Ensemble models

Hyunjoong Kim

soy.lovit@gmail.com

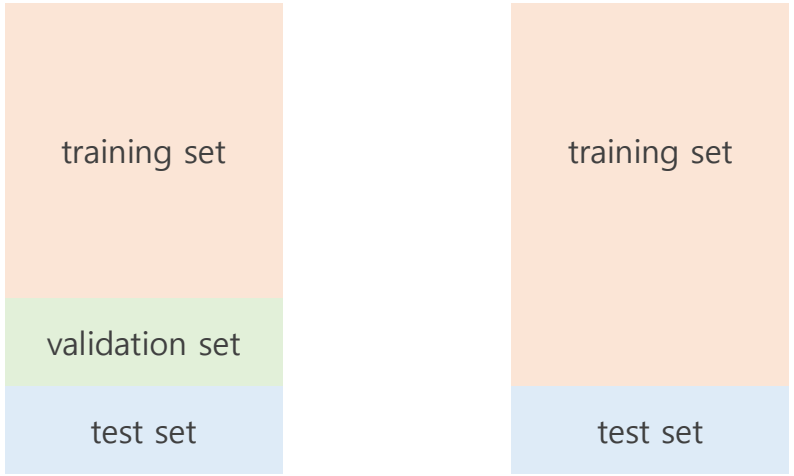
github.com/lovit

Data split

- 머신러닝 알고리즘의 성능은 학습 성능과 일반화 성능을 구분합니다.
  - SVM 은 동일한 학습 성능 (loss) 이라면 일반화 성능이 더 높은 모델을 학습하기 위하여 margin 을 최대화하는 경계면을 학습하였습니다.
  - Linear / logistic regression 은 학습데이터에 과적합되는 문제를 해결하기 위하여 L1, L2 regularization 을 이용하였습니다.
  - 학습 성능은 training data 에 대한 모델의 성능이며, 일반화 성능은 학습된 모델이 배포된 환경에서의 기대성능입니다.

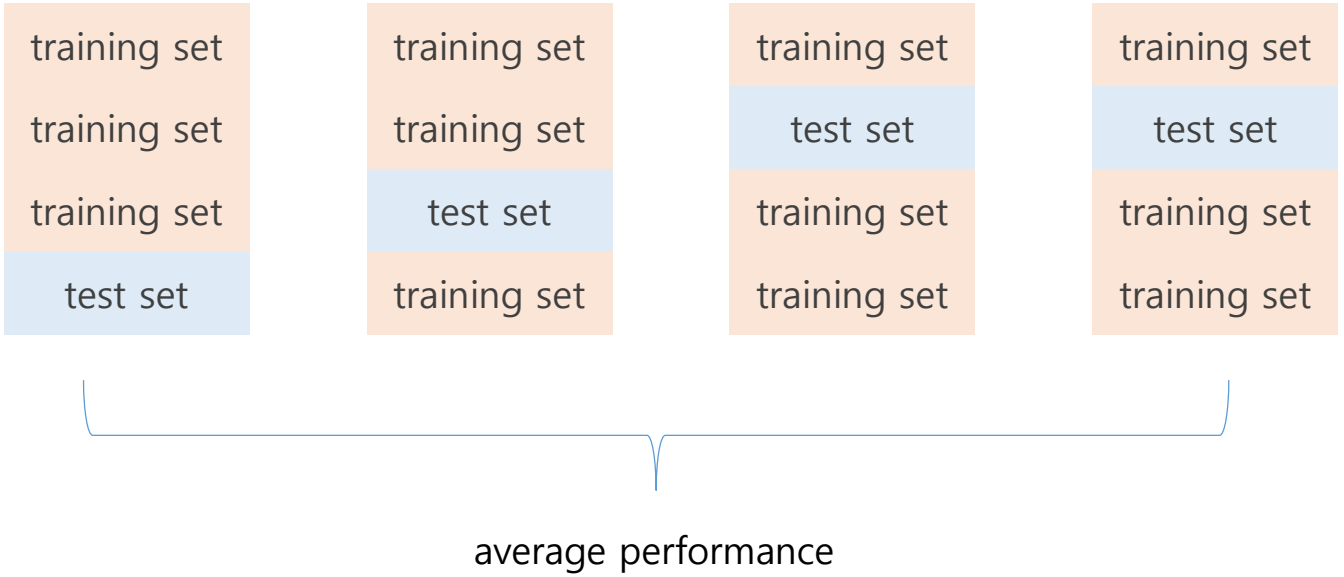
Data split

- 일반화 성능을 측정하기 위하여 데이터를 분할합니다.
  - Training set : 모델의 학습에 이용합니다.
  - (Validation set) : 하이퍼 패러매터의 조절을 위해 이용됩니다.
    하이퍼 패러매터의 탐색과정이 생략된다면 validation 은 하지 않아도 됩니다.
  - Test set : 일반화 성능을 측정합니다.



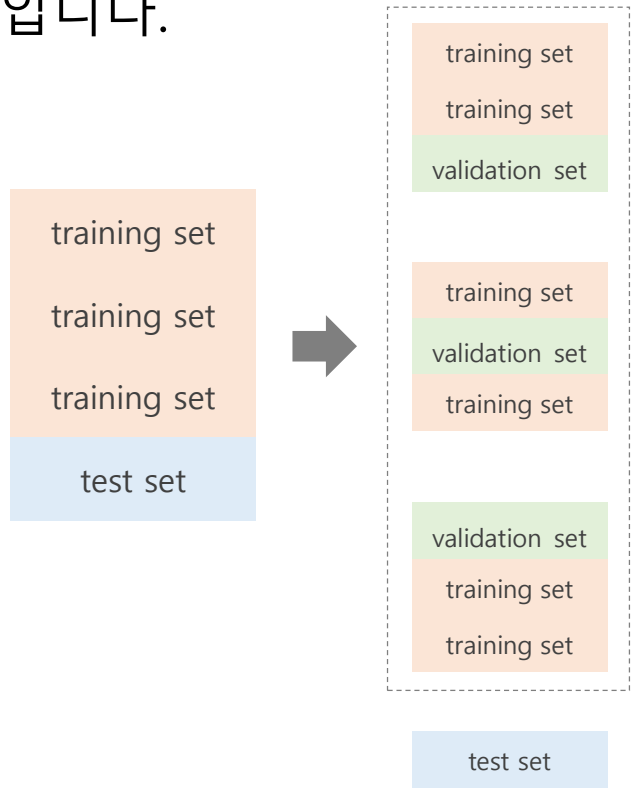
Data split

- 데이터셋의 크기가 작을 때에는 validation / test set 으로 데이터를 할당하기 어렵습니다.
- Cross validation 은 이 때 이용하는 방법으로, 교집합이 없는 test set 을 여러 번 만든 뒤, 평균 성능을 모델의 성능으로 이용합니다.



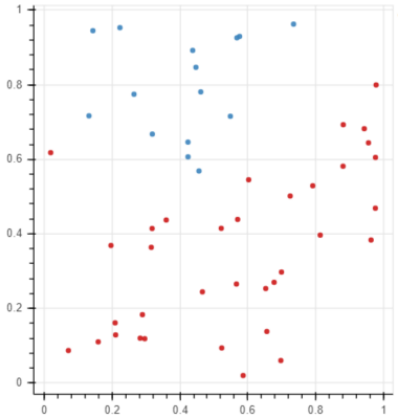
Data split

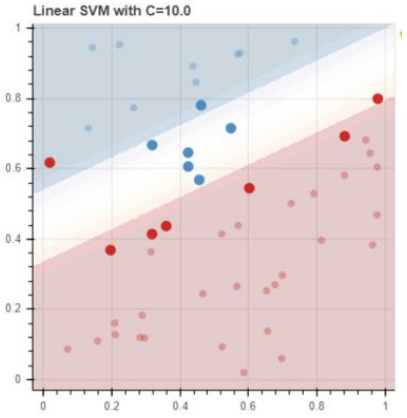
- Training set 의 일부를 validation set 으로 이용할 수도 있습니다.
- 적절한 하이퍼 패러매터의 탐색도 validation set 에 대한 학습입니다.
- 핵심은 학습이나 하이퍼 패러매터 탐색에 이용되지 않은 데이터로 모델의 성능을 평가하는 것입니다.



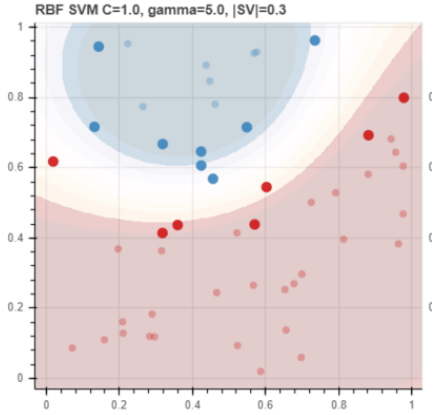
Bias and Variance

- 아래의 데이터를 구분하는 모델은 다양하며, 성능도 다른 경향을 보입니다.
  - Low biased model : 모델이 데이터의 분포를 잘 학습한 경우입니다.
  - Low variance model : 비슷한 x 에 대하여 비슷한 예측값을 보입니다.

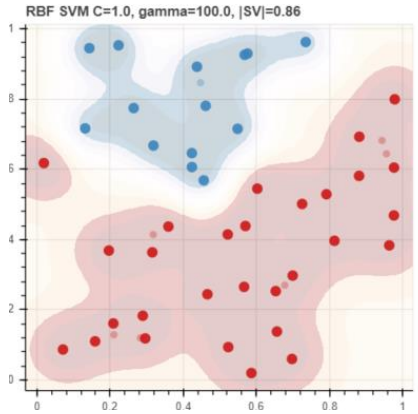




high bias, low variance



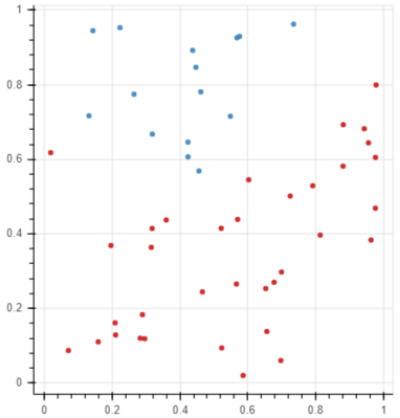
middle bias & variance

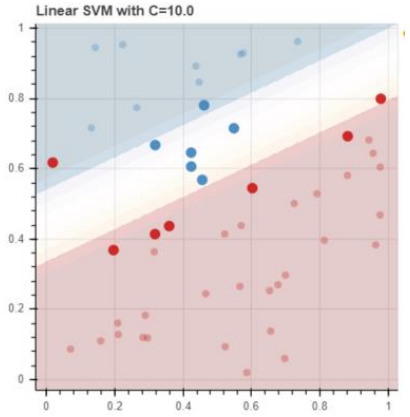


low bias, high variance

Bias and Variance

- High bias 를 지니는 모델은 데이터 분포 가정 ($\hat{f}$ 식의 구조) 이 틀린 가능성이 높습니다.
  - 모델이 지나치게 단순하여 데이터의 분포를 제대로 학습하지 못합니다. (under fitting)

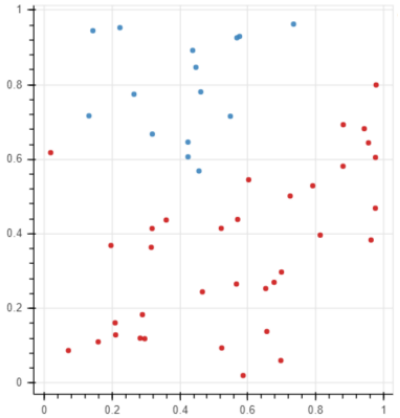


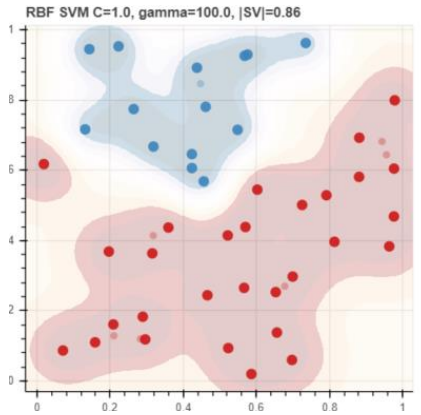


high bias, low variance

Bias and Variance

- High variance 를 지니는 모델은 과적합이 이뤄졌을 가능성이 높습니다.
  - 학습데이터의 노이즈까지도 학습될 가능성이 높습니다. (over fitting)

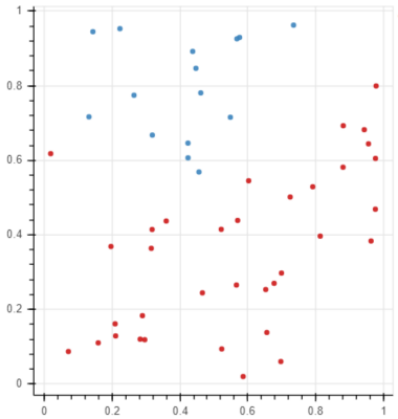


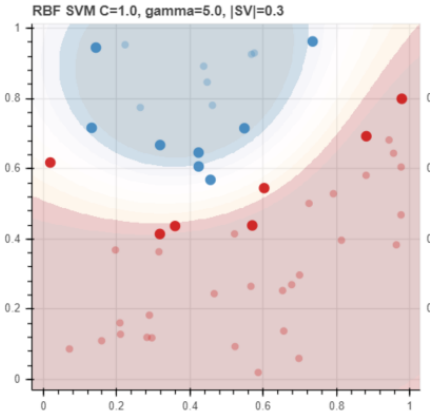


low bias, high variance

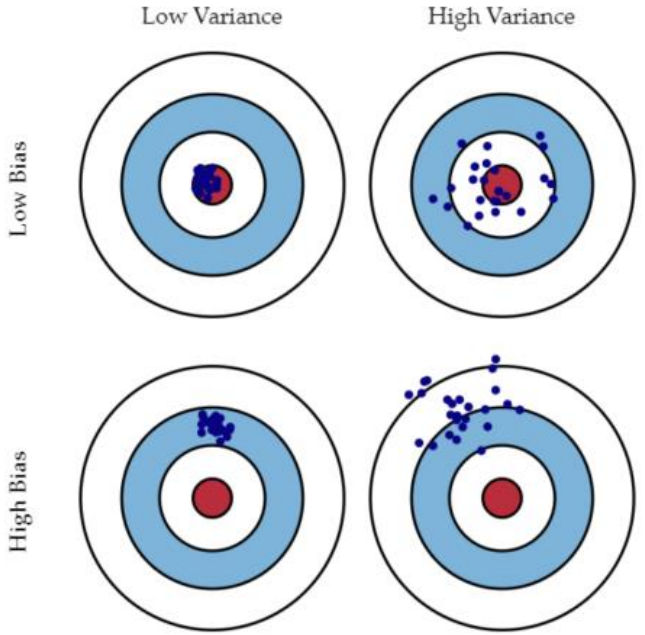
Bias and Variance

- Bias – variance 는 역관계에 있을 가능성이 높으며, bias 와 variance 가 모두 작을수록 학습이 잘 된 모델입니다. 일반화 성능이 좋습니다.





