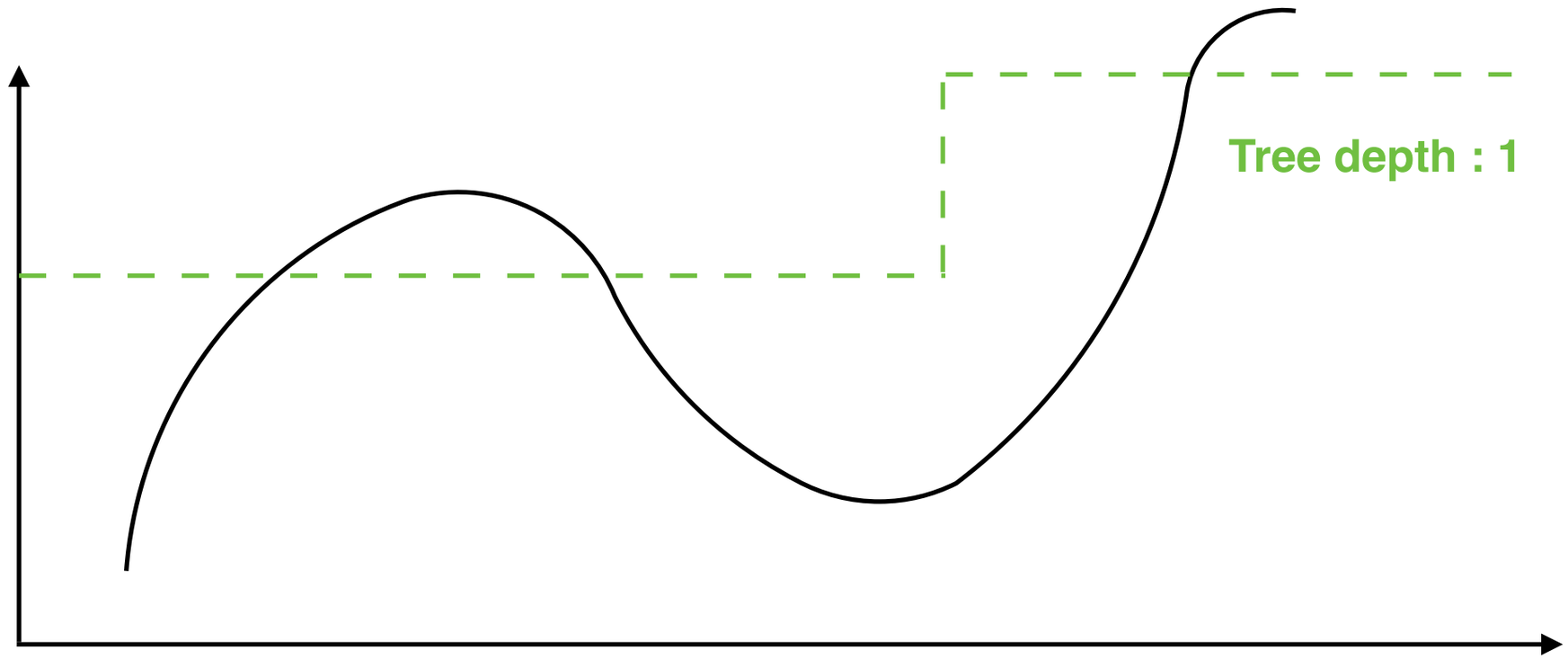
- Ex1) 1차원 feature를 가진 data를 트리로 regression 하기.



Gradient Boosted Regression Tree

- 더러운 수식에 들어가기 전에 그림으로 먼저 컨셉을 짚어보자

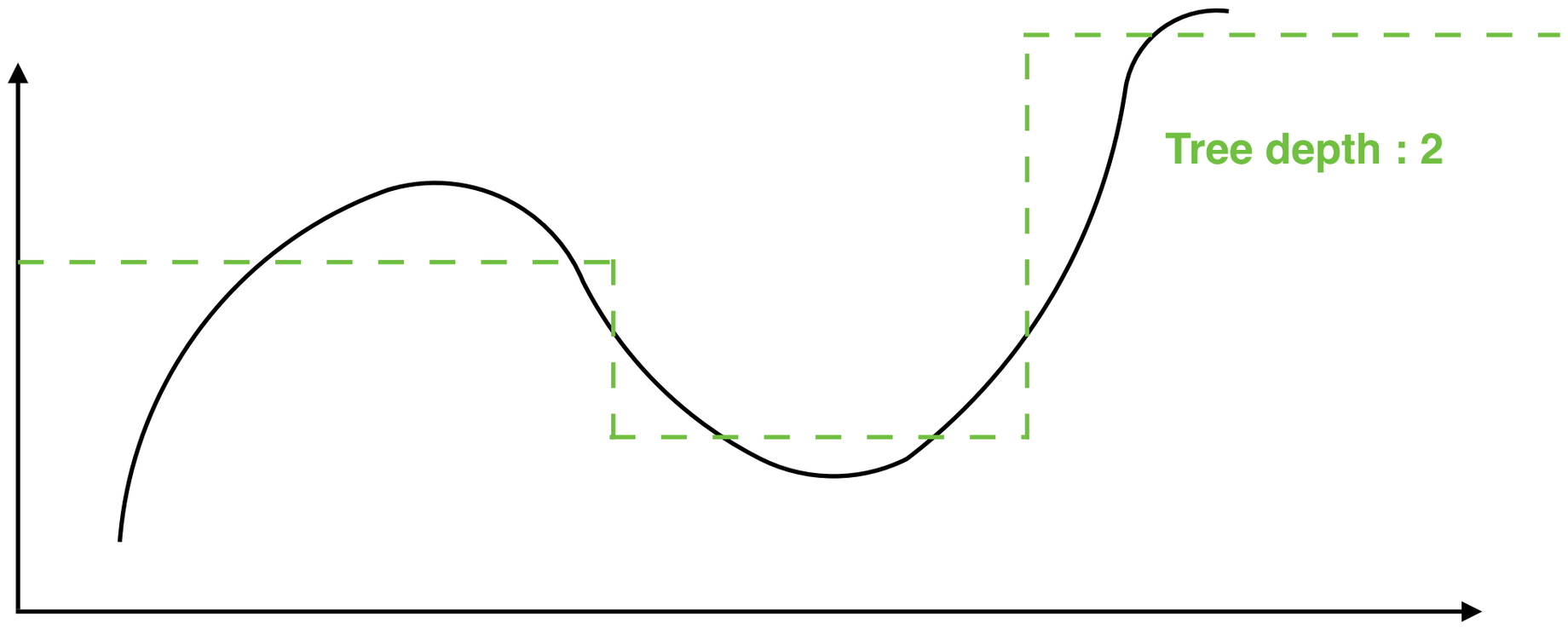
- Ex1) 1차원 feature를 가진 data를 트리로 regression 하기.



Gradient Boosted Regression Tree

- 더러운 수식에 들어가기 전에 그림으로 먼저 컨셉을 짚어보자

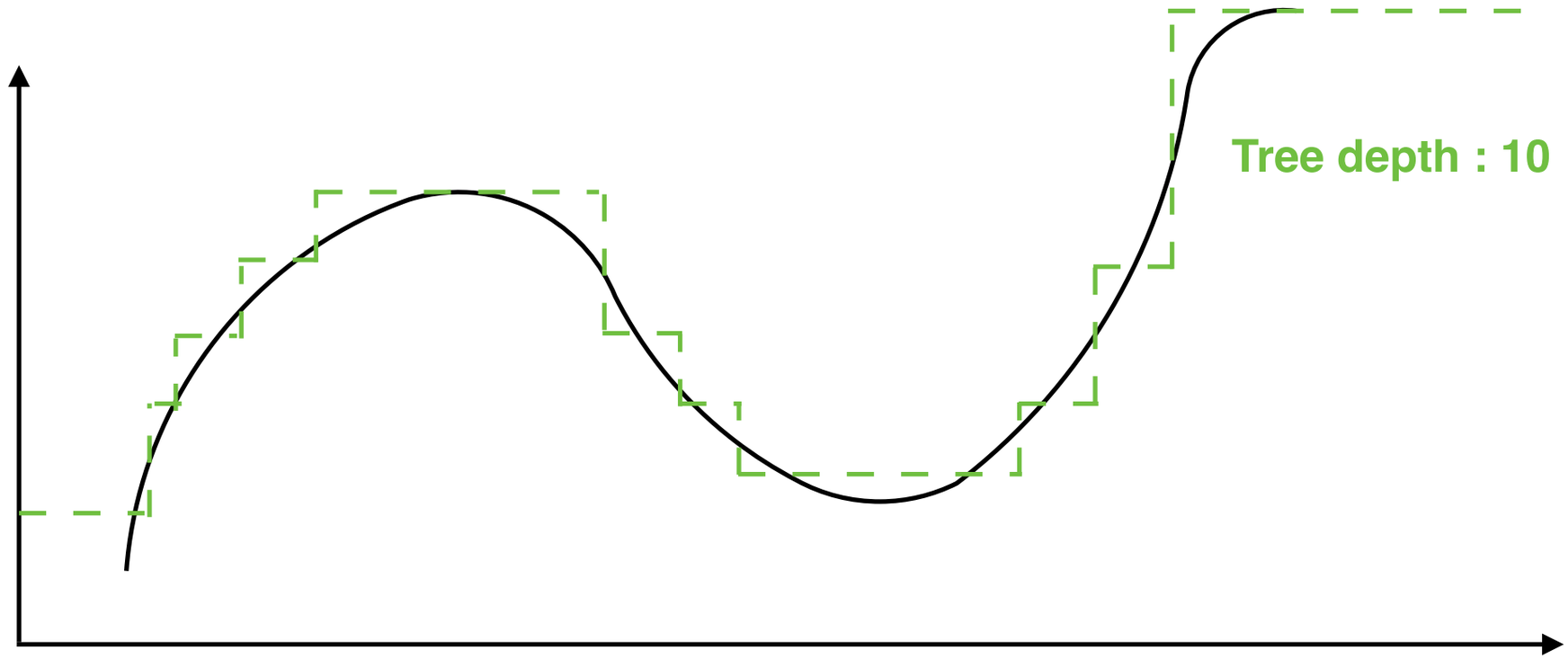
- Ex1) 1차원 feature를 가진 data를 트리로 regression 하기.



