Boosting

Jinwon Sohn

jwsohn@yonsei.ac.kr

Reference

- Greedy Function Approximation, Friedman, 2001.
- The Elements of Statistical Learning, Springer.

Numerical optimization

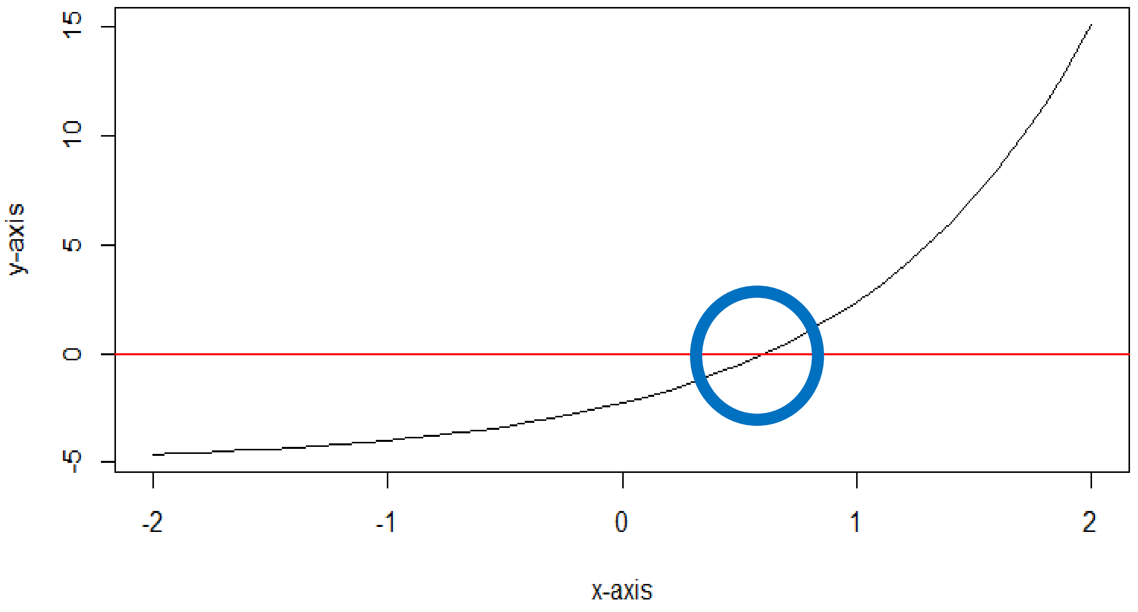
- In many problems, we can not obtain the closed solution for given equations.
- For example, the coefficients of logistic regression.
- Ex)Solving, $\arg\max_{\beta_0,\beta_1} \prod_{i=1}^{N} \left(\frac{1}{1+e^{-(\beta_0+\beta_1 x_i)}}\right)^{r_i} \left(1-\frac{1}{1+e^{-(\beta_0+\beta_1 x_i)}}\right)^{1-r_i}$

No closed solutions exist!

Newton's method

- $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$

- $x_{n+1} = x_n - \frac{f(x_n)(x_n - x_{n-1})}{f(x_n) - f(x_{n-1})}$



Newton’s method

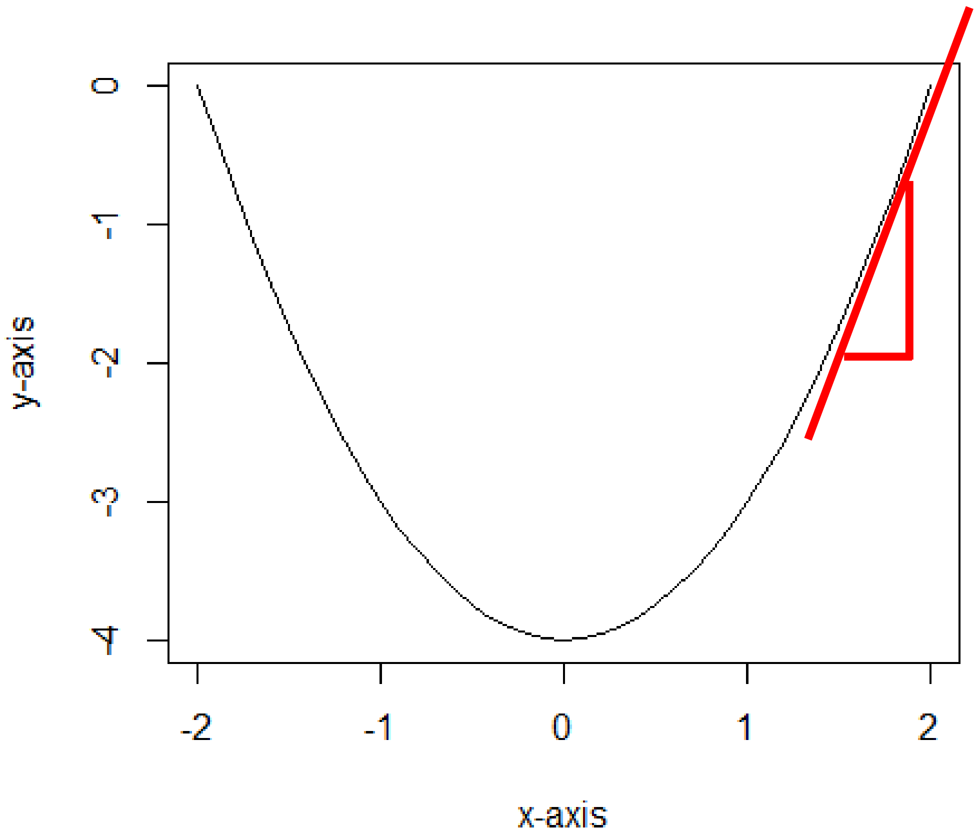
- The solution might be local optimum which is the usual problem in numerical optimization.
  > By using multiple initial points, we can alleviate the problem.