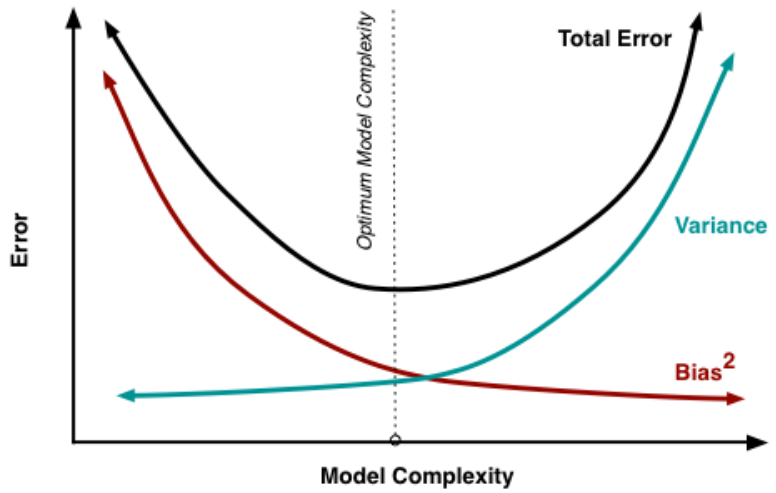
middle bias & variance



http://scott.fortmann-roe.com/docs/BiasVariance.html

Bias and Variance

- Validation data 에 대하여 모델이 복잡할수록 bias 는 감소하고, variance 는 증가하는 경향이 있습니다. 그 합의 최소점이 되는 모델을 선택하여 일반화 성능이 가장 좋은 모델을 선택합니다.



http://scott.fortmann-roe.com/docs/BiasVariance.html

Bias and Variance

- 모델의 일반화 성능은 bias 와 variance 의 합으로 이뤄져 있습니다.

$$E\left[\left(y-\hat{f}\right)^2\right]=\sigma^2+\text{bias}\left[\hat{f}\right]+\text{variance}\left[\hat{f}\right]$$

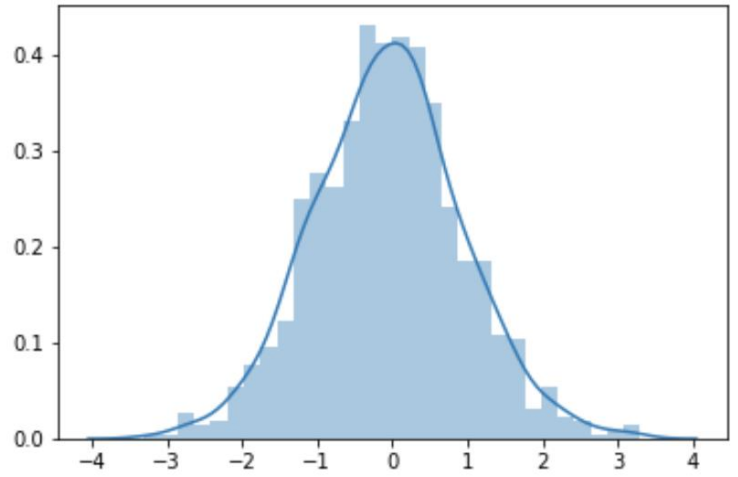
$\hat{f}$: trained model

Bias and Variance

- 보다 엄밀한 bias and variance 에 대한 분석과 일반화 성능에 대해서는 아래의 블로그를 읽는 것을 추천드립니다.
  - https://sebastianraschka.com/blog/2016/model-evaluation-selection-part1.html
  - https://sebastianraschka.com/blog/2016/model-evaluation-selection-part2.html
  - https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html
  - http://scott.fortmann-roe.com/docs/BiasVariance.html

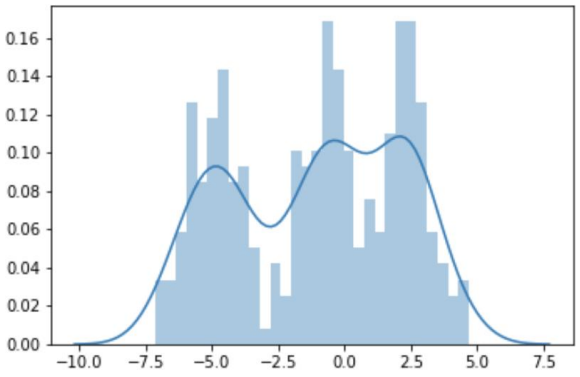
Bootstrap sampling

- 확률 밀도 함수가 알려져 있다면 다양한 통계를 계산할 수 있습니다.
  - 아래 그림은 $x \sim N(0,1)$ 의 정규분포를 따르는 샘플입니다.
  - 평균, 분산 뿐 아니라 confidence interval 도 손쉽게 계산할 수 있습니다.
  - p % 신뢰구간은 $(CDF_{\frac{p}{2}}(x), CDF_{1-\frac{p}{2}}(x))$ 으로 정의할 수 있습니다.

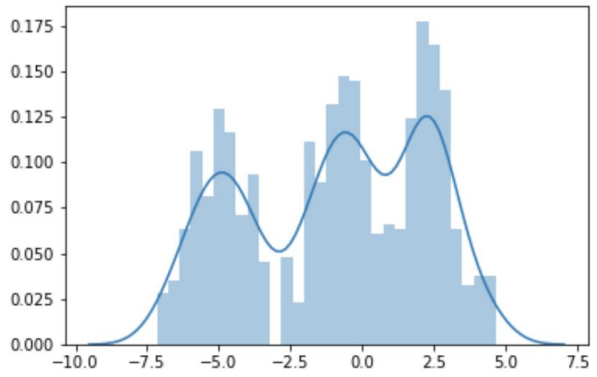


Bootstrap sampling

- 확률식이 알려져 있지 않거나 복잡하면 신뢰구간을 계산하기 어렵습니다.
  - Bootstrap sampling (resampling) 은 이 때 이용할 수 있는 방법으로,
  - n 개의 데이터에 대하여 b 번 복원 추출을 수행합니다. b 개의 데이터를 정렬한 다음 상한과 하한값을 선택합니다.
  - 확률식이 알려진 경우에도 이용할 수 있는 방법입니다.



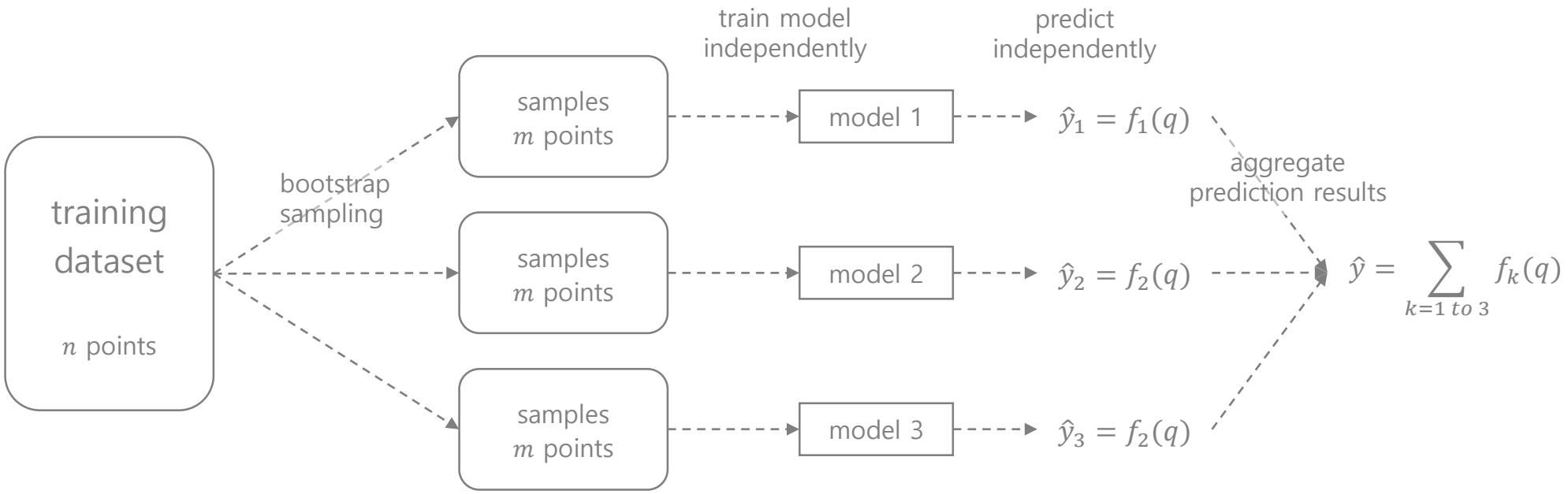
3 modal normal distribution



bootstrapped sample distribution

Bagging

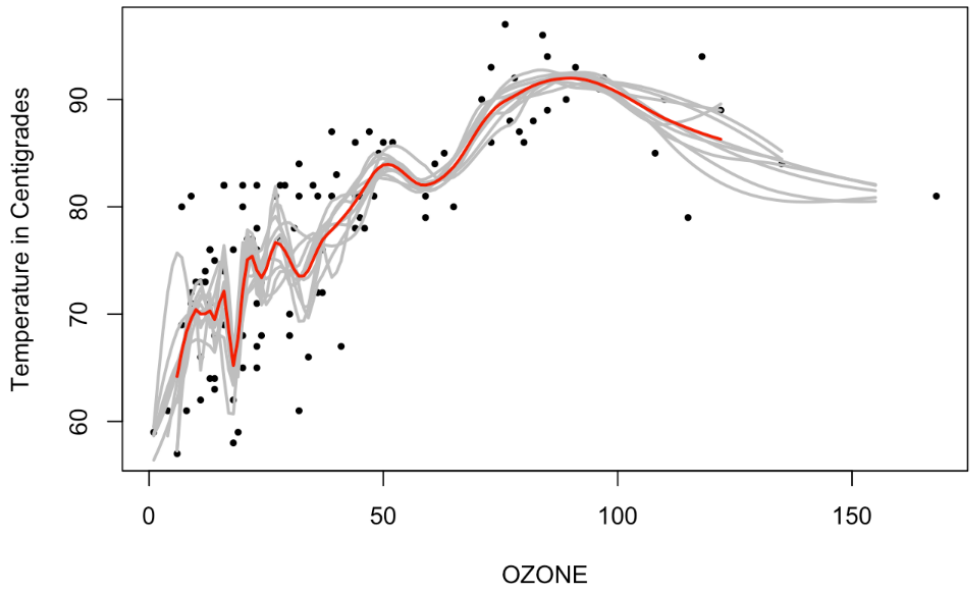
- Bootstrap aggregating 은 샘플링 된 데이터로부터 독립적인 여러 모델을 학습한 뒤, 각 모델의 결과를 종합하여 최종 결과로 이용합니다.



Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

Bagging

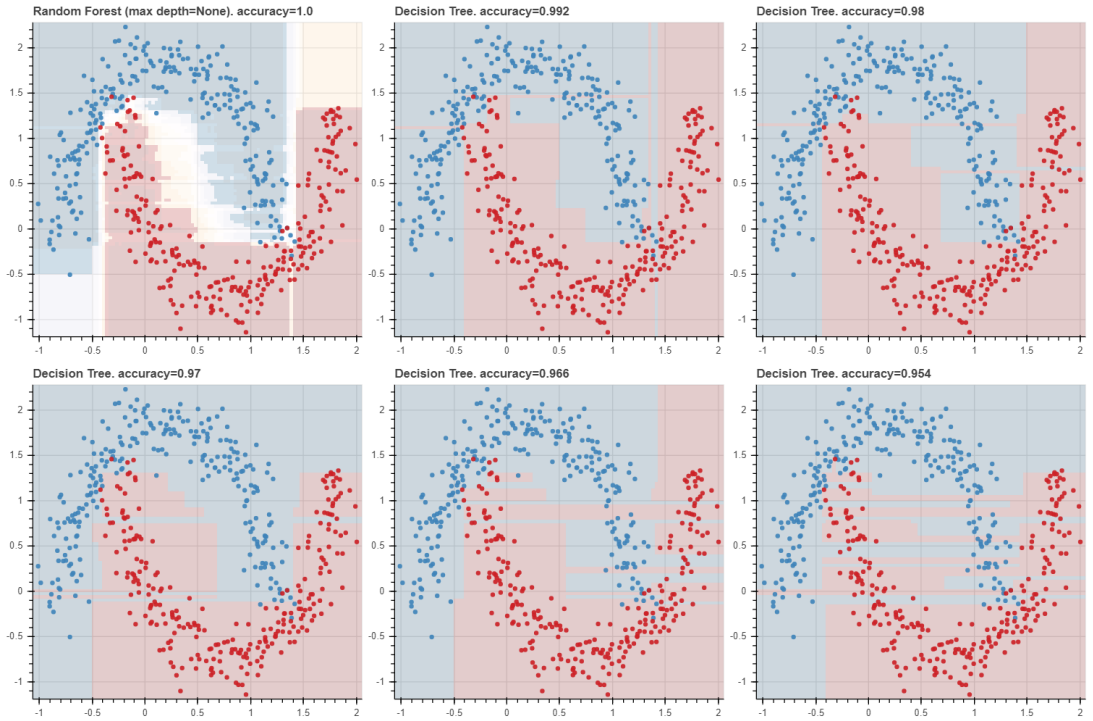
- 학습 성능이 뛰어난 모델들은 노이즈까지도 패턴으로 학습하여 과적합이 발생합니다.
- Bootstrap 을 통하여 조금씩 다른 데이터로 학습을 한 뒤, 이들의 예측값의 평균을 취하면 과적합이 되지 않을 가능성이 높습니다.



https://rpubs.com/ameilij/SNBagging

Random Forest

- Random Forest 는 대표적인 Bagging 방법입니다. 서로 다른 학습데이터에 과적합시킨 결정단면들을 겹치면 곡선과 비슷한 경계면을 얻을 수 있습니다.
- Low bias / high variance 를 지닌 모델들을 종합하여 variance 를 줄입니다 (일반화성능 증가)



Random Forest

- Correlation 이 적은 확률 변수들의 평균을 취하면 개별 변수의 분산보다 더 적은 분산을 얻을 수 있습니다
- Random Forest 는 서로 상관성이 적은 여러 개의 과적합된 모델들을 이용하여 전체 예측의 variance 를 줄이려는 것입니다.

$$Var(\bar{z}) = Var\left(\frac{Z_1+Z_2+\ldots+Z_n}{n}\right) = \frac{1}{n^2}\sum_{i,j} Cov(Z_i, Z_j)$$

$$= \frac{1}{n^2}\left(\sum_i Var(Z_i) + \sum_{i\neq j} Cov(Z_i, Z_j)\right)$$

$$= \frac{1}{n^2}(n\sigma^2 + 0) = \frac{\sigma^2}{n}$$

Random Forest

- 그러나 의사결정나무는 변별력이 좋은 변수 (information gain 이 큰) 를 우선적으로 이용하기 때문에 비슷한 형태의 모델들이 학습될 가능성이 높습니다.
- 이를 해결하기 위하여 변수의 일부를 임의로 제거한 뒤, (1) bootstrap 에 의해 서로 다른 데이터의 (2) 서로 다른 변수들을 이용한 의사결정나무들을 학습합니다.
  이를 subspace method (column sampling, feature bagging) 이라 합니다.