Gradient Boosted Regression Tree

- 더러운 수식에 들어가기 전에 그림으로 먼저 컨셉을 짚어보자

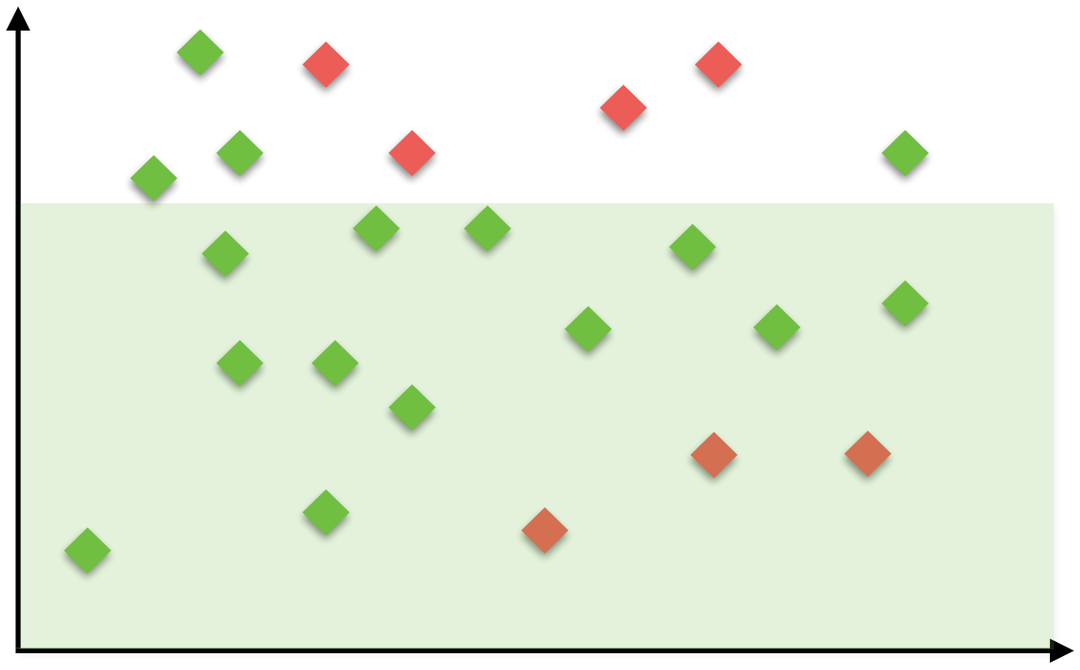
- Ex1) 1차원 feature를 가진 data를 트리로 regression 하기.



Gradient Boosted Regression Tree

- 더러운 수식에 들어가기 전에 그림으로 먼저 컨셉을 짚어보자

- Ex2) 2차원 feature를 가진 data를 앙상블 트리로 classification 하기.

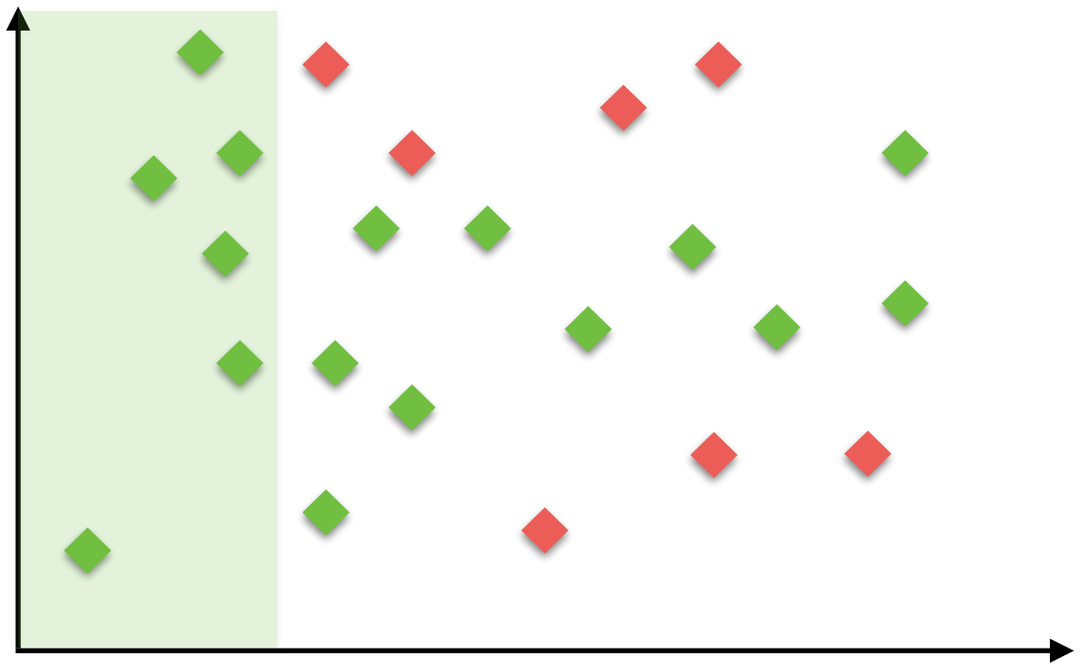


TREE1 of depth 1

Gradient Boosted Regression Tree

- 더러운 수식에 들어가기 전에 그림으로 먼저 컨셉을 짚어보자

- Ex2) 2차원 feature를 가진 data를 앙상블 트리로 classification 하기.

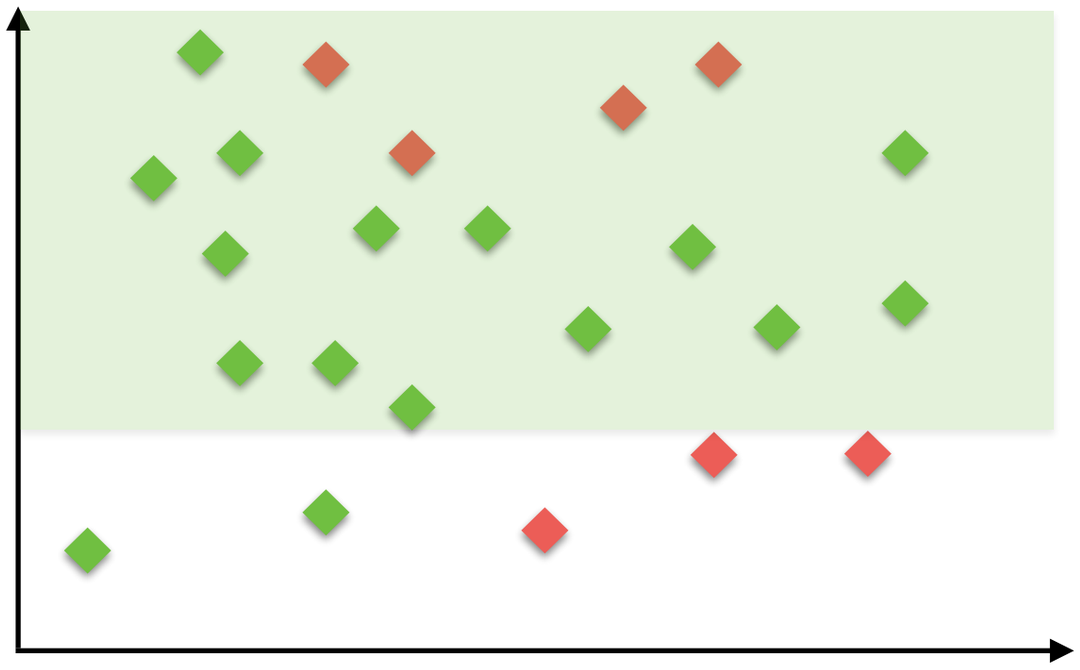


TREE2 of depth 1

Gradient Boosted Regression Tree

- 더러운 수식에 들어가기 전에 그림으로 먼저 컨셉을 짚어보자

- Ex2) 2차원 feature를 가진 data를 앙상블 트리로 classification 하기.