- Other mathematical properties.. are skipped in this class.

Gradient descent (or ascent)

- $x_{n+1} = x_n - \alpha \frac{df}{dx_n}$
- $\theta_{n+1} = \theta_n - \alpha \frac{dL}{d\theta_n}$

- $x_{n+1} = x_n - \alpha f'(x_n)$
- $x_{n+1} = x_n - \alpha \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$



Line search gradient descent

- $x_{n+1} = x_n - \alpha_n \frac{df}{dx_n}$
- $\theta_{n+1} = \theta_n - \alpha_n \frac{dL}{d\theta_n}$

where α_n is α to minimize a function $L\left(x_n - \alpha \frac{df}{dx_n}\right)$.

Line search gradient descent

- $\theta_{n+1} = \theta_n - \alpha_n \frac{dL}{d\theta_n}$
- $n \longrightarrow \infty$, $\theta_n \longrightarrow \theta^*$(Solution!)
- $\theta^* = \sum_{n=0}^{\infty}\left(\theta_0 - \alpha_n \frac{dL}{d\theta_n}\right)$ where θ_0 is initial value set normally as 0.
- $\theta^* \approx \sum_{n=0}^{N}\left(-\alpha_n \frac{dL}{d\theta_n}\right)$ for some large N.
- The term, $-\alpha_n \frac{dL}{d\theta_n}$, is called 'boost' or 'step'.
  We will see this later.

Boosting

- An ensemble model composed of the sum of weak models.
- A stump or simple linear regression model are kinds of weak model.
- In other word, the week model has the high bias.
- Let’s start!

Boosting

- So,

$$G_H(x) = \sum_{h=1}^{H} w_h g_h(x;\theta_h)$$

The learner, $G_H(x)$, averages the **weak learners**, $g_h(x)$, with the weights w_h. In order to improve the performance of $F(x)$, $\Theta = \{\theta_1, \dots, \theta_H\}$ and $\omega = \{w_1, \dots, w_H\}$ have to be optimized.

Boosting

- Namely, the following loss function should be minimized with respect to Θ, ω simultaneously.

$$\arg\min_{\Theta,\omega} \sum_{i=1}^{N} L(y_i, G_H(x_i))$$

$$= \arg\min_{\Theta,\omega} \sum_{i=1}^{N} L\left(y_i, \sum_{h=1}^{H} w_h g_h(x; \theta_h) \right)$$

Boosting

- However, it requires intensive computation. Imagine How much time we need, in order to find the optimal values in $|\Theta| \times |\omega|$-spaces.

- A simple alternative can approximate the loss function with relatively lighter computation. We call "*Forward Stagewise Additive Modeling*".

Forward Stagewise Additive Modeling

- Optimize the parameters **one by one by moving in the forward direction**.
- Let $f_h(x) = f_{h-1}(x) + wg(x;\theta) \Longrightarrow G_H(x) = \sum_{h=1}^{H} f_h(x)$
- Then, $f_h(x)$ will be decided by optimizing w and θ in terms of

$$\arg\min_{w,\theta} \sum_{i=1}^{N} L(y_i, f_h(x_i)), \qquad h = 1,2,\ldots,H.$$

$$= \arg\min_{w,\theta} \sum_{i=1}^{N} L(y_i, f_{h-1}(x_i) + wg(x_i;\theta)), \quad h = 1,2,\ldots,H.$$