Breiman, L. (2001). Random forests. *Machine learning*, *45*(1), 5-32.

Random Forest

- 모델의 변별력과 상관성에 관련된 척도들입니다.
  - margin, $mr(x,y) = P_\Theta(h(x,\Theta) = y) - \max_{j \neq y} P_\Theta(h(x,\Theta) = j)$
  - strength, $s = E_{x,y}[mr(x,y)]$
  - 정답 레이블과 차순위 레이블 간의 예측확률의 차이가 클수록 모델이 확신을 가지고 분류를 하고 있다는 의미입니다.

Bernard, S., Heutte, L., & Adam, S. (2010, August). A study of strength and correlation in random forests. In *International Conference on Intelligent Computing* (pp. 186-191). Springer, Berlin, Heidelberg.

Random Forest

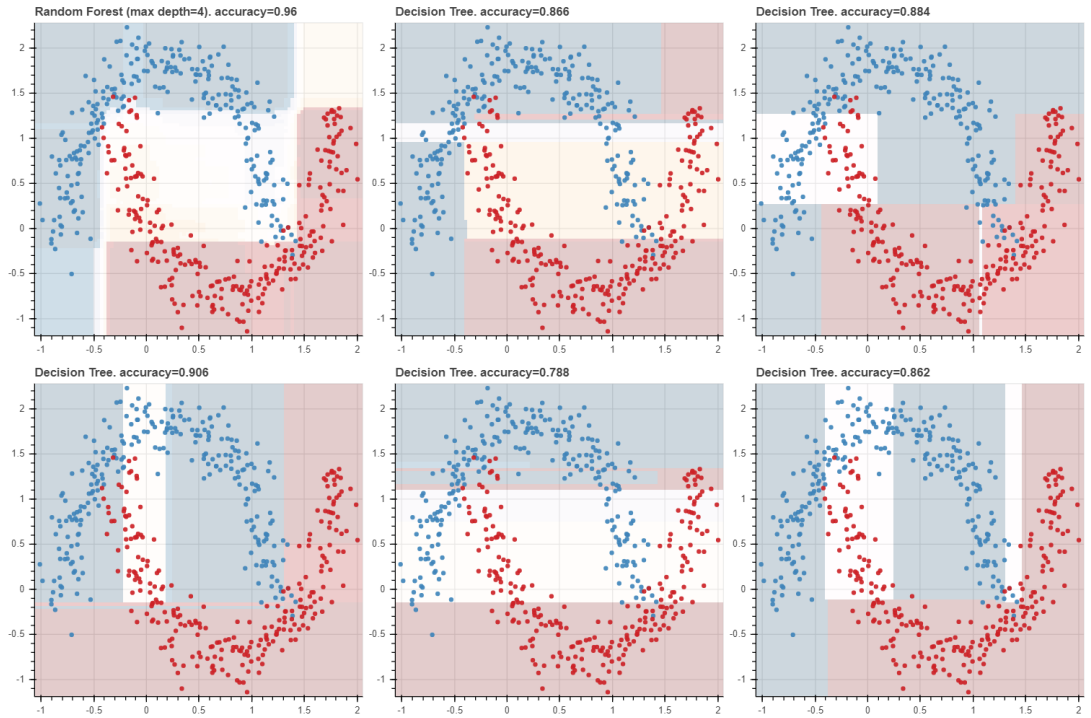
- 모델의 변별력과 상관성에 관련된 척도들입니다.
  - raw margin, $rm(\Theta, x, y) = I(h(x, \Theta) = y) - I\left(h(x, \Theta) = \hat{j}(x, y)\right)$

    $\hat{j}(x, y) = \arg\max_{j \neq y} P_{\Theta}(h(x, \Theta) = j)$
  - correlation$(\Theta, \Theta`) = corr\left(rm(\Theta, x, y), rm\left(\Theta`, x, y\right)\right)$
  - 모델 간에 정답을 맞추는 영역이 서로 다르며, 독립적이다는 의미입니다.

Bernard, S., Heutte, L., & Adam, S. (2010, August). A study of strength and correlation in random forests. In *International Conference on Intelligent Computing* (pp. 186-191). Springer, Berlin, Heidelberg.

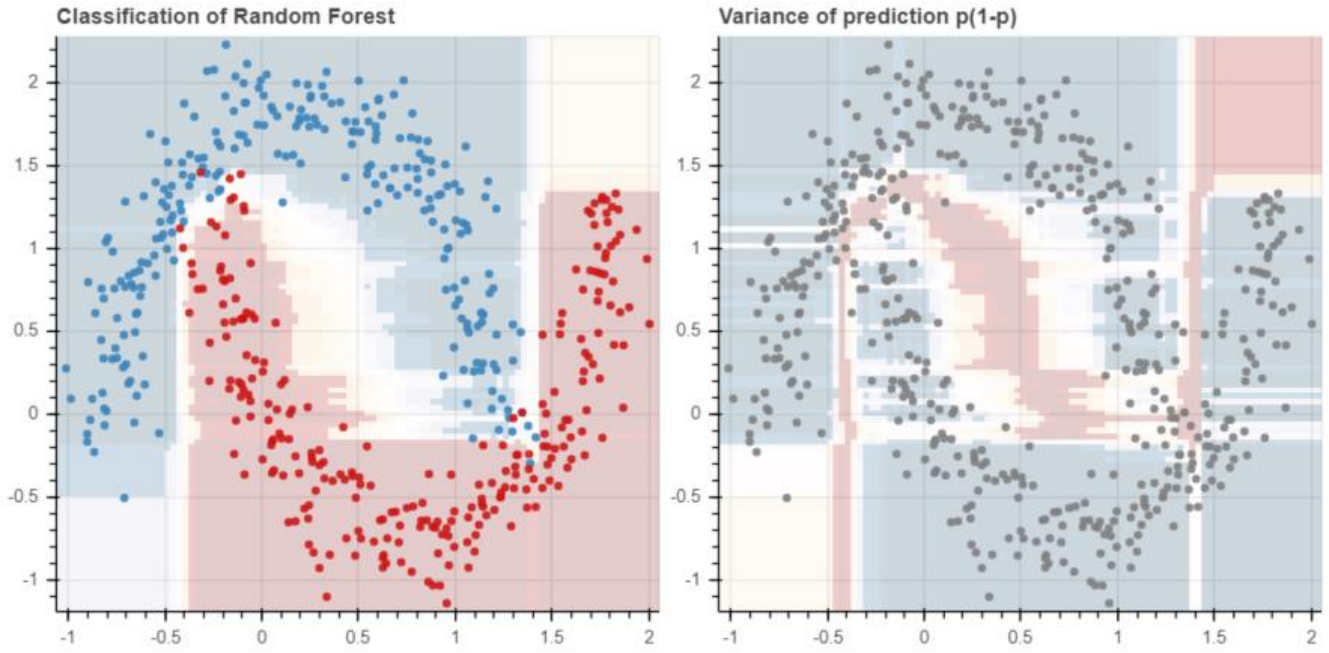
Random Forest

- Under-fitting 시킨 단면들도 여러 개 겹치면 충분한 분류 성능을 지니는 단면을 얻을 수도 있습니다.



Random Forest

- Confidence 도 계산할 수 있습니다. Prediction prob. 를 이용하여 $p(1-p)$ 혹은 $entropy$ 를 계산하여 확인해봅니다.
- 데이터가 존재하지 않거나 서로 다른 클래스의 데이터들이 뭉쳐 있는 지역에서 낮은 confidence (높은 variance) 가 발생함을 확인할 수 있습니다.



Random Forest

- Regression 문제에서도 각 모델 f_k 의 예측값 $\hat{y}_k = f_k(q)$ 의 분산을 통하여 confidence 를 계산할 수 있습니다.

$$\sigma_i = \sqrt{\frac{\sum_1^K \left(\hat{y}_{k,i} - \hat{y}_i\right)^2}{K}}$$

$\hat{y}_{k,i}$: predicted value, $f_k(x_i)$

Random Forest

- Classifiers 는 벡터 공간에서 동일한 클래스의 데이터들이 모여있는 지역을 구분함으로써 분류작업을 수행합니다.
  - Neural network 는 hidden layer 를 통하여 직접적으로 representation 을 변환하고
  - Support Vector Machine 은 kernel trick 을 이용하여 가상의 representation 변환을 합니다.
  - Decision Tree 는 rectangular 형태의 공간들을 탐색합니다. 위 두 모델과 달리 여러 변수들을 조합한 경계면을 만들기 어렵습니다. Random Forest 는 여러 개의 decision tree 의 경계면을 겹침으로써 variance 를 낮추는 동시에 곡선의 경계면을 학습합니다.

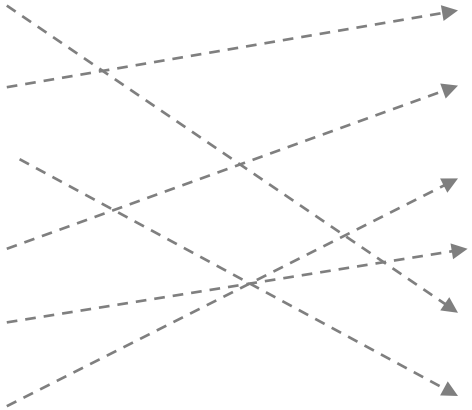
Random Forest

- Out of bag error
  - n 개의 데이터에서 n 개의 데이터를 bootstrap 할 경우 $\frac{n}{3}$ 이 추출되지 않습니다.
  - 이를 out of bag dataset 이라 하며, random forest 를 구성하는 각 의사결정나무의 성능을 측정할 수 있습니다. (out of bag error)
  - 이들의 성능의 평균을 out of bag score 라 합니다. 이는 각 의사결정나무의 내부적인 validation set 의 역할과 같으며, random forest 의 전체 성능은 test set 을 이용하여 검증합니다.

Random Forest

- Permutation importance 는 변수 간 중요도를 측정할 수 있습니다.
  - 모델의 학습을 완료한 뒤, 데이터의 한 변수의 값을 임의로 shuffle 합니다. Shuffle 하기 전후 성능의 차이가 크면 해당 변수가 분류에 큰 영향을 준다는 의미입니다.

Var 2	Var 3	Var 4
…	A	…
…	B	…
…	C	…
…	D	…
…	A	…
…	D	…

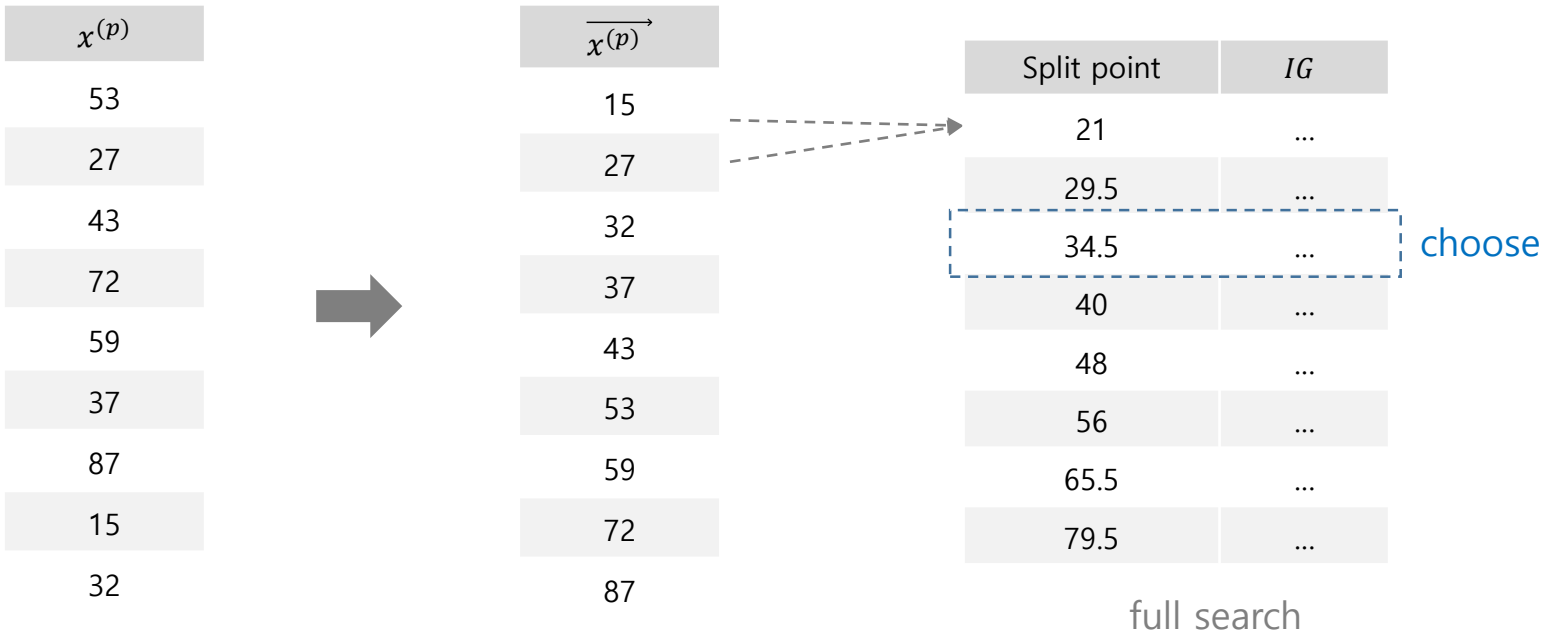


Var 2	Var 3	Var 4
…	B	…
…	D	…
…	D	…
…	A	…
…	A	…
…	C	…

https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html

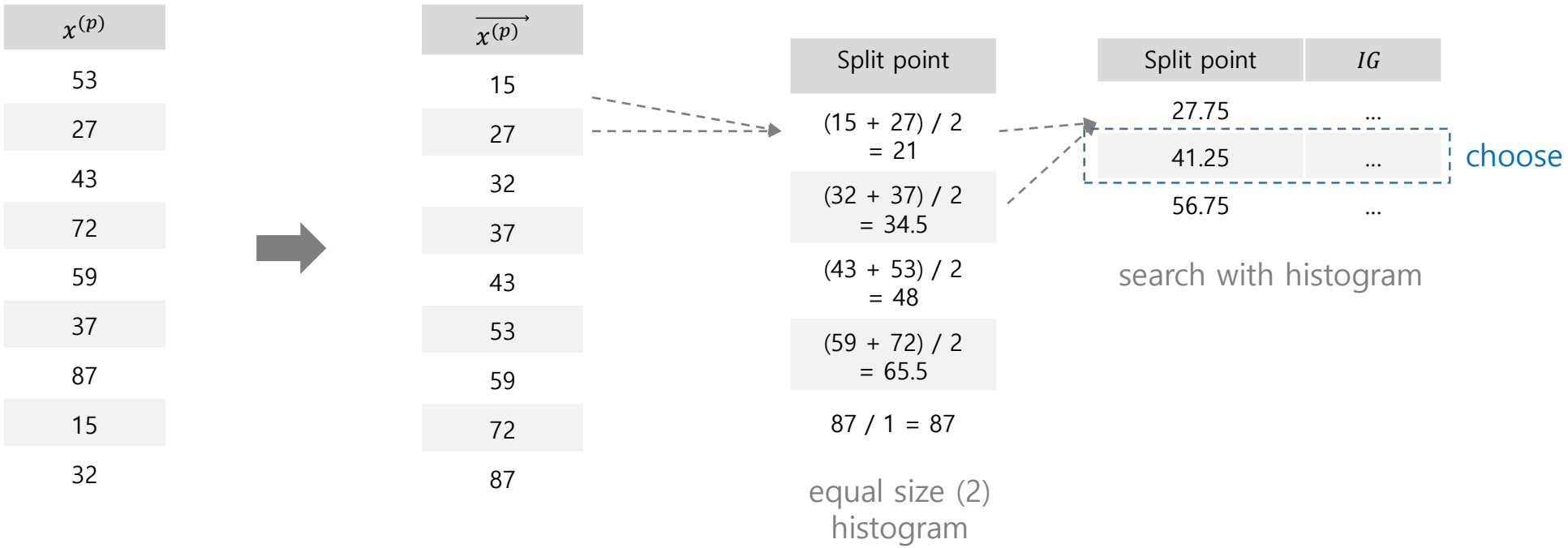
Histogram based Forest

- 의사결정나무의 학습은 분기 지점 탐색에서 가장 많은 계산이 일어납니다.
  - 모든 점의 탐색 대신 histogram 을 이용할 수도 있습니다.