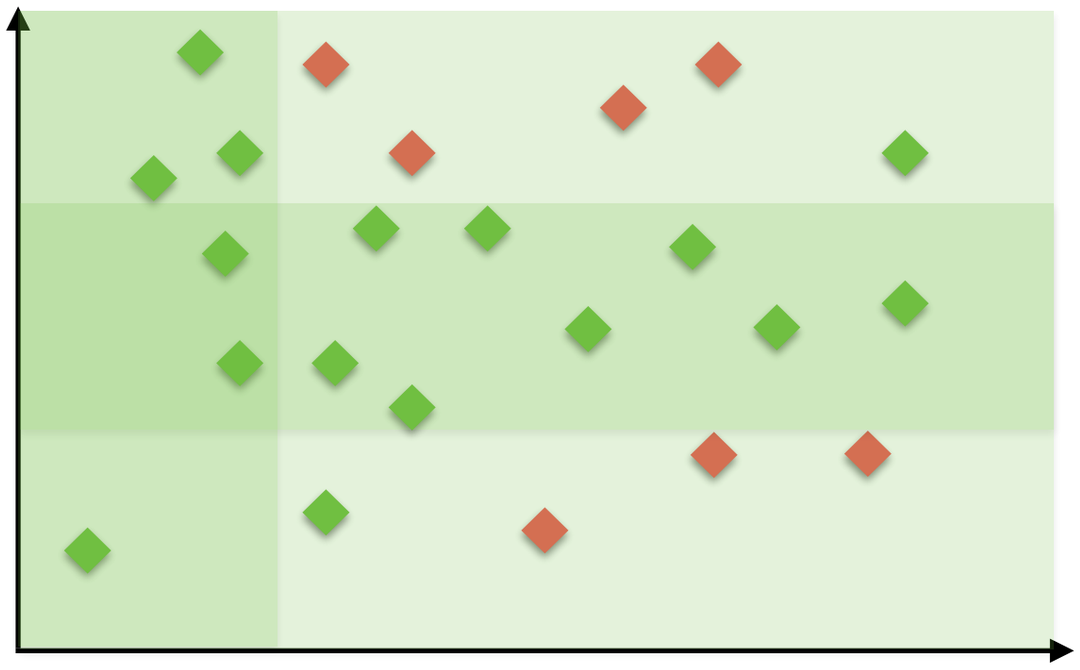


TREE3 of depth 1

Gradient Boosted Regression Tree

- 더러운 수식에 들어가기 전에 그림으로 먼저 컨셉을 짚어보자

- Ex2) 2차원 feature를 가진 data를 앙상블 트리로 classification 하기.



Ensemble of all TREEs

Gradient Boosted Regression Tree

- 그렇다면 짱짱이라고 하는 GBRT에 대해선 좀 자세히 보도록 하자.

Gradient Boosted Regression Tree

- 그렇다면 짱짱이라고 하는 GBRT에 대해선 좀 자세히 보도록 하자.

- 우선 ML의 공통적인 목표 함수를 review해 보면:

$$Obj(\Theta) = L(\Theta) + \Omega(\Theta)$$

training loss　　Regularisation

Gradient Boosted Regression Tree

- 그렇다면 짱짱이라고 하는 GBRT에 대해선 좀 자세히 보도록 하자.

- 우선 ML의 공통적인 목표 함수를 review해 보면:

$$Obj(\Theta) = \underbrace{L(\Theta)}_{\text{training loss}} + \underbrace{\Omega(\Theta)}_{\text{Regularisation}}$$

- training data의 loss : $L = \sum_{i=1}^{n} l(y_i, \hat{y}_i)$
  - square loss: $l(y_i, \hat{y}_i) = (y_i - \hat{y}_i)^2$
  - logistic loss: $l(y_i, \hat{y}_i) = y_i \ln(1 + e^{-\hat{y}_i}) + (1 - y_i) \ln(1 + e^{\hat{y}_i})$

Gradient Boosted Regression Tree

- 그렇다면 짱짱이라고 하는 GBRT에 대해선 좀 자세히 보도록 하자.

- 우선 ML의 공통적인 목표 함수를 review해 보면:

$$Obj(\Theta) = \underbrace{L(\Theta)}_{\text{training loss}} + \underbrace{\Omega(\Theta)}_{\text{Regularisation}}$$

- Regularisation : how complicated the model is
  - L2 norm: $\Omega(w) = \lambda||w||^2$
  - L1 norm: $\Omega(w) = \lambda||w||_1$

Gradient Boosted Regression Tree

- Loss 를 최적화하면 예측값이 더 잘나오고, regularization을 최적화하면 좀 더 simple한 모델이 된다. Simple한 모델은 unseen data 예측에 variance 가 작아서 좀 더 stable 하다.

Gradient Boosted Regression Tree

- Loss 를 최적화하면 예측값이 더 잘나오고, regularization을 최적화하면 좀 더 simple한 모델이 된다. Simple한 모델은 unseen data 예측에 variance 가 작아서 좀 더 stable 하다.

- Regression tree는 decision tree와 같은 decision rule 을 가지며, 각 leaf node당 score를 가지고 있다.

Gradient Boosted Regression Tree

- Loss 를 최적화하면 예측값이 더 잘나오고, regularization을 최적화하면 좀 더 simple한 모델이 된다. Simple한 모델은 unseen data 예측에 variance 가 작아서 좀 더 stable 하다.

- Regression tree는 decision tree와 같은 decision rule 을 가지며, 각 leaf node당 score를 가지고 있다.

- 이제 regression tree의 예제를 보며 배워보자!

Gradient Boosted Regression Tree

- Input:

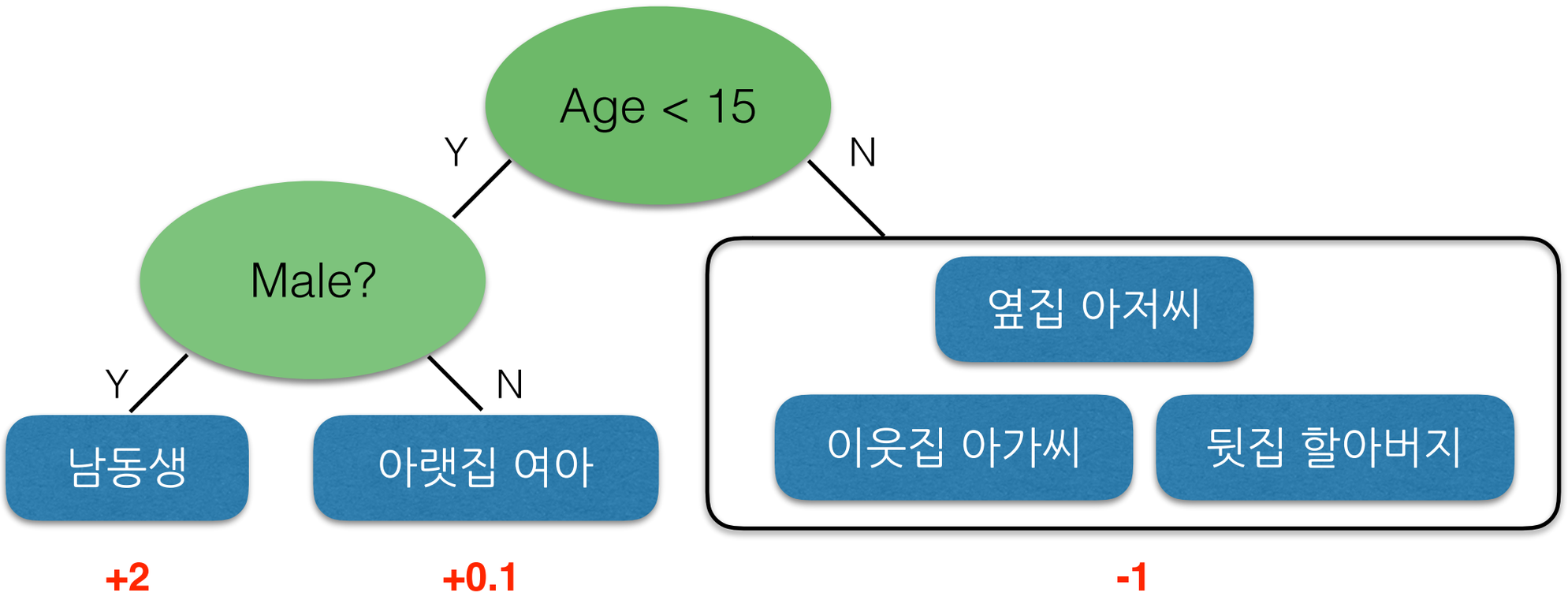
  - 나이, 직업, 성 등의 feature vector 로 나타냄.