Forward Stagewise Additive Modeling

- Let $L\big(y, f(x)\big) = (y - f(x))^2$, squared loss.
- Let $f_0(x) = 0$, then
- $f_1(x)$ can be obtained by solving

$$\arg\min_{w,\theta} \sum_{i=1}^{N} L(y_i, f_0(x_i) + wg(x_i;\theta))$$

$=$

$$\arg\min_{w,\theta} \sum_{i=1}^{N} (y_i - wg(x_i;\theta))^2$$

Forward Stagewise Additive Modeling

- $f_2(x)$ can be obtained by solving

$$\arg\min_{w,\theta} \sum_{i=1}^{N} L(y_i, f_1(x_i) + wg(x_i;\theta))$$

$$\arg\min_{w,\theta} \sum_{i=1}^{N} (y_i - f_1(x_i) - wg(x_i;\theta))^2$$

$$\arg\min_{w,\theta} \sum_{i=1}^{N} \left(r_{1,i} - wg(x_i;\theta)\right)^2$$

residual

- Denote these optimized parameters as w_1 and θ_1.

Forward Stagewise Additive Modeling

- $f_3(x)$ can be obtained by solving

$$\arg\min_{w,\theta} \sum_{i=1}^{N} L(y_i, f_2(x_i) + wg(x_i;\theta))$$

$$\arg\min_{w,\theta} \sum_{i=1}^{N} (y_i - f_2(x_i) - wg(x_i;\theta))^2$$

$$\arg\min_{w,\theta} \sum_{i=1}^{N} (y_i - f_1(x_i) - w_1 g(x_i;\theta_1) - wg(x_i;\theta))^2$$

Forward Stagewise Additive Modeling

$$\underset{w,\theta}{\arg\min} \sum_{i=1}^{N} \left(r_{1,i} - w_1 g(x_i;\theta_1) - w g(x_i;\theta)\right)^2$$

$$\underset{w,\theta}{\arg\min} \sum_{i=1}^{N} \left(r_{2,i} - w g(x_i;\theta)\right)^2$$

residual

- Denote these optimized parameters as w_2 and θ_2.

Forward Stagewise Additive Modeling

- Thus, $f_H(x)$ can be obtained by solving

$$\underset{w,\theta}{\arg\min} \sum_{i=1}^{N} L(y_i, f_{H-1}(x_i) + wg(x_i;\theta))$$

$$\underset{w,\theta}{\arg\min} \sum_{i=1}^{N} \left(r_{H-1,i} - wg(x_i;\theta)\right)^2$$

residual

- $f_H(x) = f_{H-1}(x) + w_H g(x_i;\theta_H)$.

- Thus, the final model is $\sum_{h=1}^{H} f_h(x) = \sum_{h=1}^{H} w_H g(x_i;\theta_H)$.

AdaBoost

- Set $L(y, f(x)) = e^{(-yf(x))}$. 'exponential loss'.
- Set the base estimator $g_h = g(x; \theta_h)$ be a 'stump', decision tree with one depth. 'Boost'
- Weights of observations are considered. 'Adaptive!'
- If applying *FSAM* to above setting,
- then you can obtain following algorithm.
- Please, refer to the page 344 in ESL for the proof.

AdaBoost

1. Initialize the observation weight $w_i = \frac{1}{N}, \quad i = 1, \ldots, N$.

2. For $h = 1$ to H:

   (a) Fit a classifier g_h to the training data using weights w_i.

   (b) Compute

$$\mathrm{e}rr_h = \frac{\sum_i w_i I(y_i \neq g(x_i; \theta_h))}{\sum_i w_i}.$$

   (c) Compute $\alpha_h = \log\big((1 - \mathrm{e}rr_h)/\mathrm{e}rr_h\big)$.

   (d) Set $w_i \leftarrow w_i e^{\alpha_h I(y_i \neq g(x_i; \theta_h))}, \quad i = 1, \ldots, N$.

3. Output $G(x) = sign[\sum_h \alpha_h g(x; \theta_h)]$

Gradient Boosting

- Greedy Function Approximation, Friedman, 2001.

- History…
  The Evolution of Boosting Algorithms - From Machine Learning to Statistical Modelling, Mayr, 2014.

Gradient Boosting

- Consider previous steepest descent with the line search algorithm.

- $\theta_{n+1} = \theta_n - \alpha_n \frac{dL}{d\theta_n}$
- $\theta^* \approx \sum_{n=0}^{N} \left(-\alpha_n \frac{dL}{d\theta_n} \right) = \sum_{n=0}^{N} p_n$ for some large N.
- The increment, $p_n = -\alpha_n \frac{dL}{d\theta_n}$, is called 'boost' or 'step'.