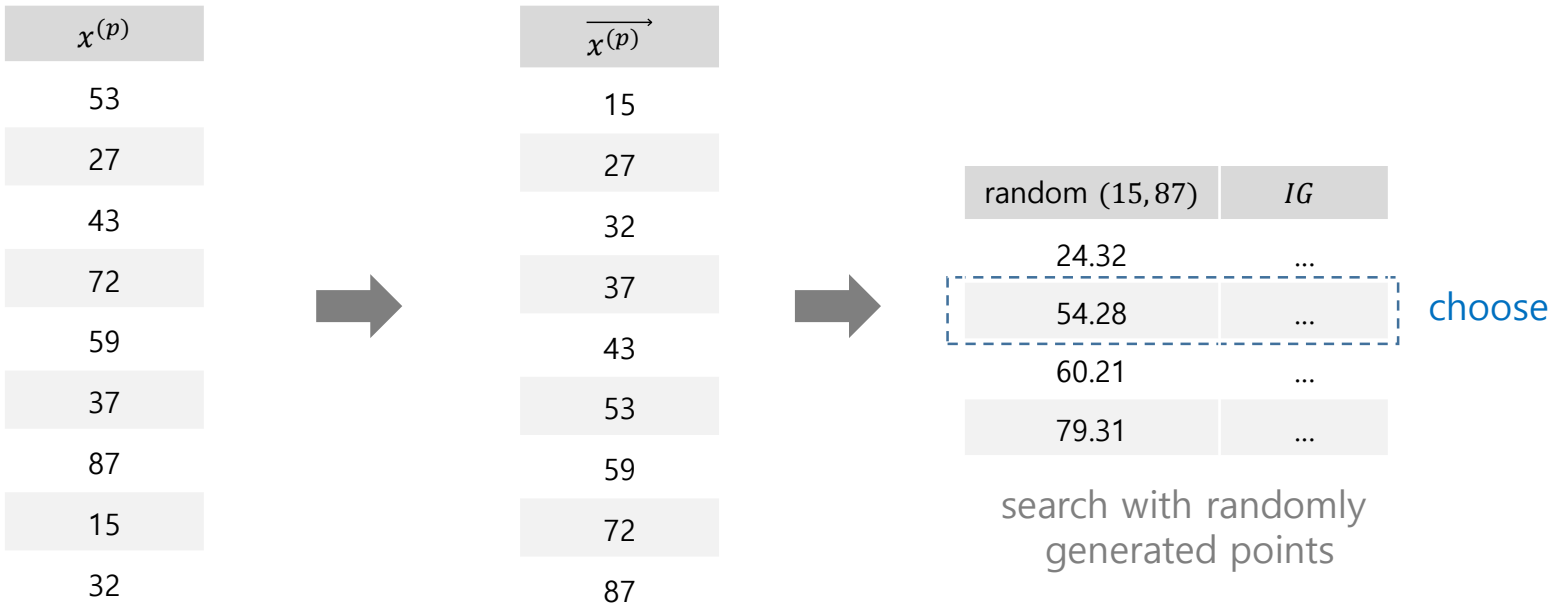
Histogram based Forest

- 의사결정나무의 학습은 분기 지점 탐색에서 가장 많은 계산이 일어납니다.
  - 모든 점의 탐색 대신 histogram 을 이용할 수도 있습니다.



Extra Trees

- 의사결정나무의 학습은 분기 지점 탐색에서 가장 많은 계산이 일어납니다.
  - $x^{(p)}$ 의 범위 내에서 임의의 분기 지점을 생성한 뒤, IG 가 가장 큰 점을 선택합니다.
  - Randomness 를 이용하여 반복 학습을 할 때는, 매번 정확히 학습할 필요가 없습니다.



Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. Machine learning, 63(1), 3-42.

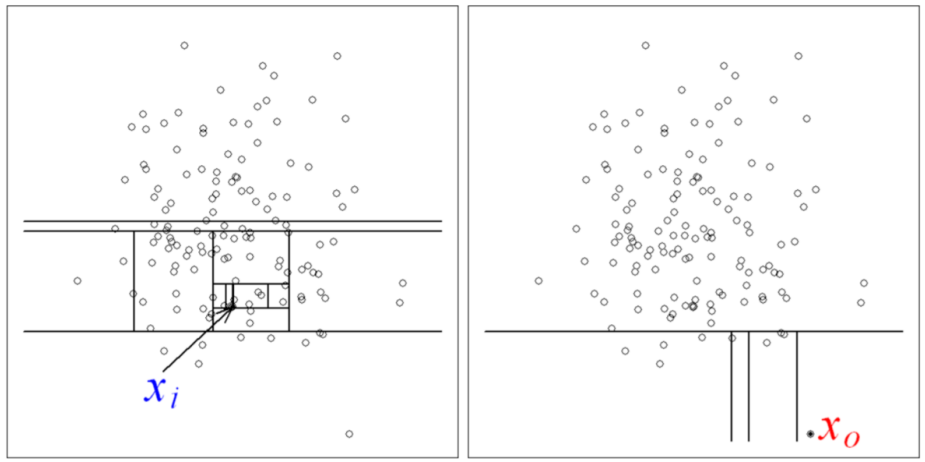
Isolation Forest

- Decision trees 를 이용하여 outliers 를 탐색합니다.
  - Forest 를 구성하는 각 decision tree 는 information gain 등을 계산하지 않고, 임의의 변수에 대해 임의의 값으로 분기를 합니다.
  - 다른 데이터들과 떨어진 데이터는 몇 번의 분기만으로도 크기가 1 인 leaf 가 됩니다.
  - 학습데이터의 개수가 다르기 때문에 모든 tree 가 동일한 개수의 (약 256 개) 샘플에 대해서만 위 과정을 진행합니다.

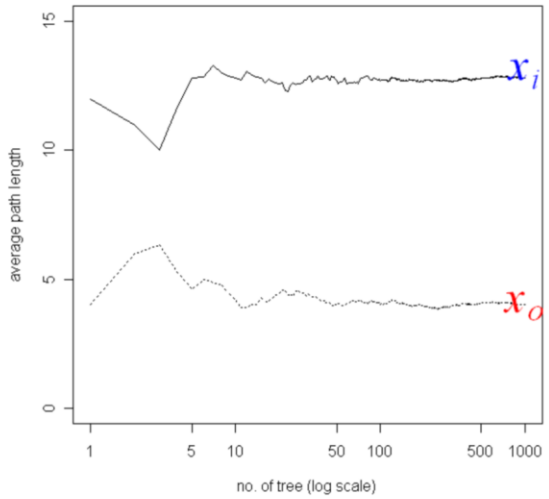
Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE.

Isolation Forest

- Decision trees 를 이용하여 outliers 를 탐색합니다.
  - Subsamples 의 각 점들이 각각 leaf 가 되는 평균 깊이를 측정합니다.
  - 다른 점들과 떨어진 점들은 수 번의 분기 만으로도 고립됩니다.



(a) Isolating x_i (b) Isolating x_o

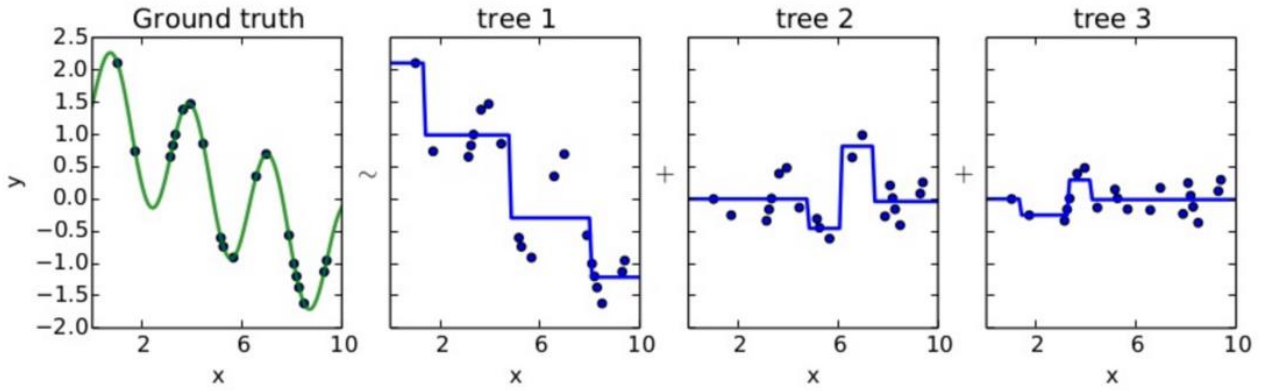


(c) Average path lengths converge

Liu, F. T., Ting, K. M., & Zhou, Z. H. (2008, December). Isolation forest. In 2008 Eighth IEEE International Conference on Data Mining (pp. 413-422). IEEE.

Bagging vs Boosting

- 좋은 모델은 bias 와 variance 가 모두 낮은 모델입니다.
- Bagging 은 low bias, high variance 인 모델들을 종합함으로써 variance 를 낮춥니다.
- Boosting 은 high bias, low variance 인 종합함으로써 bias 를 낮춥니다.
  - 분포 가정이 잘못된 (under-fitting 된) 모델들의 분포를 조금씩 조절합니다.
  - $f_1, f_2, \ldots f_{k-1}$ 까지의 모델에 의하여 생성된 오류를 f_k 모델이 보완하도록 학습합니다.



Boosting

- Boosting 은 현재의 모델들로 해결하지 못한 나머지 부분 (residuals) 을 해결할 수 있는 모델을 하나씩 더해가는 방법입니다.

$$y = f_1(x) + e_1$$

$$e_1 = f_2(x) + e_2$$

$$e_2 = f_3(x) + e_3$$

$$\rightarrow y = f_1(x) + f_2(x) + f_3(x) + e_3$$

Adaptive Boosting

- Adaptive Boosting (adaboost^1) 는 residual 을 직접 학습하는대신, $f_{1:k-1}$ 들이 잘 예측하지 못한 데이터 포인트를 더 잘 맞추도록 f_k 를 학습합니다.
- f_i 는 매우 단순한 (depth = 4 ~ 8, decision tree) 모델을 이용합니다.

w_i : weight of (x_i, y_i)

ϵ_k : 오류 율

$\alpha_k = \frac{1}{2}\ln\left(\frac{1-\epsilon_k}{\epsilon_k}\right)$: f_k 의 모델 가중치

$$w_i^{k+1} = \begin{cases} w_i^k \times e^{-\alpha_k} & \text{IF } F_k(x_i) = y_i \\ w_i^k \times e^{\alpha_k} & \text{IF } F_k(x_i) \neq y_i \end{cases}$$

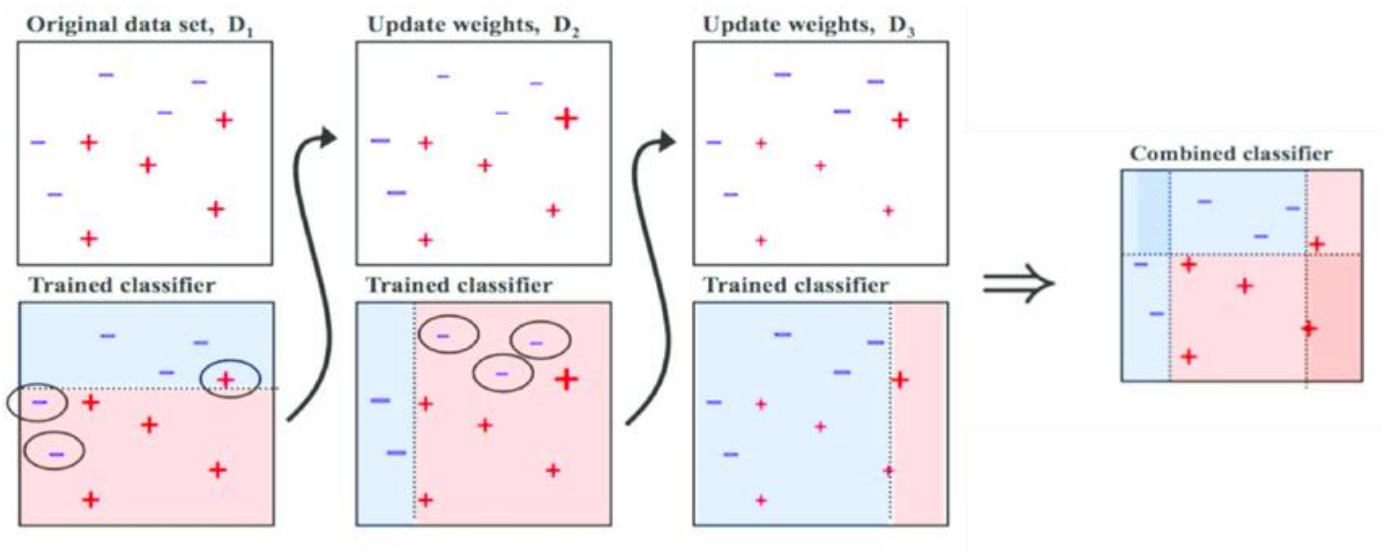


Illustration from ^2

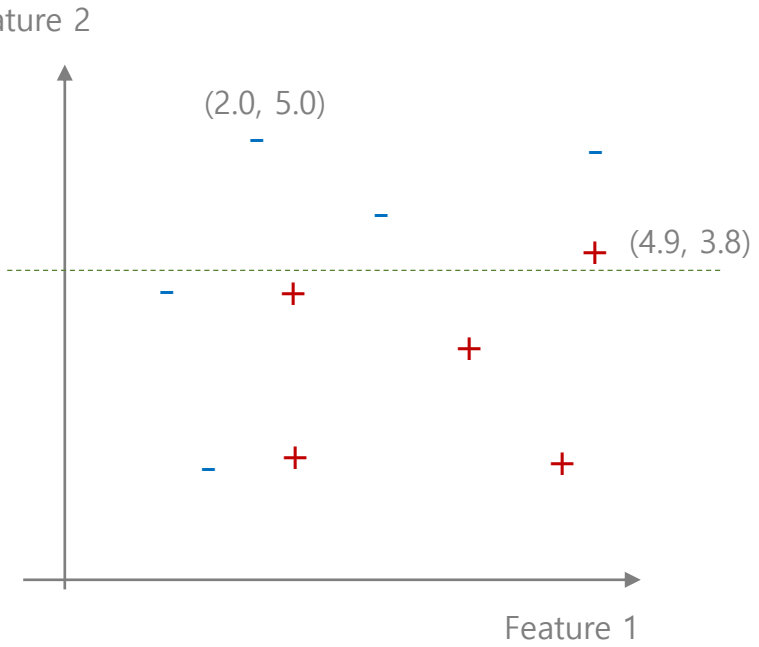
^1: Freund, Y., & Schapire, R. E. (1995, March). A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory* (pp. 23-37). Springer, Berlin, Heidelberg.