Gradient Boosted Regression Tree

- Input:

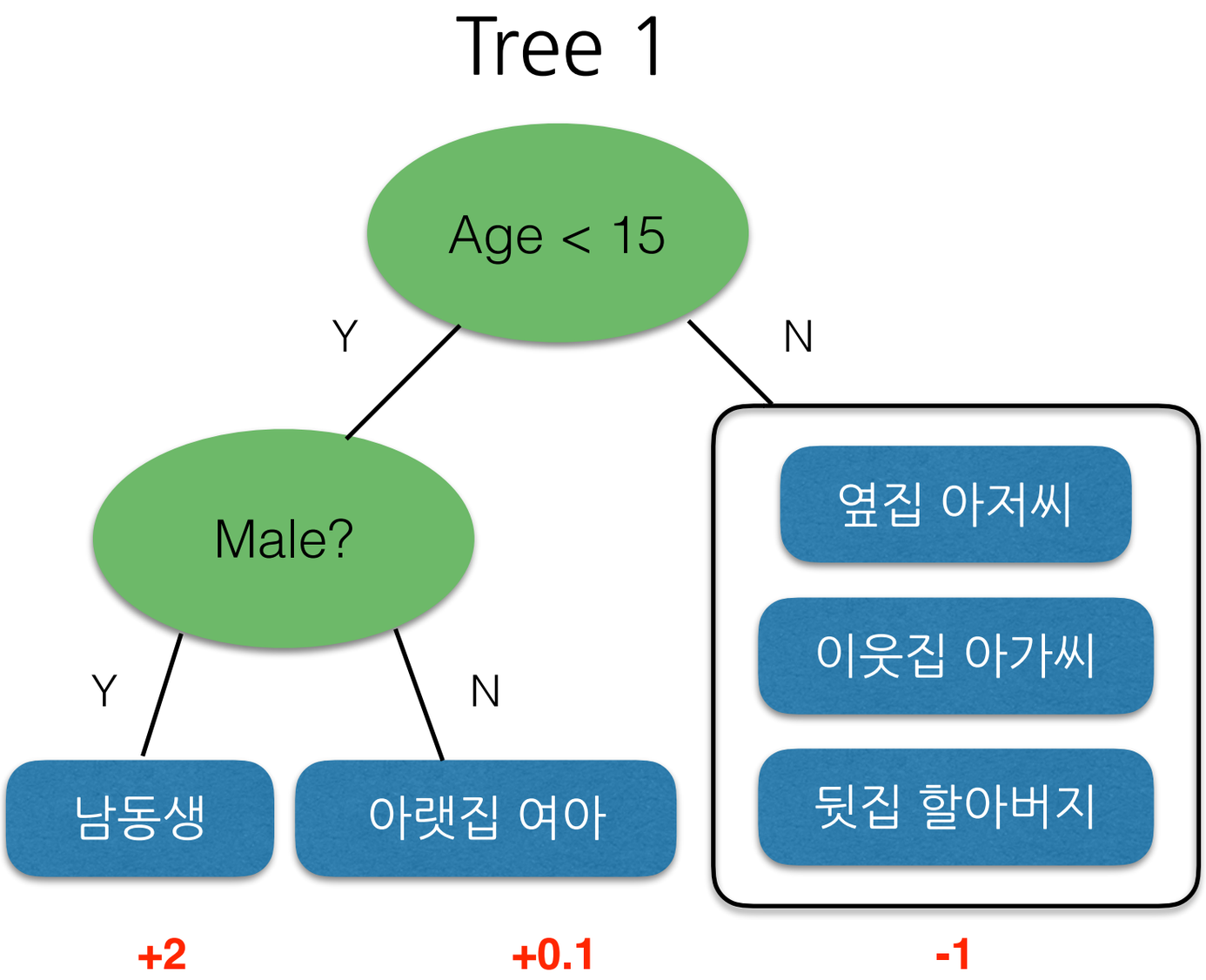
  - 나이, 직업, 성 등의 feature vector 로 나타냄.

- Problem: 이 사람은 컴퓨터 게임을 좋아하나요?