^2: https://medium.com/diogo-menezes-borges/boosting-with-adaboost-and-gradient-boosting-9cbab2a1af81

Adaptive Boosting

- Initial sample weight, $w_i^{k=1} = \frac{1}{n}$ 으로 설정합니다.
- $X^{(k=1)} = \{(x_1, y_1), \ldots (x_n, y_n)\}$ 으로 f_1 을 학습합니다.

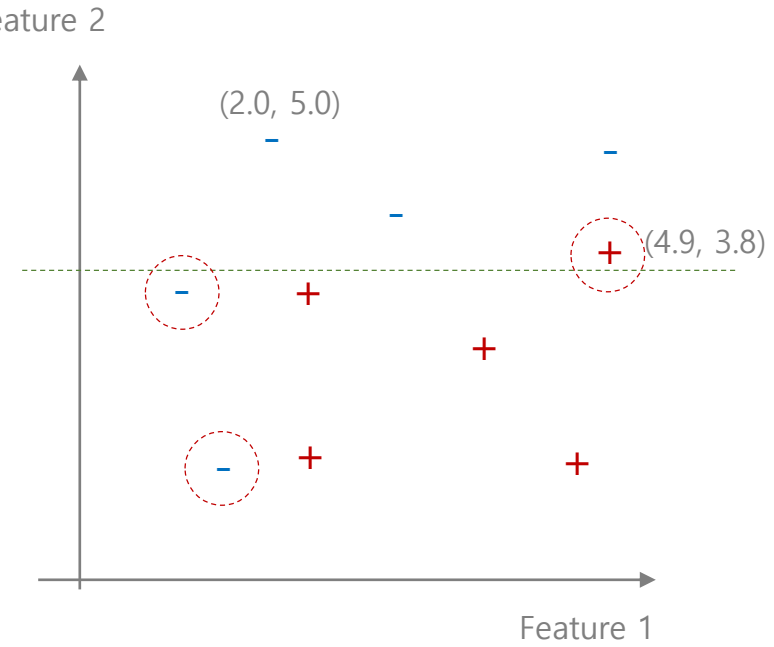
Feature 1	Feature 2	Class	w_i^1
2.0	5.0	-	1/10
5.0	4.8	-	1/10
3.5	4.5	-	1/10
1.0	3.3	-	1/10
1.5	1.5	-	1/10
2.2	3.3	+	1/10
4.9	3.8	+	1/10
4.0	2.5	+	1/10
2.2	1.6	+	1/10
4.5	1.6	+	1/10



Adaptive Boosting

- $\epsilon_1 = \frac{3}{10}, \alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right)$ 을 이용하여 w_i^{k+1} 를 업데이트 합니다.
- $F_1 = f_1$ 로 모델을 업데이트 합니다.

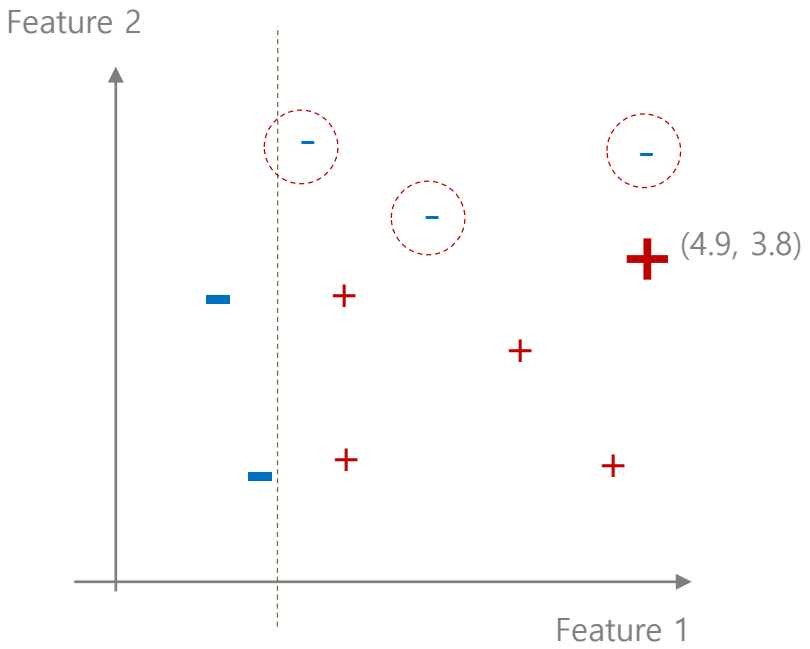
Feature 1	Feature 2	Class	w_i^1	w_i^2
2.0	5.0	-	1/10	$0.1 \times e^{-\alpha_1}$
5.0	4.8	-	1/10	$0.1 \times e^{-\alpha_1}$
3.5	4.5	-	1/10	$0.1 \times e^{-\alpha_1}$
1.0	3.3	-	1/10	$0.1 \times e^{+\alpha_1}$
1.5	1.5	-	1/10	$0.1 \times e^{+\alpha_1}$
2.2	3.3	+	1/10	$0.1 \times e^{-\alpha_1}$
4.9	3.8	+	1/10	$0.1 \times e^{+\alpha_1}$
4.0	2.5	+	1/10	$0.1 \times e^{-\alpha_1}$
2.2	1.6	+	1/10	$0.1 \times e^{-\alpha_1}$
4.5	1.6	+	1/10	$0.1 \times e^{-\alpha_1}$



Adaptive Boosting

- w_i^2 를 이용하여 weighted Gini index 를 이용하는 decision tree f_2 를 학습합니다.
- $\epsilon_2 = \frac{3}{10}, \alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_2}{\epsilon_2}\right)$ 를 이용하여 w_i^3 를 정의합니다.
- $F_2 = \sum_k f_k$ 로 모델을 업데이트 합니다.

Feature 1	Feature 2	Class	w_i^2	w_i^3
2.0	5.0	-	$0.1 \times e^{-\alpha_1}$	$0.1 \times e^{-\alpha_1} \times e^{+\alpha_2}$
5.0	4.8	-	$0.1 \times e^{-\alpha_1}$	$0.1 \times e^{-\alpha_1} \times e^{+\alpha_2}$
3.5	4.5	-	$0.1 \times e^{-\alpha_1}$	$0.1 \times e^{-\alpha_1} \times e^{+\alpha_2}$
1.0	3.3	-	$0.1 \times e^{+\alpha_1}$	$0.1 \times e^{+\alpha_1} \times e^{-\alpha_2}$
1.5	1.5	-	$0.1 \times e^{+\alpha_1}$	$0.1 \times e^{+\alpha_1} \times e^{-\alpha_2}$
2.2	3.3	+	$0.1 \times e^{-\alpha_1}$	$0.1 \times e^{-\alpha_1} \times e^{-\alpha_2}$
4.9	3.8	+	$0.1 \times e^{+\alpha_1}$	$0.1 \times e^{+\alpha_1} \times e^{-\alpha_2}$
4.0	2.5	+	$0.1 \times e^{-\alpha_1}$	$0.1 \times e^{-\alpha_1} \times e^{-\alpha_2}$
2.2	1.6	+	$0.1 \times e^{-\alpha_1}$	$0.1 \times e^{-\alpha_1} \times e^{-\alpha_2}$
4.5	1.6	+	$0.1 \times e^{-\alpha_1}$	$0.1 \times e^{-\alpha_1} \times e^{-\alpha_2}$



Adaptive Boosting

- 이 과정을 모델의 분류 성능이 수렴하거나, 하이퍼 패러매터로 설정한 max num. of decision tree 까지 반복합니다.
- Weighted Gini index 를 이용하여 개별 decision tree 를 학습하는 대신, w_i^k 를 이용하여 weighted bootstrap 을 수행합니다. w_i^k 가 높은 점이 여러 번 중복 추출되어 f_k 는 그 점들을 우선적으로 분류하는 모델을 학습합니다.
- $\epsilon_k = \frac{1}{2}$ 이면 $\alpha_k = \frac{1}{2}\ln\left(\frac{1-\epsilon_k}{\epsilon_k}\right) = 0$ 이어 $w_i^k = w_i^{k+1}$ 이 됩니다. 하지만 resampling 을 수행하면 $f_k \neq f_{k+1}$ 인 모델이 학습될 수 있습니다.

Adaptive Boosting

- Adaboost 는 regression 에도 이용될 수 있습니다. w_i^{k+1} 만 다른 방식으로 정의합니다.
  - step 1. define loss

    $L_i \coloneqq loss(F_k(x_i), y_i),\ D \coloneqq \sup|F_k(x_i) - y_i|$

    $L_i = \frac{|F_k(x_i)-y_i|}{D}$: linear

    $L_i = \frac{|F_k(x_i)-y_i|^2}{D}$: square law

    $L_i = 1 - \exp\left(\frac{|F_k(x_i)-y_i|}{D}\right)$: linear

  - step 2. define $\bar{L} = \sum_i w_{k,i}\,, \beta = \frac{\bar{L}}{1-\bar{L}}$
  - step 3. $w_i^{k+1} = w_i^k \times \beta^{1-L_i}$

Drucker, H. (1997, July). Improving regressors using boosting techniques. In *ICML* (Vol. 97, pp. 107-115).

Adaptive Boosting vs Random Forest

- Random Forest (RF) 는 모든 나무의 중요도가 같으며, 그 중에는 성능이 좋지 않은 모델도 포함되어 있습니다. Adaboost 는 오류율을 기반으로 중요도를 다르게 정의합니다.
- RF 는 임의성 (bootstrap, feature bagging) 에 의존하기 때문에 재현이 되지 않을 수 있습니다.
- RF 는 개별 모델 f_k 를 추가할 때마다 variance 를 줄여가며 F_k 의 성능을 향상시키며, Adaboost 는 f_k 를 추가할 때마다 F_k 의 bias 를 줄여가며 성능을 향상시킵니다.

Adaptive Boosting

- Boosting 에 decision tree 외에도 다른 모델들을 이용할 수도 있습니다. 하지만 다른 모델들은 학습 시간이 상대적으로 오래걸리며, 패러매터의 크기도 큽니다.
  - (100 차원 10 classes logistic regression) x (100 models)
    = 1000 x 100

Gradient Boosting

- Adaboost 는 모든 점들의 가중치 $w_{k+1,i}$ 가 f_k 의 성능인 ϵ_k 에 의하여 일괄적으로 업데이트 됩니다. 개별 점들마다 $w_{k+1,i}$ 를 다르게 정의할 수 있다면 더 효율적으로 f_k 를 학습할 수 있습니다.
- Gradient boosting 부터 f_k 가 residual 을 직접 학습합니다.

$$f_k(x_i) = e_i$$

Gradient Boosting

Pseudo code

$$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$$

For m in $1{:}\,M$:

$$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$$

$$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$$

Return $F_M(x)$

Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- $y = F(x)$ 인 gradient boosting regression model 을 학습합니다.
  - Loss, L 를 정의해야 합니다.

    예시에서는 L2 loss, $L = \sum_i (y_i - F(x_i))^2$ 를 이용합니다.

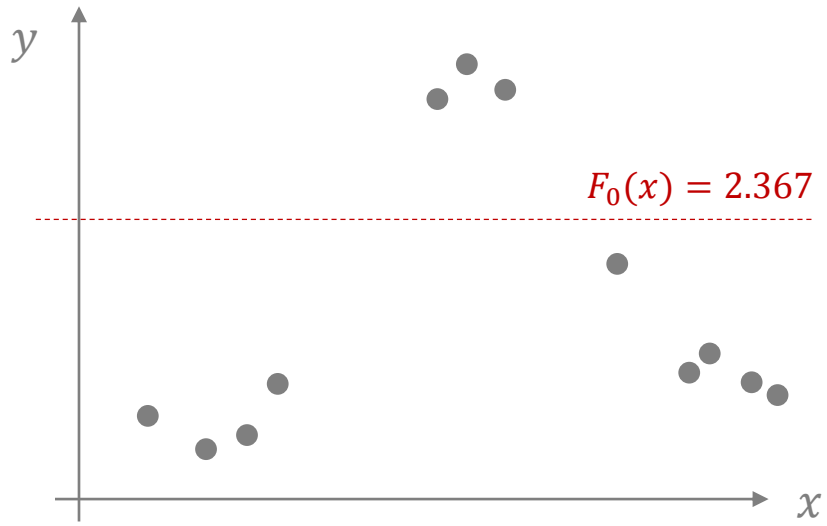
x	y
0	0.5
0.5	0.2
0.7	0.3
0.9	0.6
3	5
3.2	5.2
3.5	5.1
5.2	3.3
5.5	2.1
5.6	2.2
6	2
6.2	1.9

Gradient Boosting

- Loss 를 최소화하는 γ 를 $F_0(x_i) = \gamma$ 로 정의합니다.

$$\frac{\mathrm{d}L}{d(F(x_i))} = 2\sum_i (y_i - F_0(x_i)) = 0$$

$$\rightarrow F_0(x_i) = \bar{y} = 2.367$$



$F_0(x) = \arg\min_\gamma \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^n$

$\gamma_{i,m} = \arg\min_\gamma \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

x	y
0	0.5
0.5	0.2
0.7	0.3
0.9	0.6
3	5
3.2	5.2
3.5	5.1
5.2	3.3
5.5	2.1
5.6	2.2
6	2
6.2	1.9

Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

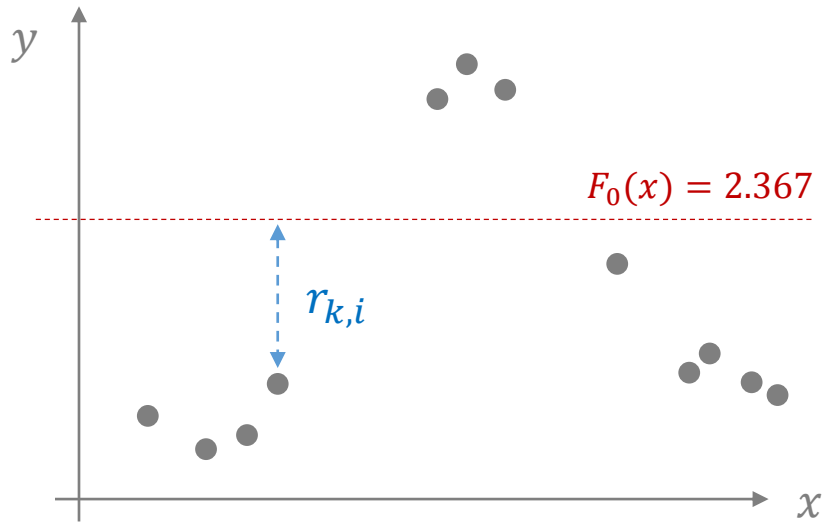
train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- Residual, $r_{k,i} = y_i - F_{k-1}(x_i)$ 를 정의합니다.
  - $f_1(x_i) = r_{0,i}$ 를 정확하게 예측할 수 있다면,
    $F_1(x_i) = f_0(x_i) + f_1(x_i) = y_i$ 가 됩니다.



r	x	y
-1.87	0	0.5
-2.17	0.5	0.2
-2.07	0.7	0.3
-1.77	0.9	0.6
2.63	3	5
2.83	3.2	5.2
2.73	3.5	5.1
0.93	5.2	3.3
-0.27	5.5	2.1
-0.17	5.6	2.2
-0.37	6	2
-0.47	6.2	1.9

Gradient Boosting

$F_0(x) = \arg\min_\gamma \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

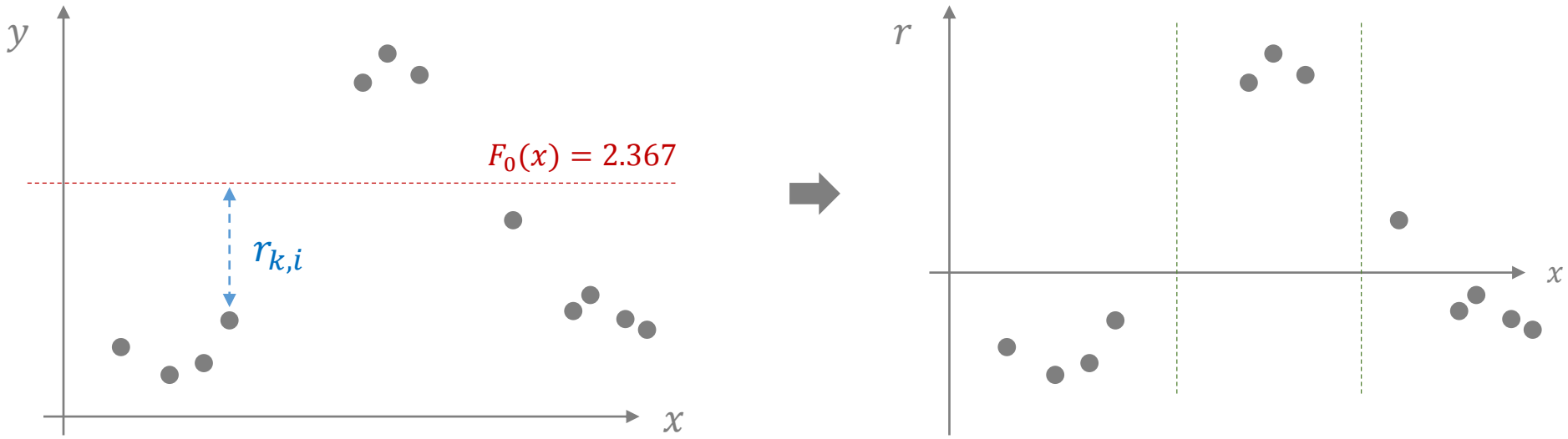
train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^n$

$\gamma_{i,m} = \arg\min_\gamma \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- Residual, $r_{k,i} = y_i - F_{k-1}(x_i)$ 를 정의합니다.
  - $f_1(x_i) = r_{0,i}$ 를 정확하게 예측할 수 있다면,
    $F_1(x_i) = f_0(x_i) + f_1(x_i) = y_i$ 가 됩니다.
  - $r_{k,i} = f_k(x_i)$ 를 학습합니다. x 가 세 부분으로 나뉘어집니다.



Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

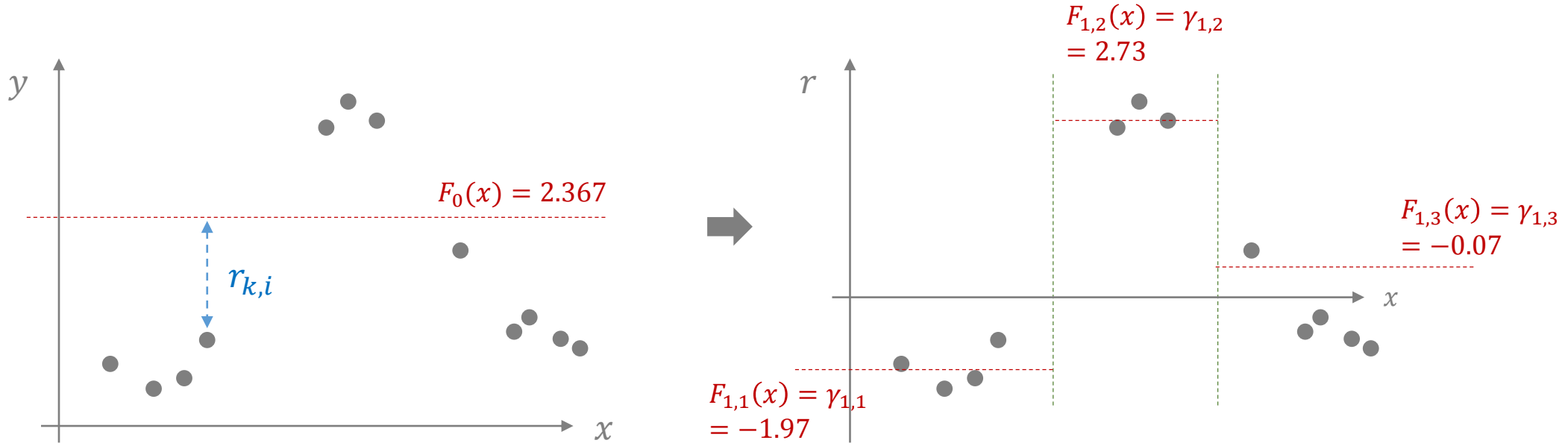
$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- 각 영역에서 $L(F_k(x_i), r_{k,i})$ 를 최소화하는 γ 를 탐색합니다.

  L2 loss 에서 γ 는 $\bar{y}$, $\bar{\gamma}_j$ 입니다.



Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

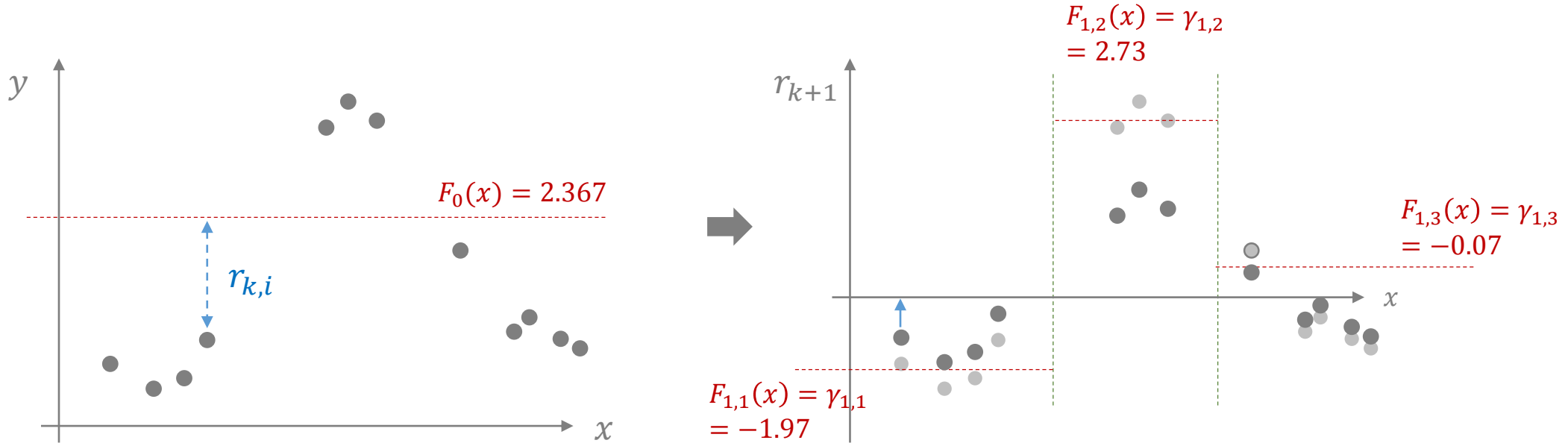
$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- 예측값 $\gamma_{m,j}$ 에 learning rate, ν 를 곱한만큼 residual 을 줄입니다.

  $r_{k+1,i} = r_{k,i} - \nu\gamma_{k,j}$ 입니다.



Gradient Boosting

$F_0(x) = \arg\min_\gamma \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^n$

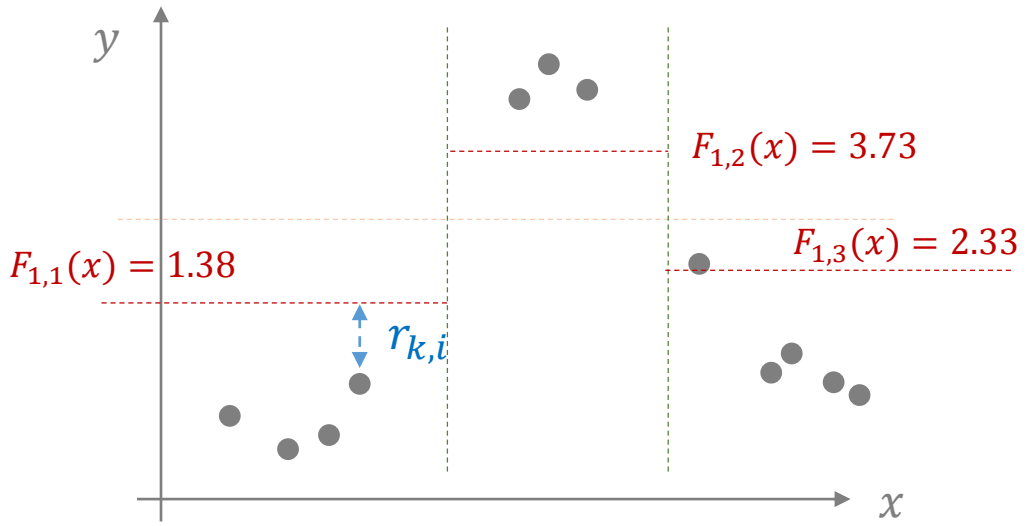
$\gamma_{i,m} = \arg\min_\gamma \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- 예측값 $\gamma_{m,j}$ 에 learning rate, ν 를 곱한만큼 residual 을 줄입니다.

  $r_{k+1,i} = r_{k,i} - \nu\gamma_{k,j}$ 입니다.



$\nu = 0.5$

r_{k+1}	r_k	x	y
-0.88	-1.87	0	0.5
-1.18	-2.17	0.5	0.2
-1.08	-2.07	0.7	0.3
-0.78	-1.77	0.9	0.6
1.27	2.63	3	5
1.47	2.83	3.2	5.2
1.37	2.73	3.5	5.1
0.97	0.93	5.2	3.3
-0.23	-0.27	5.5	2.1
-0.13	-0.17	5.6	2.2
-0.33	-0.37	6	2
-0.43	-0.47	6.2	1.9

Gradient Boosting

$F_0(x) = \arg\min_\gamma \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

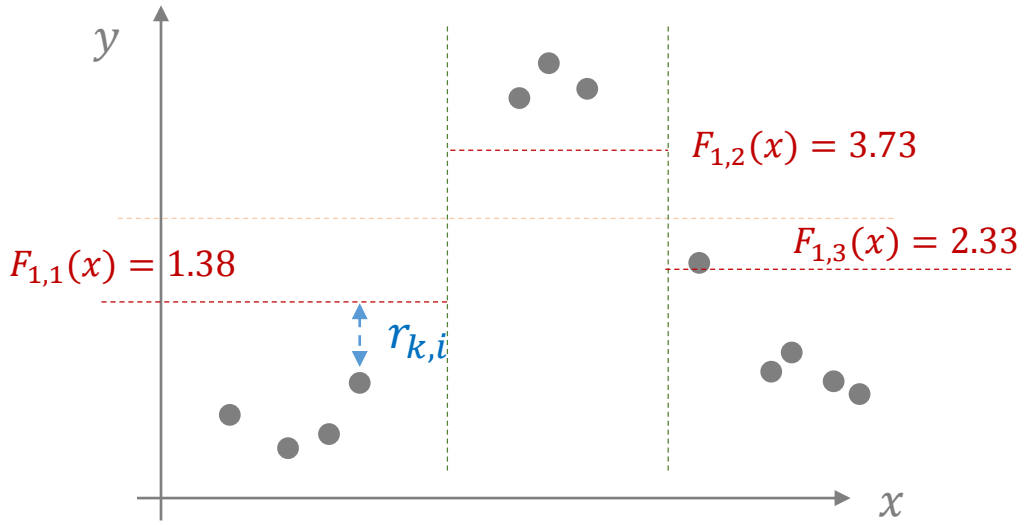
train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^n$

$\gamma_{i,m} = \arg\min_\gamma \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- 이 과정을 반복합니다. 경사하강법처럼 learning rate 를 곱하여 모델을 개선함으로써 bias 가 다시 늘어나는 것을 방지합니다.
- 다른 round 에서는 다른 부분으로 x 를 나눕니다. Loss 가 큰 x_i 에 집중적으로 학습이 됩니다.