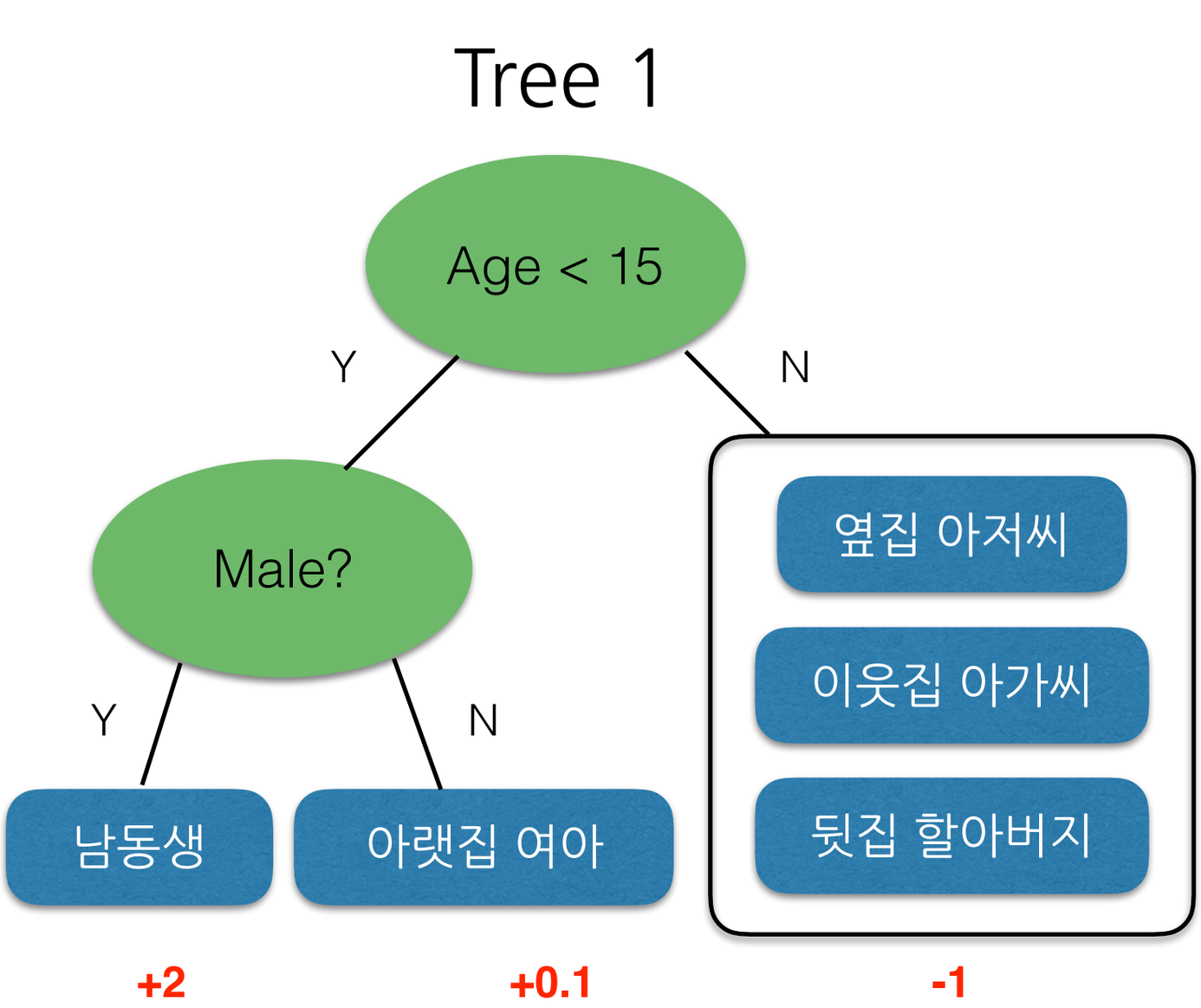
Gradient Boosted Regression Tree

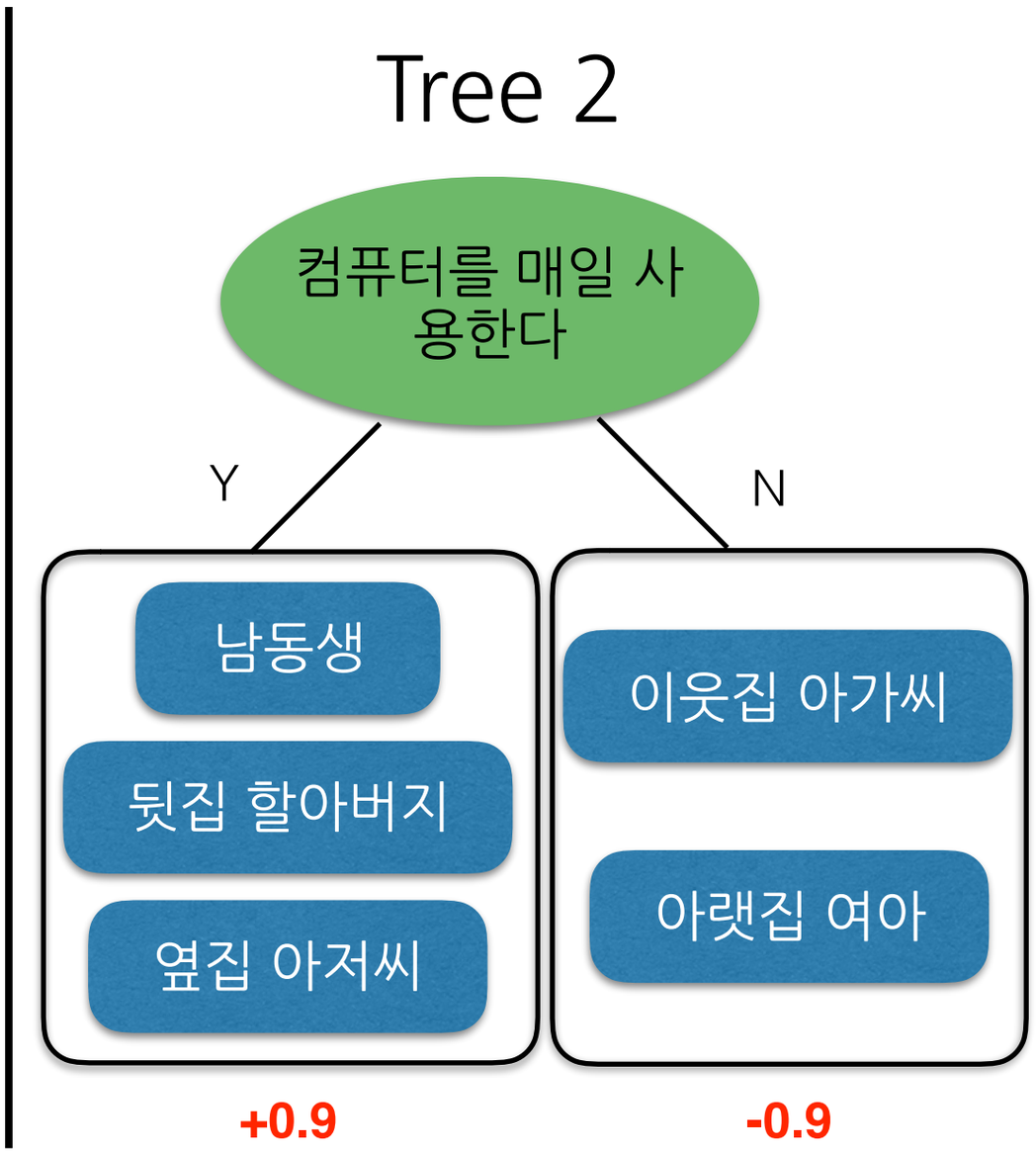
- Tree Ensemble



Gradient Boosted Regression Tree

- Tree Ensemble





Gradient Boosted Regression Tree

- 이런 tree ensemble을 통해서 fine tuning을 해 나가는 것이다!

Gradient Boosted Regression Tree

- 이런 tree ensemble을 통해서 fine tuning을 해 나가는 것이다!
- tree가 split되는 로직을 보면 input scaling을 요하지 않기 때문에 feature normalization이 필요없다!

Gradient Boosted Regression Tree

- 이런 tree ensemble을 통해서 fine tuning을 해 나가는 것이다!

- tree가 split되는 로직을 보면 input scaling을 요하지 않기 때문에 feature normalization이 필요없다!

- 트리 모델링을 시작해보자! (Let’s get dirty…)

Gradient Boosted Regression Tree

- Model : assuming we have K trees

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F$$

Gradient Boosted Regression Tree

- Model : assuming we have K trees

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in F$$

- Parameters
  - 각 tree의 구조와 leaf의 score를 포함한다
  - weight를 학습하는 게 아니라 function(tree)를 학습하는 것!

Gradient Boosted Regression Tree

- 어떻게 함수를 학습할 수 있지?

Gradient Boosted Regression Tree

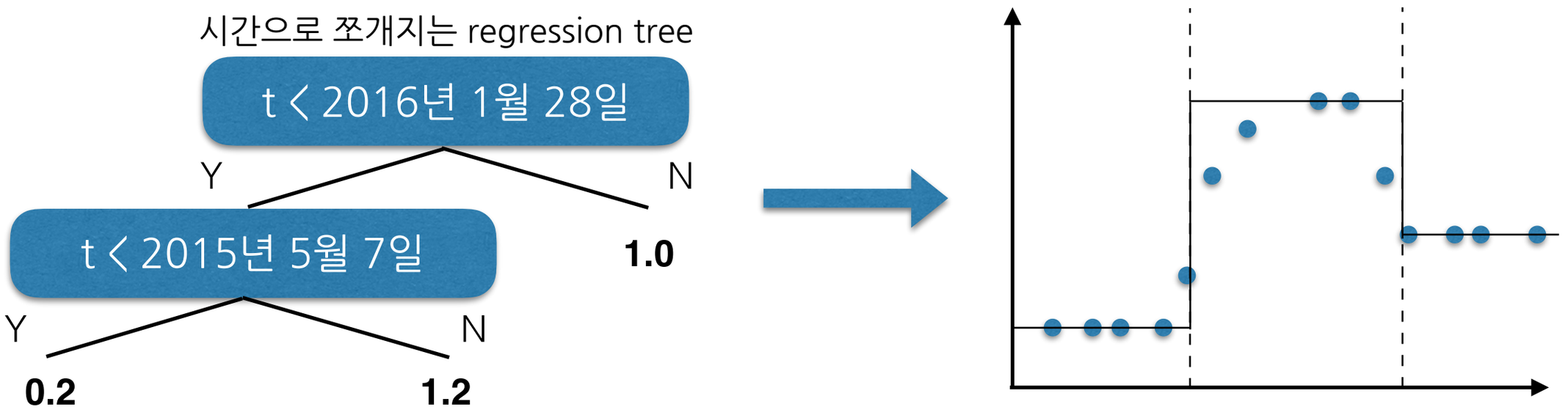
- 어떻게 함수를 학습할 수 있지?
- 이전과 똑같이 objective(loss,regularization)을 정해놓고 최적화 하면 된다!

Gradient Boosted Regression Tree

- 어떻게 함수를 학습할 수 있지?
- 이전과 똑같이 objective(loss,regularization)을 정해놓고 최적화 하면 된다!
- 예를 들어볼까:
  - 하나의 input t(time)을 가지는 regression tree
    - 내가 불특정 시간 t에서 사랑노래를 듣고 싶어하는지 예측을 해보고 싶다.

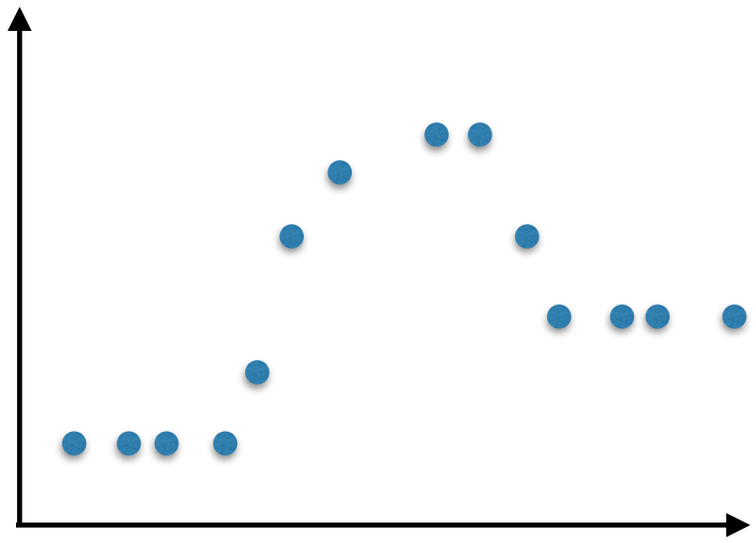
Gradient Boosted Regression Tree

- 어떻게 함수를 학습할 수 있지?
- 이전과 똑같이 objective(loss,regularization)을 정해놓고 최적화 하면 된다!
- 예를 들어볼까:
  - 하나의 input t(time)을 가지는 regression tree
    - 내가 불특정 시간 t에서 사랑노래를 듣고 싶어하는지 예측을 해보고 싶다.