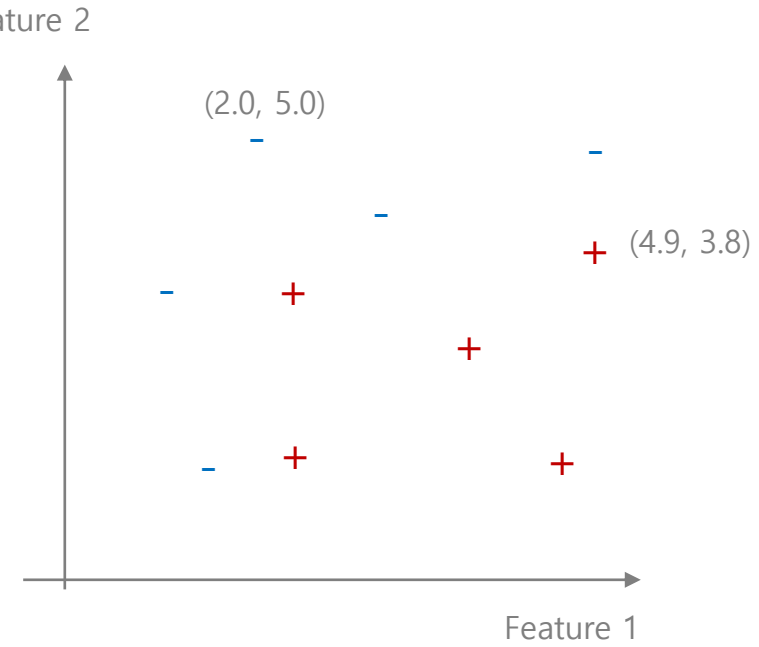
r_{k+1}	x	y
-0.88	0	0.5
-1.18	0.5	0.2
-1.08	0.7	0.3
-0.78	0.9	0.6
1.27	3	5
1.47	3.2	5.2
1.37	3.5	5.1
0.97	5.2	3.3
-0.23	5.5	2.1
-0.13	5.6	2.2
-0.33	6	2
-0.43	6.2	1.9

Gradient Boosting

$$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$$
For m in $1:M$:
$$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$
train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$
$$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$$
$$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$$
Return $F_M(x)$

- Classification 은 $F_k(x_i) = logit_i = \pi_i$ 을 예측합니다
- NLL loss 를 이용하면 $L = -\sum y_i \log p_i$ 입니다.

Feature 1	Feature 2	Class
2.0	5.0	-
5.0	4.8	-
3.5	4.5	-
1.0	3.3	-
1.5	1.5	-
2.2	3.3	+
4.9	3.8	+
4.0	2.5	+
2.2	1.6	+
4.5	1.6	+



Gradient Boosting

$$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$$

For m in $1: M$:

$$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$$

$$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$$

Return $F_M(x)$

$$L = -\sum y_i \log p_i$$

$$= -\sum_i (y_i \log p_i + (1 - y_i) \log(1 - p_i))$$

$$= -\sum_i \left(y_i \log\left(\frac{1-p_i}{p_i}\right) - \log(1 - p_i)\right)$$

$$= \sum_i -y_i \pi_i + \log(1 + e^{\pi_i}),\ \pi_i = \log\frac{p_i}{1-p_i},\ p_i = \frac{e^{\pi_i}}{1+e^{\pi_i}}$$

$$\frac{dL}{d\pi_i} = \sum_i -y_i + \frac{e^{\pi_i}}{1+e^{\pi_i}}$$

$$\rightarrow \sum_i -y_i + \frac{e^{\pi_i}}{1+e^{\pi_i}} = \sum_i -y_i + p_i$$

$$\rightarrow \sum_i -y_i + p_i = 0$$

$$\rightarrow p_i = \frac{1}{n}\sum y_i = \gamma_0$$

Gradient Boosting

$$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$$

For m in $1:M$:

$$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

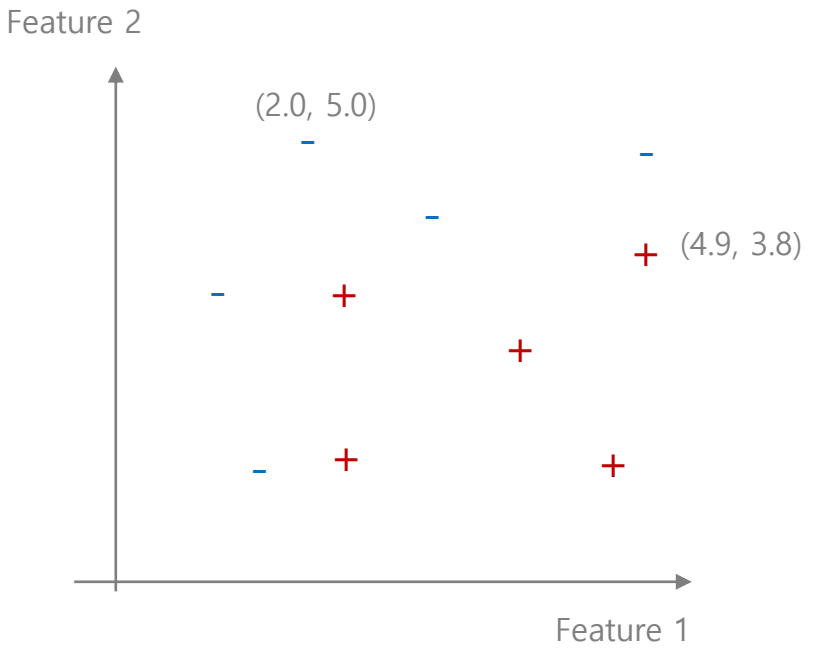
$$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$$

$$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$$

Return $F_M(x)$

- $\gamma_0 = \log\frac{0.5}{0.5} = 0$

Feature 1	Feature 2	Class	γ_0
2.0	5.0	-	0
5.0	4.8	-	0
3.5	4.5	-	0
1.0	3.3	-	0
1.5	1.5	-	0
2.2	3.3	+	0
4.9	3.8	+	0
4.0	2.5	+	0
2.2	1.6	+	0
4.5	1.6	+	0



Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- $p_i - y_i$ 를 pseudo residual, $r_{k,i}$ 로 이용합니다.
  - Softmax regression 에서도 residual 이 정의되지 않았습니다.

Feature 1	Feature 2	Class	γ_0	$r_{k,i}$
2.0	5.0	-	0	-0.5
5.0	4.8	-	0	-0.5
3.5	4.5	-	0	-0.5
1.0	3.3	-	0	-0.5
1.5	1.5	-	0	-0.5
2.2	3.3	+	0	0.5
4.9	3.8	+	0	0.5
4.0	2.5	+	0	0.5
2.2	1.6	+	0	0.5
4.5	1.6	+	0	0.5

Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

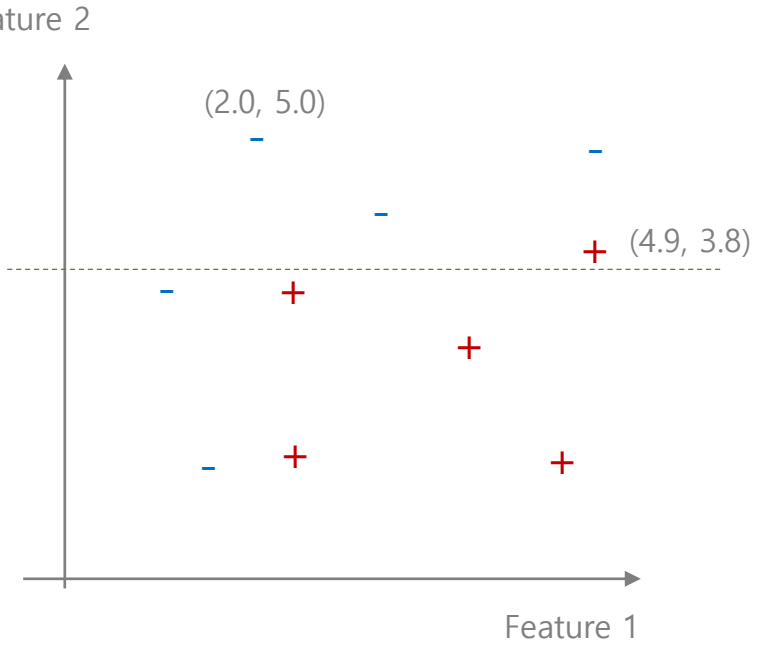
$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- $\{(x_i, r_{k,i})\}$ 를 f_1 으로 학습합니다.
  - 녹색선을 경계면으로 학습한 경우를 가정합니다.

Feature 1	Feature 2	Class	γ_0	$r_{k,i}$
2.0	5.0	-	0	-0.5
5.0	4.8	-	0	-0.5
3.5	4.5	-	0	-0.5
1.0	3.3	-	0	-0.5
1.5	1.5	-	0	-0.5
2.2	3.3	+	0	0.5
4.9	3.8	+	0	0.5
4.0	2.5	+	0	0.5
2.2	1.6	+	0	0.5
4.5	1.6	+	0	0.5



Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

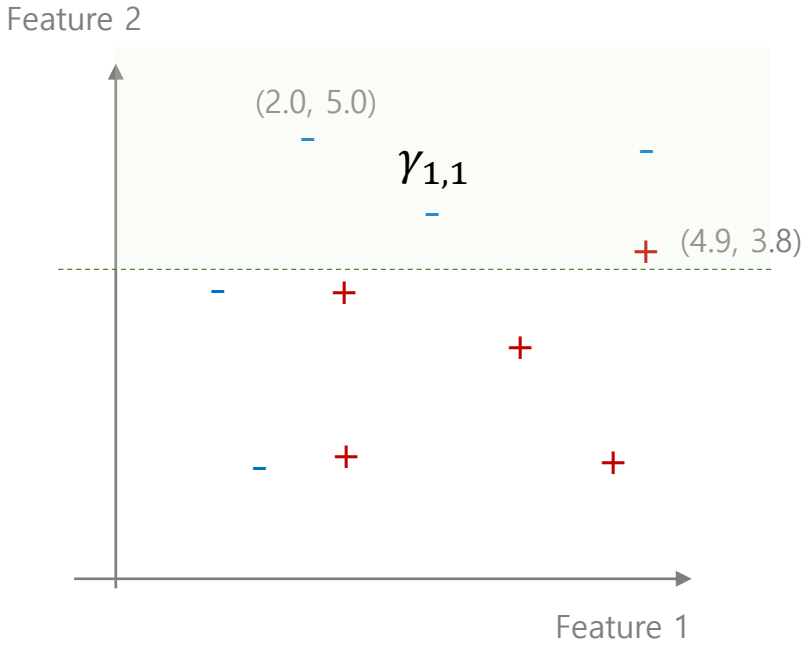
$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- 한 영역 $R_{k,j}$ 에 대한 pseudo residual 의 예측값 $\gamma_{k,j}$ 를 계산해야 합니다. 하지만 loss 식이 복잡하여 수식으로 γ 를 계산할 수 없습니다.

$$\arg\min_{\gamma} \sum_{x_i \in R_{k,j}} L(y_i, F_{k-1}(x_i) + \gamma)$$

$$L(y_i, F_{k-1}(x_i) + \gamma) = -y_i(\pi_{k-1,i} + \gamma) + \log(1 + e^{\pi_{k-1,i} + \gamma})$$



Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

- Tayler expansion 을 이용하여 loss 의 근사식을 계산합니다.
  Loss 식 대신 근사식으로부터 최적의 γ 를 계산합니다.

$$L(y_i, F_{k-1}(x_i) + \gamma) = \sum -y_i(\pi_{k-1,i} + \gamma) + \log(1 + e^{\pi_{k-1,i} + \gamma})$$

$$\cong L(y_i, \pi) + \gamma \frac{dL}{d\pi} + \frac{\gamma^2}{2!}\frac{d^2L}{d\pi^2}$$

$$\frac{dL}{d\gamma} \cong \frac{d}{d\gamma}\left(L(y_i, \pi) + \gamma \frac{dL}{d\pi} + \frac{\gamma^2}{2!}\frac{d^2L}{d\pi^2}\right) = \frac{dL}{d\pi} + \gamma \frac{d^2L}{d\pi^2} = 0$$

$$\rightarrow \gamma^* = -\frac{dL}{d\pi} \Big/ \frac{d^2L}{d\pi^2}$$

Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

Return $F_M(x)$

$$\gamma^* = -\frac{dL}{d\pi} / \frac{d^2L}{d\pi^2}$$

$$\frac{dL}{d\pi} = \sum_i -y_i + \frac{e^{\pi_i}}{1+e^{\pi_i}} = \sum_i -y_i + p_i \rightarrow G_{k,j}$$

$$\frac{d^2L}{d\pi^2} = \frac{d}{d\pi}\sum_i -y_i + \frac{e^{\pi_i}}{1+e^{\pi_i}}$$

$$= \sum_i \frac{e^{\pi_i}}{1+e^{\pi_i}} - \frac{e^{\pi_i} \times e^{\pi_i}}{(1+e^{\pi_i})^2}$$

$$= \sum_i \frac{e^{2\pi_i} + e^{\pi_i} - e^{2\pi_i}}{(1+e^{\pi_i})^2} = \sum_i \frac{e^{\pi_i}}{(1+e^{\pi_i})^2} = \sum_i \frac{1}{1+e^{\pi_i}} \times \frac{e^{\pi_i}}{1+e^{\pi_i}} = \sum_i p_i(1-p_i) \rightarrow H_{k,j}$$

Gradient Boosting

$$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$$

For m in $1{:}M$:

$$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$$

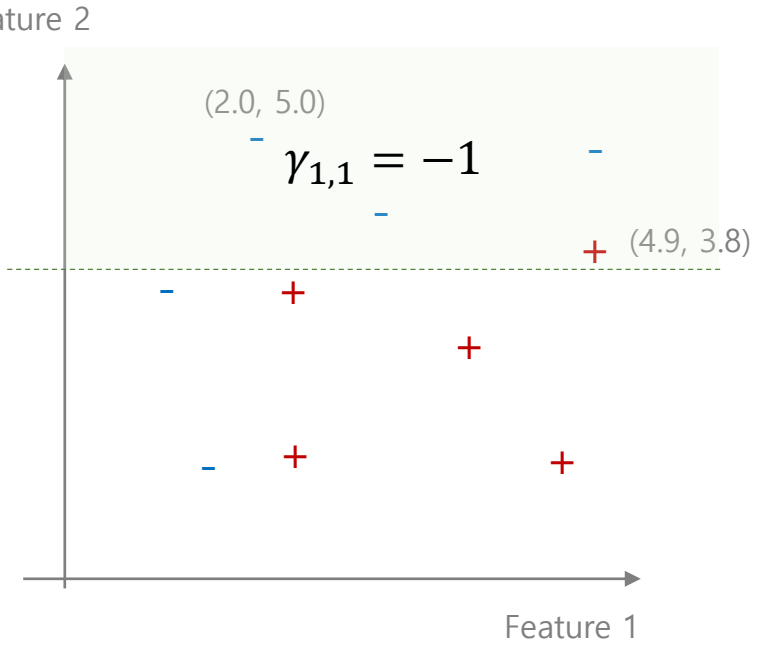
$$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$$