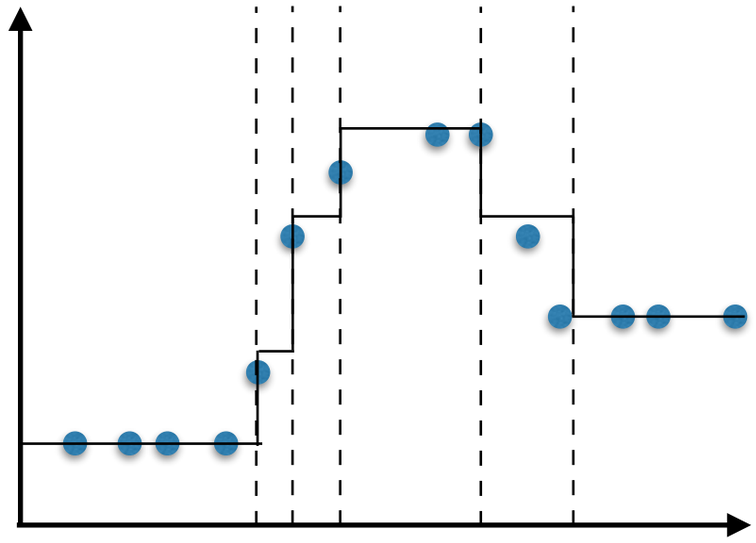


Gradient Boosted Regression Tree

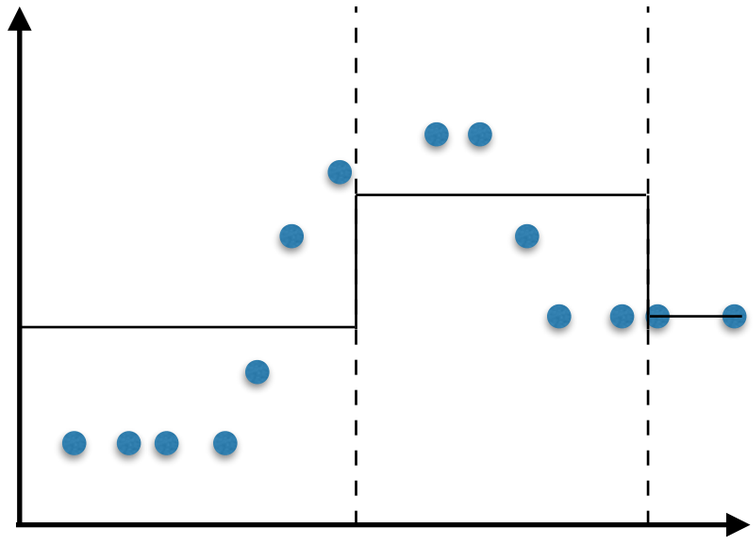
- Step function 학습 예



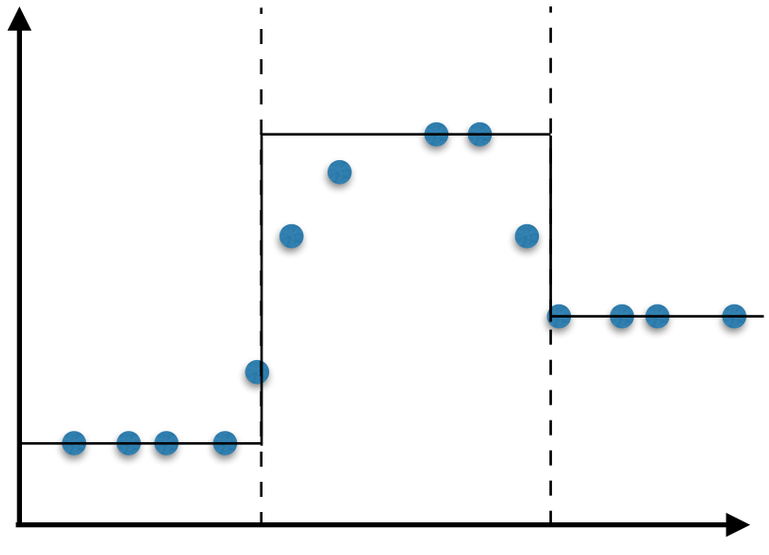
User data



Too many splits : Regularisation loss too high



Wrong split points : Loss too high



Good balance of L and R

Gradient Boosted Regression Tree

- 다시 Object 로 돌아가서:

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant
\end{aligned}
$$

Goal: find f_t to minimize this

Gradient Boosted Regression Tree

- 다시 Object 로 돌아가서:

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{i=1}^{t} \Omega(f_i) \\
&= \sum_{i=1}^{n} l\left(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right) + \Omega(f_t) + constant
\end{aligned}
$$

Goal: find f_t to minimize this

- Square loss 를 사용한다면:

$$
\begin{aligned}
Obj^{(t)} &= \sum_{i=1}^{n} \left(y_i - \left(\hat{y}_i^{(t-1)} + f_t(x_i)\right)\right)^2 + \Omega(f_t) + const \\
&= \sum_{i=1}^{n} \left[2(\hat{y}_i^{(t-1)} - y_i) f_t(x_i) + f_t(x_i)^2\right] + \Omega(f_t) + const
\end{aligned}
$$

Gradient Boosted Regression Tree

- Square loss 가 아니라면 상당이 복잡한데…

Gradient Boosted Regression Tree

- Square loss 가 아니라면 상당이 복잡한데…
- Taylor expansion을 써보자:

$$\text{Taylor Expansion}: \quad f(x+\Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$$

$$\text{Define}: \quad g_i = \frac{\partial \quad l(y_i, \hat{y}^{(t-1)})}{\partial \quad \hat{y}^{(t-1)}} \text{ , and } h_i = \frac{\partial^2 \quad l(y_i, \hat{y}^{(t-1)})}{\partial^2 \quad \hat{y}^{(t-1)}}$$

Gradient Boosted Regression Tree

- Square loss 가 아니라면 상당이 복잡한데…
- Taylor expansion을 써보자:

$$\text{Taylor Expansion}: \quad f(x+\Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2$$

$$\text{Define}: \quad g_i = \frac{\partial \quad l(y_i, \hat{y}^{(t-1)})}{\partial \quad \hat{y}^{(t-1)}} \text{ , and } h_i = \frac{\partial^2 \quad l(y_i, \hat{y}^{(t-1)})}{\partial^2 \quad \hat{y}^{(t-1)}}$$

$$Obj^{(t)} \approx \sum_{i=1}^{n} \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) + constant$$

$$Obj^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \quad \text{with constants removed}$$

Gradient Boosted Regression Tree

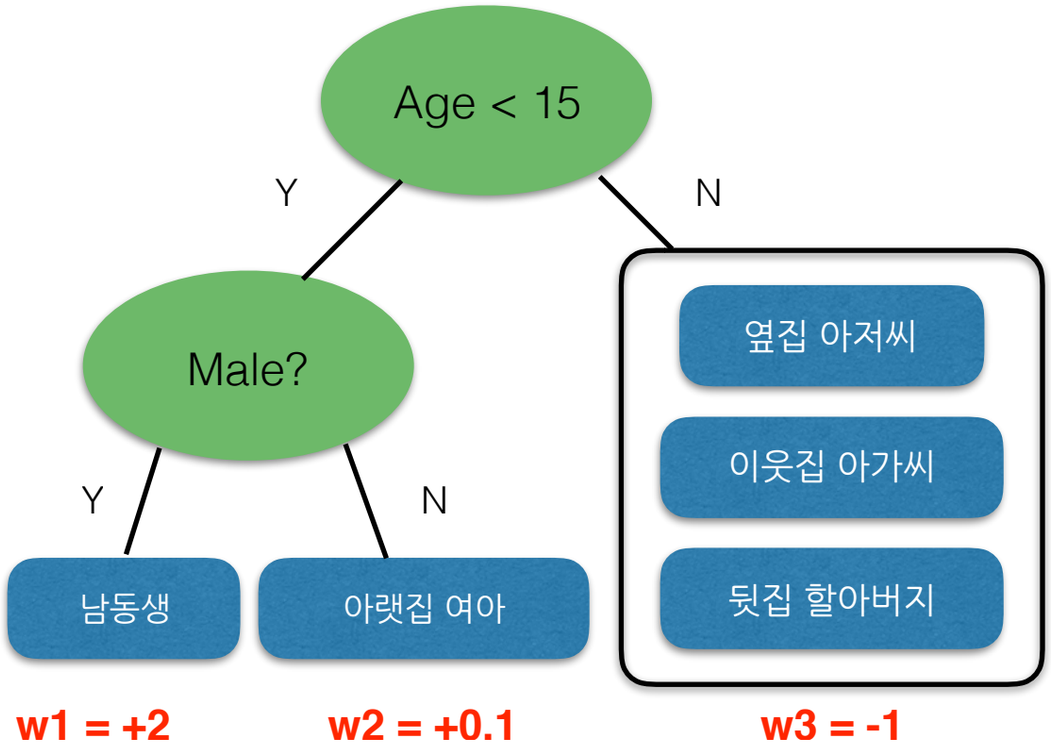
- Tree의 정의를 좀 더 디테일하게 해보자 :
  - Tree를 leaf score의 벡터와 각 instance를 leaf로 매핑해주는 함수(q)로 나타낸다:

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

Gradient Boosted Regression Tree

- Tree의 정의를 좀 더 디테일하게 해보자 :
  - Tree를 leaf score의 벡터와 각 instance를 leaf로 매핑해주는 함수(q)로 나타낸다:

$$f_x(x) = w_{q(x)}, \quad w \in \mathbf{R}^T, q : \mathbf{R}^d \rightarrow 1, 2, ..., T$$



q(남동생) = 1

q(이웃집 아가씨) = 3

Gradient Boosted Regression Tree

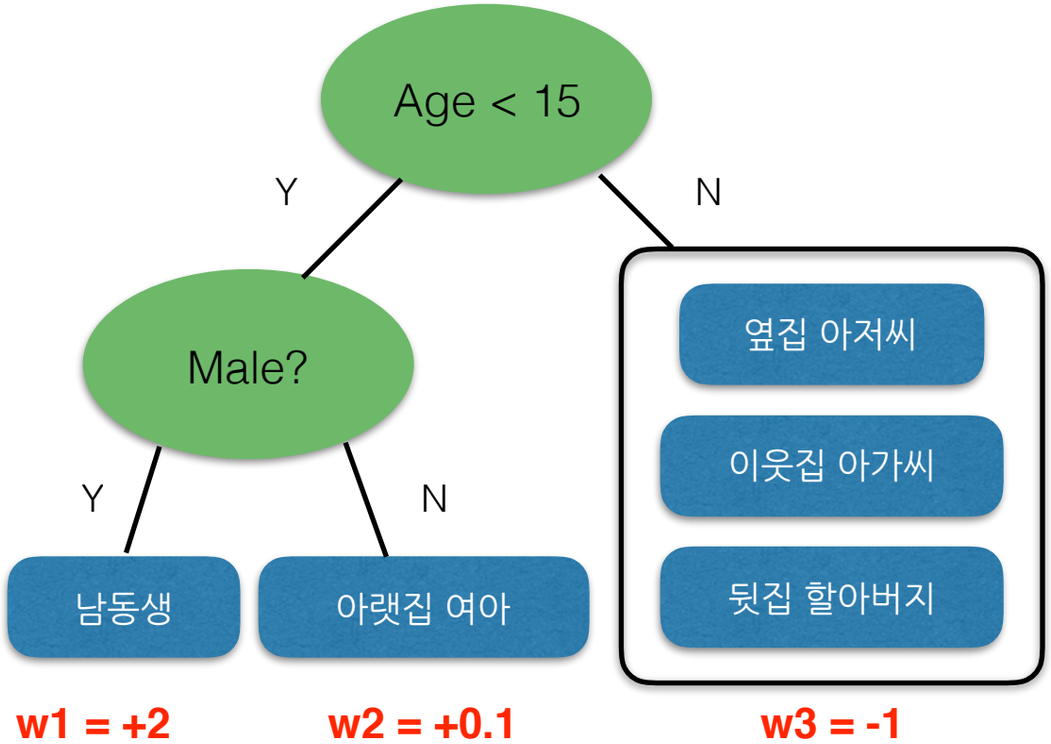
- Tree의 complexity도 정의해 보고:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Gradient Boosted Regression Tree

- Tree의 complexity도 정의해 보고:

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$



$$\Omega = \gamma \cdot 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

Gradient Boosted Regression Tree

- 마지막으로 leaf j에 있는 instance set을 정의:

$$I_j = \{i|q(x_i) = j\}$$

Gradient Boosted Regression Tree

- 마지막으로 leaf j에 있는 instance set을 정의:

$$I_j = \{i|q(x_i) = j\}$$

- Objective 함수를 다시 적어보면:

$$Obj^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t)$$

Gradient Boosted Regression Tree

- 마지막으로 leaf j에 있는 instance set을 정의:

$$I_j = \{i|q(x_i) = j\}$$

- Objective 함수를 다시 적어보면:

$$\begin{aligned} Obj^{(t)} &\approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \\ &= \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2 \end{aligned}$$

Gradient Boosted Regression Tree

- 마지막으로 leaf j에 있는 instance set을 정의:

$$I_j = \{i|q(x_i) = j\}$$

- Objective 함수를 다시 적어보면:

$$\begin{aligned} Obj^{(t)} &\approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \\ &= \sum_{i=1}^{n} \left[g_i w_{q(x_i)} + \frac{1}{2} h_i w_{q(x_i)}^2 \right] + \gamma T + \lambda \frac{1}{2} \sum_{j=1}^{T} w_j^2 \\ &= \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T \end{aligned}$$

Gradient Boosted Regression Tree

$$= \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T$$

Gradient Boosted Regression Tree

$$= \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T$$

$$= \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T$$

Gradient Boosted Regression Tree

$$= \sum_{j=1}^{T} \left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2} (\sum_{i \in I_j} h_i + \lambda) w_j^2 \right] + \gamma T$$

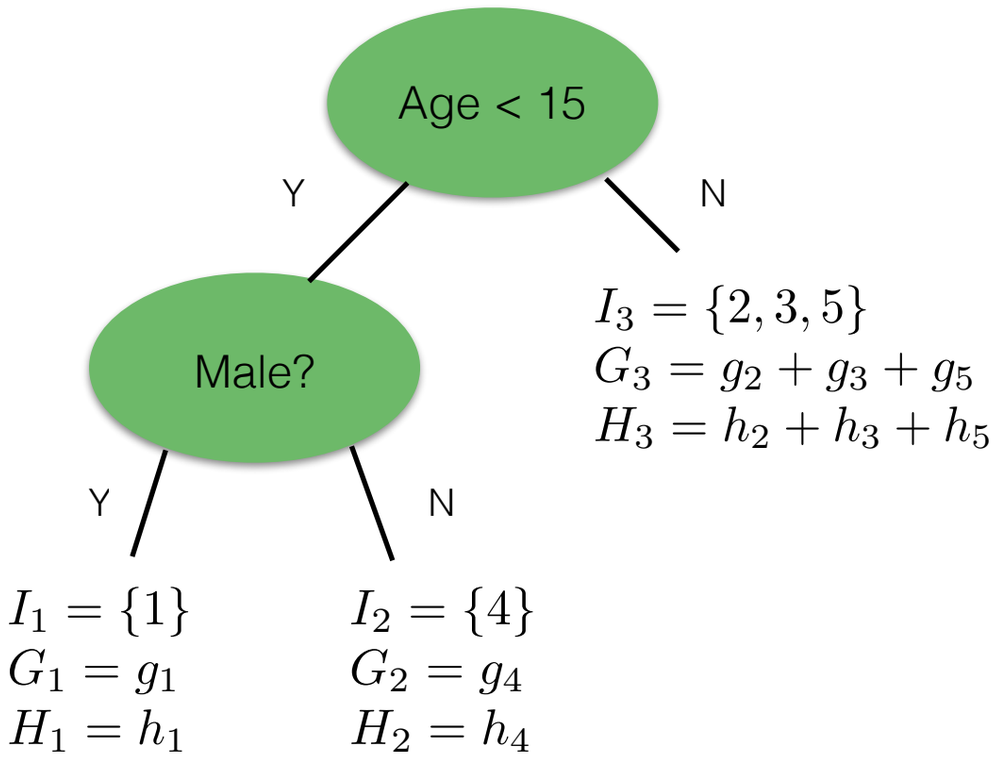
$$= \sum_{j=1}^{T} \left[G_j w_j + \frac{1}{2} (H_j + \lambda) w_j^2 \right] + \gamma T$$

- w에 관한 2차 방정식이므로 optimal weight w*:

$$w_j^* = -\frac{G_j}{H_j + \lambda} \quad Obj = -\frac{1}{2} \sum_{j=1}^{T} \frac{G_j^2}{H_j + \lambda} + \gamma T$$

Gradient Boosted Regression Tree

Index	Instance	Gradient Statistics
1	남동생	g1, h1
2	이웃집 아가씨	g2, h2
3	뒷집 할아버지	g3, h3
4	아랫집 여아	g4, h4
5	옆집 아저씨	g5, h5



$$Obj = -\sum_j \frac{G_j^2}{H_j + \lambda} + 3\gamma$$

Gradient Boosted Regression Tree

- 알고리즘:

Gradient Boosted Regression Tree

- 알고리즘:

  (1) 가능한 tree structure q들을 listing한다

Gradient Boosted Regression Tree

- 알고리즘:

  (1) 가능한 tree structure q들을 listing한다

  (2) 각 q의 Obj 값을 구한다.

Gradient Boosted Regression Tree

- 알고리즘:

  (1) 가능한 tree structure q들을 listing한다

  (2) 각 q의 Obj 값을 구한다.

  (3) 가장 Obj값이 낮은 tree structure를 찾고 optimal weight w*를 사용한다.

Gradient Boosted Regression Tree

- 알고리즘:

  (1) 가능한 tree structure q들을 listing한다

  (2) 각 q의 Obj 값을 구한다.

  (3) 가장 Obj값이 낮은 tree structure를 찾고 optimal weight w*를 사용한다.

- 하지만 엄청나게 많은 tree structure q들이 있겠지…

Gradient Boosted Regression Tree

- 그래서 practical하게: Greedy learning of the Tree:

Gradient Boosted Regression Tree

- 그래서 practical하게: Greedy learning of the Tree:

  (1) depth 0인 트리로 시작

Gradient Boosted Regression Tree

- 그래서 practical하게: Greedy learning of the Tree:

  (1) depth 0인 트리로 시작

  (2) 각 leaf 노드에서 split을 했을때의 gain을 연산:

Gradient Boosted Regression Tree

- 그래서 practical하게: Greedy learning of the Tree:

  (1) depth 0인 트리로 시작

  (2) 각 leaf 노드에서 split을 했을때의 gain을 연산:

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

Left child score　　Right child score　　split을 하지 않을 때 score

Gradient Boosted Regression Tree

- 그래서 practical하게: Greedy learning of the Tree:

  (1) depth 0인 트리로 시작

  (2) 각 leaf 노드에서 split을 했을때의 gain을 연산:

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

Left child score　　Right child score　　split을 하지 않을 때 score

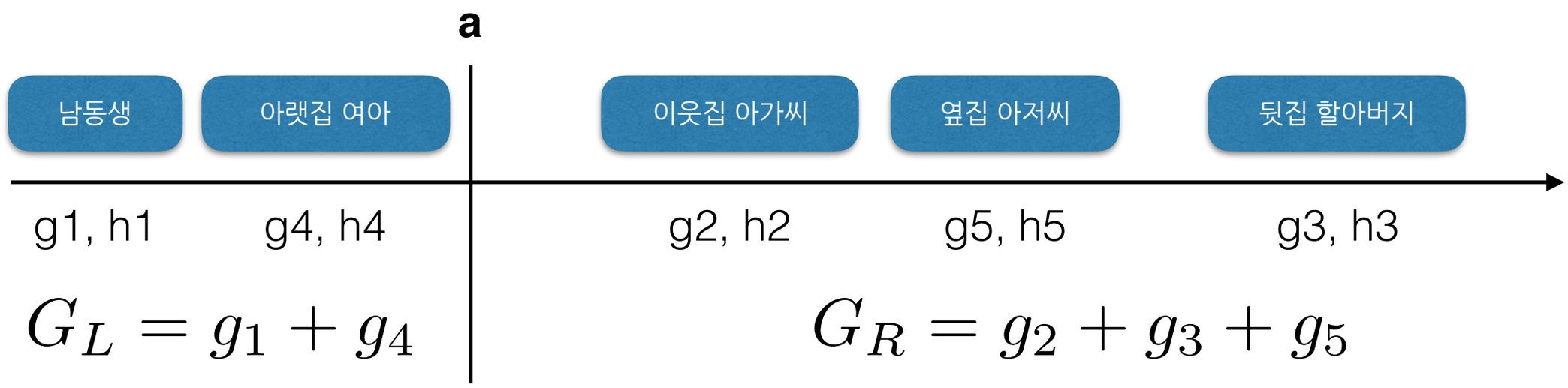
- 어떻게 best split을 찾을까?

Gradient Boosted Regression Tree

- $x_j < a$로 split을 했을 때의 gain은 얼마지? x_j가 나이라고 가정해보자.

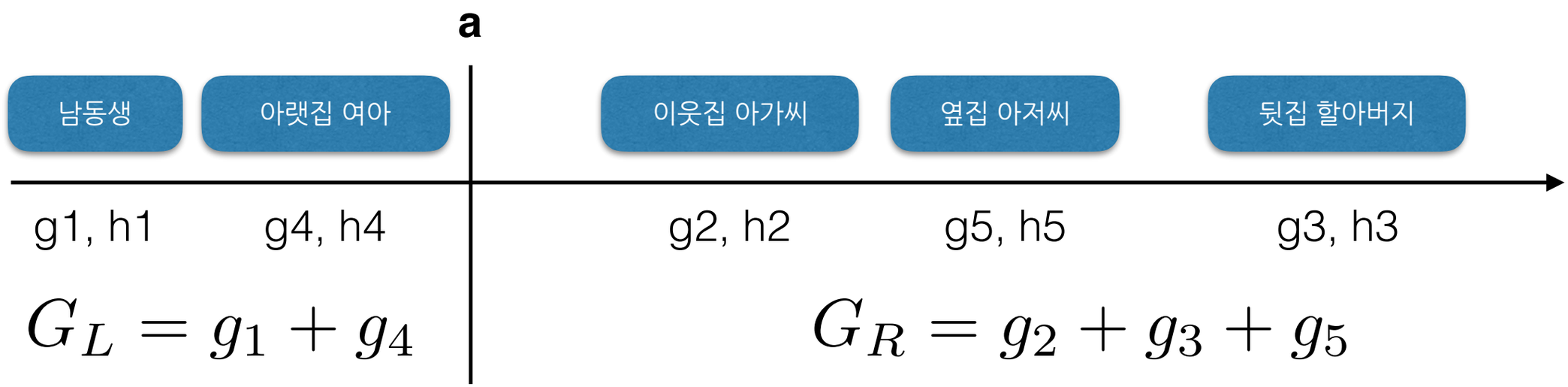
Gradient Boosted Regression Tree

- $x_j < a$로 split을 했을 때의 gain은 얼마지? x_j가 나이라고 가정해보자.



Gradient Boosted Regression Tree

- $x_j < a$로 split을 했을 때의 gain은 얼마지? x_j가 나이라고 가정해보자.



- 이 값들을 사용해서 gain을 구하고:

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

- Linear 스캔을 통해 best split을 찾으면 된다!

Gradient Boosted Regression Tree

- 각 leaf node에서 모든 feature을 iterate 하는데

Gradient Boosted Regression Tree

- 각 leaf node에서 모든 feature을 iterate 하는데
  - Leaf node의 instance들을 feature value로 정렬

Gradient Boosted Regression Tree

- 각 leaf node에서 모든 feature을 iterate 하는데
  - Leaf node의 instance들을 feature value로 정렬
  - Linear 스캐닝을 통해서 best split 찾고

Gradient Boosted Regression Tree

- 각 leaf node에서 모든 feature을 iterate 하는데
  - Leaf node의 instance들을 feature value로 정렬
  - Linear 스캐닝을 통해서 best split 찾고
  - 모든 feature들 중에서 가장 좋은 split으로 한다

Gradient Boosted Regression Tree

- 각 leaf node에서 모든 feature을 iterate 하는데
  - Leaf node의 instance들을 feature value로 정렬
  - Linear 스캐닝을 통해서 best split 찾고
  - 모든 feature들 중에서 가장 좋은 split으로 한다
- Depth가 K인 tree로 만드는 데 걸리는 Time Complexity는 $O(n \cdot d \cdot K \cdot log(n))$

Gradient Boosted Regression Tree

- Gain 이 음수일 경우도 있다는 사실!

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

Gradient Boosted Regression Tree

- Gain 이 음수일 경우도 있다는 사실!

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

  - training loss reduction < regularisation 일 때.

  - Predictiveness 와 simplicity의 tradeoff.

Gradient Boosted Regression Tree

- Gain 이 음수일 경우도 있다는 사실!

$$Gain = \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

- training loss reduction < regularisation 일 때.

- Predictiveness 와 simplicity의 tradeoff.

- Best split이 음수일 때 stop하는 pre-stopping과 tree 를 먼저 max depth로 키운 후에 negative gain split 을 pruning 하는 post pruning 기법이 있다.

Gradient Boosted Regression Tree

- GBRT 총 정리:

Gradient Boosted Regression Tree

- GBRT 총 정리:
  - 각 iteration에서 나무 하나를 추가한다

Gradient Boosted Regression Tree

- GBRT 총 정리:
  - 각 iteration에서 나무 하나를 추가한다
  - iteration은 grad stats를 계산하는 것으로 시작:

$$g_i = \frac{\partial \quad l(y_i, \hat{y}^{(t-1)})}{\partial \quad \hat{y}^{(t-1)}}, h_i = \frac{\partial^2 \quad l(y_i, \hat{y}^{(t-1)})}{\partial^2 \quad \hat{y}^{(t-1)}}$$

Gradient Boosted Regression Tree

- GBRT 총 정리:
  - 각 iteration에서 나무 하나를 추가한다
  - iteration은 grad stats를 계산하는 것으로 시작:

$$g_i = \frac{\partial \quad l(y_i, \hat{y}^{(t-1)})}{\partial \quad \hat{y}^{(t-1)}}, h_i = \frac{\partial^2 \quad l(y_i, \hat{y}^{(t-1)})}{\partial^2 \quad \hat{y}^{(t-1)}}$$

  - stats 사용해서 greedy하게 tree $f_t(x)$를 키운다

$$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T$$

Gradient Boosted Regression Tree

- GBRT 총 정리:
  - 각 iteration에서 나무 하나를 추가한다
  - iteration은 grad stats를 계산하는 것으로 시작:

$$g_i = \frac{\partial \quad l(y_i, \hat{y}^{(t-1)})}{\partial \quad \hat{y}^{(t-1)}}, h_i = \frac{\partial^2 \quad l(y_i, \hat{y}^{(t-1)})}{\partial^2 \quad \hat{y}^{(t-1)}}$$

  - stats 사용해서 greedy하게 tree $f_t(x)$를 키운다

$$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T$$

  - 마지막으로 tree를 지금까지 모델에다가 추가해 준다:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \epsilon f_t(x_i)$$

Gradient Boosted Regression Tree

- GBRT 총 정리:
  - 각 iteration에서 나무 하나를 추가한다
  - iteration은 grad stats를 계산하는 것으로 시작:

$$g_i = \frac{\partial \quad l(y_i, \hat{y}^{(t-1)})}{\partial \quad \hat{y}^{(t-1)}}, h_i = \frac{\partial^2 \quad l(y_i, \hat{y}^{(t-1)})}{\partial^2 \quad \hat{y}^{(t-1)}}$$

  - stats 사용해서 greedy하게 tree $f_t(x)$를 키운다

$$Obj = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j + \lambda} + \gamma T$$

  - 마지막으로 tree를 지금까지 모델에다가 추가해 준다:

$$\hat{y}_i^{(t)} = \hat{y}_i^{(t-1)} + \epsilon f_t(x_i)$$

shrinkage (~0.1) : helps prevent overfitting

References

- “Introduction to Boosted Trees” : https://homes.cs.washington.edu/~tqchen/pdf/BoostedTree.pdf

- “A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning” : http://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/

- “Yahoo! Learning to Rank Challenge Overview”

Questions?