Return $F_M(x)$

$$\gamma^* = -\frac{dL}{d\pi} \Big/ \frac{d^2L}{d\pi^2} = -\frac{G_{k,j}}{H_{k,j}} = -\frac{0.5 - 0.5 - 0.5 - 0.5}{(0.5 \times (1 - 0.5)) \times 4}$$

$$G_{k,j} = \sum_i -y_i + p_{k-1,i}$$

$$H_{k,j} = \sum_i p_{k-1,i} \times (1 - p_{k-1,i})$$



Gradient Boosting

$F_0(x) = \arg\min_{\gamma} \sum_i L(y_i, \gamma)$

For m in $1:M$:

$r_{i,m} = -\left[\frac{dL(y_i, F(x_i))}{d\,F(x_i)}\right]_{F(x)=F_{m-1}(x)}$

train $f_m(x)$ with $\{(x_i, r_{i,m})\}_{i=1}^{n}$

$\gamma_{i,m} = \arg\min_{\gamma} \sum_{x_i \in R_{j,m}} L(y_i, F_{m-1}(x_i) + \gamma)$

$F_m(x) = F_{m-1}(x) + \nu \sum_{x_i \in R_{m,j}} \gamma_{m,j} I(x_i \in R_{m,j})$

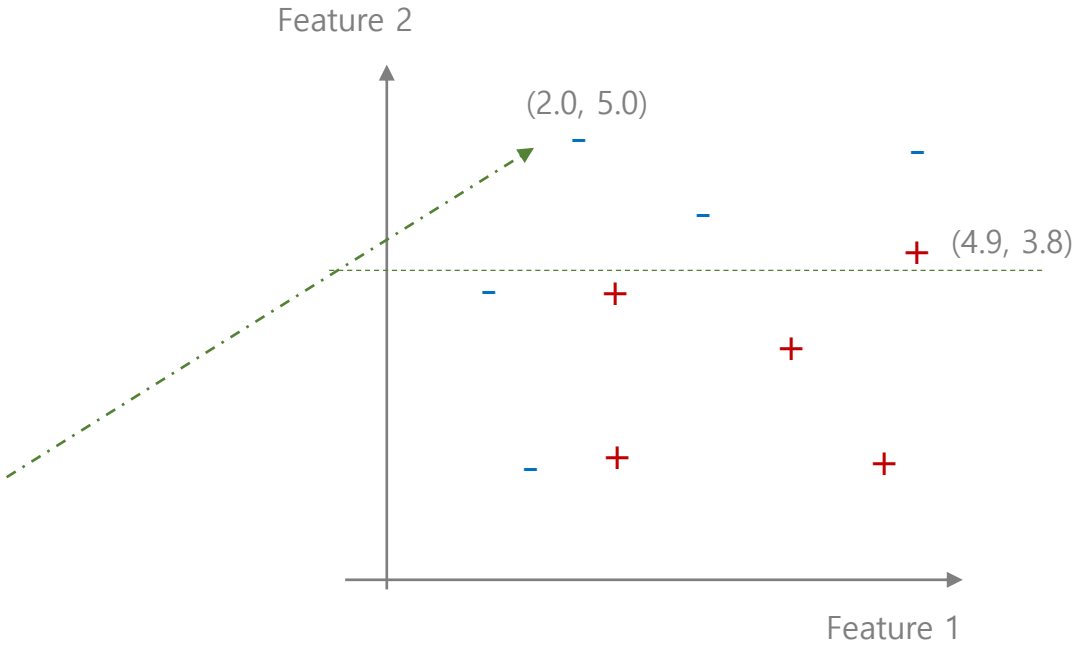
Return $F_M(x)$

- Learning rate 를 곱하여 예측값을 업데이트 합니다.
  - $\nu = 0.5$ 라면 $\pi = F_1((2.0, 5.0)) = -0.5$,

    $p_{1,i} = \frac{\exp(-0.5)}{1+\exp(-0.5)} = 0.378 < 0.5$

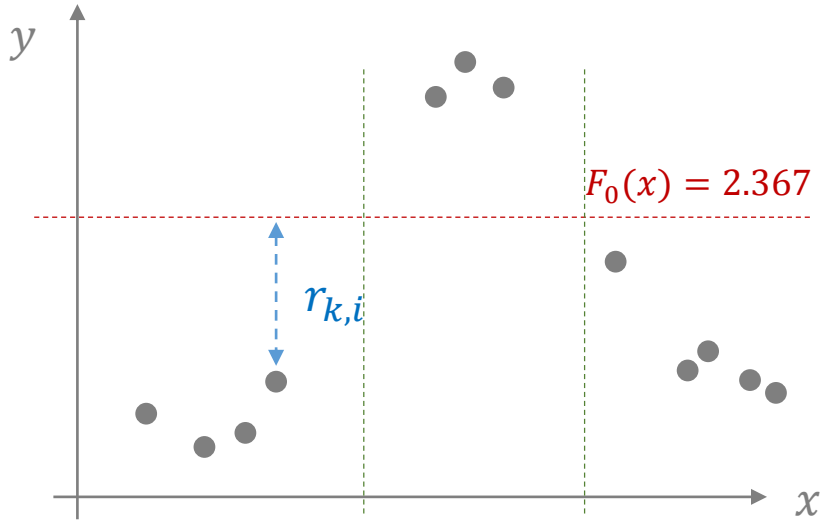
  - $p_{0,i}$ 보다 정확한 예측을 합니다.

$F_1(x_i) = f_0(x_i) + \nu f_1(x_i) = 0 + \nu(-1) = -\nu$



Gradient Boosting

- L2 loss 를 이용하는 gradient boosting regression 에서는 다음처럼 γ 가 정해집니다.
  - Decision tree 의 한 leaf 의 residuals 의 방향이 모두 같으면 업데이트가 많이 됩니다.



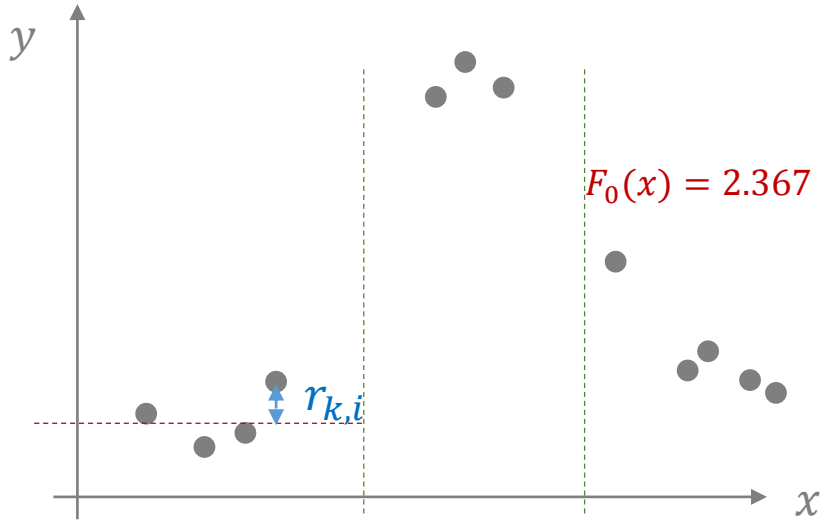
$$g_{k,j}(x_i) = \left(y_i - f_k(x_i)\right)$$

$$h_{k,j}(x_i) = -1$$

$$\gamma^*_{k,j} = -\frac{G_{k,j}}{H_{k,j}} = \frac{\text{sum of residual}}{n \text{ samples}}$$

Gradient Boosting

- L2 loss 를 이용하는 gradient boosting regression 에서는 다음처럼 γ 가 정해집니다.
  - 한 leaf 내 residual 의 방향이 다르면 그 합이 0 에 가까워 지기 때문에 업데이트가 적게 됩니다.



$$g_{k,j}(x_i) = \left(y_i - f_k(x_i)\right)$$

$$h_{k,j}(x_i) = -1$$

$$\gamma^*_{k,j} = -\frac{G_{k,j}}{H_{k,j}} = \frac{\text{sum of residual}}{n \text{ samples}}$$

Stochastic Gradient Boosting

- Pseudo residual 의 값이 작은 데이터 포인트들은 학습이 충분히 된 점들일 가능성이 높습니다. 이러한 점들보다 pseudo residual 이 큰 점들을 우선적으로 학습하는 것이 효율적입니다.
- Stochastic gradient boosting 은 $|r_{k,i}|$ 의 크기를 sample weight 로 이용하여 resampling 을 한 뒤, 이를 학습하는 모델을 추가합니다.

XGBoost

- Extreme gradient boosting 는 gradient boosting 의 학습 효율과 성능을 개선한 알고리즘입니다.
- 이후에 제안된 boosting models 인 LightGBM, CatBoost 역시 gradient boosting 의 단점들을 해결하는 과정의 결과물입니다.

Chen, T., He, T., Benesty, M., Khotilovich, V., & Tang, Y. (2015). Xgboost: extreme gradient boosting. *R package version 0.4-2*, 1-4.

XGBoost

- Gradient boosting trees 는 기본 decision tree 를 이용합니다. 하지만 XGBoost 는 tree 의 분기 기준부터 변경합니다.
- XGBoost 의 loss 에는 과적합을 방지하기 위한 regularization term 이 포함되어 있습니다.

$$L\big(y, F_k(x)\big) = \sum_i l\big(y_i, F_{k-1}(x_i) + f_k(x_i)\big) + \Omega(f_k)$$

$$= \sum_i l\big(y_i, F_{k-1}(x_i) + f_k(x_i)\big) + \frac{\lambda}{2}\sum_j w_j^2 + \gamma T_k$$

XGBoost

$$L = \sum_i l\left(y_i, F_{k-1}(x_i) + f_k(x_i)\right) + \frac{\lambda}{2}\sum_j w_j^2 + \gamma T_k$$

$$\cong \sum_i \left[l\left(y_i, F_{k-1}(x_i)\right) + g_i f_k(x_i) + \frac{1}{2} h_i f_k(x_i)^2\right] + \frac{\lambda}{2}\sum_j w_j^2 + \gamma T_k$$

상수항

$$\cong \sum_i \left[g_i f_k(x_i) + \frac{1}{2} h_i f_k(x_i)^2\right] + \frac{\lambda}{2}\sum_j w_j^2 + \gamma T_k$$

$$= \sum_i \left[g_i w_j + \frac{1}{2} h_i w_j^2\right] + \frac{\lambda}{2}\sum_j w_j^2 + \gamma T_k$$

$f_k(x_i) = w_{k,j}$

$g_i = \frac{d}{df_k} l\left(y_i, f_k(x_i)\right)$

$h_i = \frac{d}{df_k^2} l\left(y_i, f_k(x_i)\right)$

XGBoost

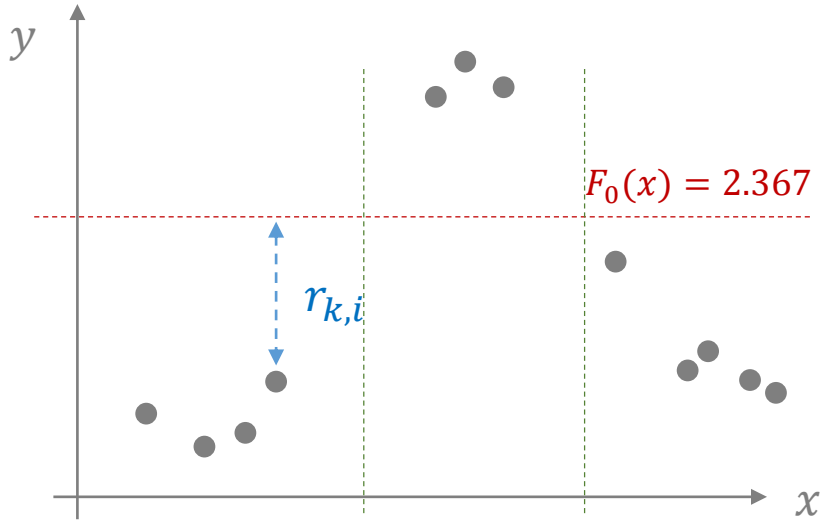
$$L = \sum_i \left[g_i w_j + \frac{1}{2} h_i w_j^2 \right] + \frac{\lambda}{2} \sum_j w_j^2 + \gamma T_k$$

$$\frac{dL}{dw_j} = \sum_{x_i \in R_{k,j}} g_i + h_i w_j + \lambda w_j$$

$$\frac{dL}{dw_j} = 0 \rightarrow w_j^* = -\frac{\sum_{x_i \in R_{k,j}} g_i}{\sum_{x_i \in R_{k,j}} h_i + \lambda} = -\frac{G_{k,j}}{H_{k,j} + \lambda}$$

XGBoost

- L2 loss 를 이용하는 regression 에서는 다음처럼 γ 가 정해집니다.
  - Leaf node 에 samples 의 개수가 작으면 $\lambda > 0$ 에 의하여 분모가 빨리 커집니다.
  - 작은 크기의 leaf 의 예측값을 잘 믿지 않는다는 의미입니다. 이를 통하여 variance 가 커지는 과적합을 방지할 수 있습니다.



$$g_{k,j}(x_i) = \left(y_i - f_k(x_i)\right)$$

$$h_{k,j}(x_i) = -1$$

$$\gamma^*_{k,j} = -\frac{G_{k,j}}{H_{k,j}+\lambda} = \frac{\text{sum of residual}}{n\ \text{samples}+\lambda}$$

XGBoost

- 그 외에도 효율적인 학습을 위한 방법들이 제안되었습니다.
  - Sparsity-aware 방법은 dummy variable 이나 텍스트처럼 0 인 값들이 많은 데이터에 대하여 split point 를 빠르게 탐색하는 방법입니다.
  - 마디를 분기할 때 0 인 값들을 각각 left-side / right-side 로 보냈을 때의 information gain 을 계산한 다음, 최적의 split point 를 선택합니다. 데이터에 0 이 많은 경우 IG 를 계산해야 하는 점들의 개수가 줄어들어 속도가 빨라집니다 (x10)

XGBoost

- XGBoost 는 논문이 제안된 이후에도 다양한 기능들을 추가하여 업데이트 되고 있으며, 다양한 문제에서 좋은 성능을 보여주고 있습니다.

- Boosting models 들은 특정 지역의 데이터들이 비슷한 target variable y 를 지닐 때, 해당 지역을 나누는 능력이 좋습니다.
  - 영상이나 텍스트처럼 input data vector 가 정보력이 좋지 않은 경우에는 패턴 별로 특정 지역을 나누기 이전에 vector representation 을 우선 학습해야 합니다. 딥러닝계 모델들이 주로 좋은 성능을 보이는 문제들입니다.
  - Dummy variables 처리 등 feature engineering 과정을 거친 데이터는 지역별로 패턴이 존재할 가능성이 높습니다. 경계선긋기만으로 동질적인 데이터가 잘 구분되는 경우에는 boosting models 이 좋은 성능을 보입니다.