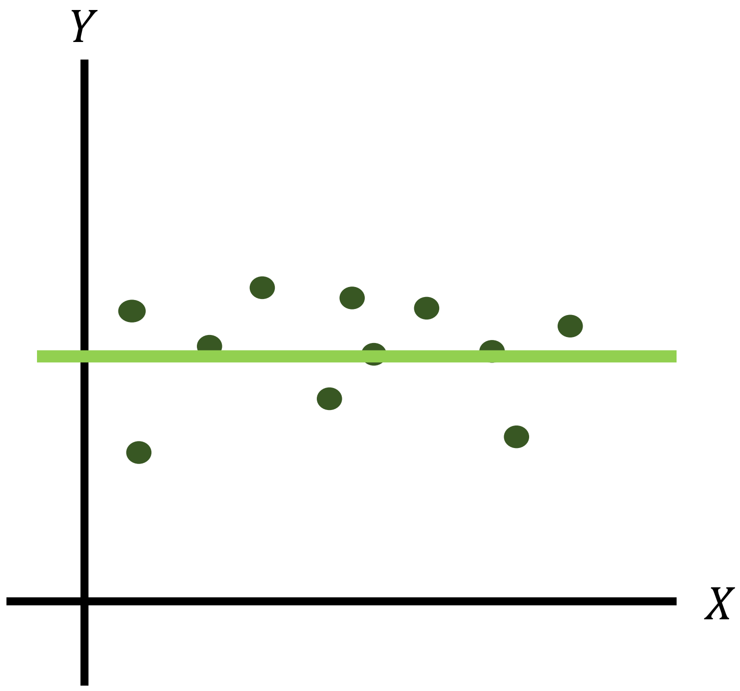
Gradient Boosting

- We can regard **a function or classifier $\boldsymbol{F}(\boldsymbol{x})$ as a parameter**, and optimize it numerically. This means that numerical optimization is used to estimate nonparametric function.

- $F_{h+1} = F_h - \alpha_h \frac{dL}{dF_h}$
- $F^* \approx \sum_{h=0}^{H} \left(-\alpha_h \frac{dL}{dF_h} \right) = \sum_{h=0}^{H} f_h$ for some large H.
- The classifier can be composed of many increment functions!

Gradient Boosting

- For example, we are interested in estimating μ from data



- We can estimate from $\bar{X}$ as well as the numerical optimization previously handled

$$\mu_{n\_+1} = \mu_n - \alpha_n \frac{dL}{d\mu_n}, \qquad L = \sum_i (y_i - \mu)^2$$

- Finally, expand your imagination that each point, X_i, has its own $\mu(X_i)$.

Gradient Boosting

- Let $F_{m-1} = \sum_{h=0}^{m-1} f_h$.

- Then $F_m = F_{m-1} + f_m$
  The increment f_m consists of $-\alpha_m$ and $\frac{dL}{dF_{m-1}}$ where $-\frac{dL}{dF_{m-1}}$ is the steepest gradient and α_m is found via the line search algorithm.

$$\alpha_m = \arg\min_{\alpha} L\left(y, F_{m-1}(x) - \alpha \frac{dL}{dF_{m-1}}\right)$$

Gradient Boosting

- The convergence steps or sequences of GB implicitly have the concept of Forward Stagewise Additive modeling.

- Since the negative gradients for each step are defined only at the specific data points, we have to construct models to generate the negative gradients.

- For m-step,

$$g_m(x;\theta_m) \approx -\frac{dL}{dF_{m-1}}$$

Gradient Boosting

- In addition, the α_m come to be a weight parameter for the m-th model.

- So, the $F^*(x)$ can be approximated as,

$$F^* \approx \sum_{h=1}^{H} \alpha_h g_h(x; \theta_h)$$

Gradient Boosting

- For example,
  set L_2 loss function for a GB model.
  set initial guess $f_0(x) = 0$ or $f_0(x) = \bar{y}$.

- Remember that the gradient of L_2 loss function is

$$\frac{dL_2}{dF(x)} = 2\big(y - F(x)\big).$$

Gradient Boosting

- $F_1(x)$ can be obtained by solving

Step 1 : $$\theta_1 = arg\,\min_{\theta} \sum_{i=1}^{N} \left(-\frac{dL_2}{dF_0} - g(x_i;\theta) \right)^2$$

$$\theta_1 = arg\,\min_{\theta} \sum_{i=1}^{N} (-2(y_i - F_0(x_i)) - g(x_i;\theta))^2$$

Negative gradient

Step 2 : $$\alpha_1 = arg\,\min_{\alpha} \sum_{i=1}^{N} (y_i - F_0(x_i) - \alpha g(x_i;\theta_1))^2$$

Step 3 : $$F_1(x) = F_0(x) + \alpha_1 g(x_i;\theta_1)$$

Gradient Boosting

- $F_2(x)$ can be obtained by solving

Step 1 : $$\theta_2 = arg\min_{\theta} \sum_{i=1}^{N} \left(-\frac{dL_2}{dF_1} - g(x_i;\theta) \right)^2$$

$$\theta_2 = arg\min_{\theta} \sum_{i=1}^{N} (-2(y_i - F_1(x_i)) - g(x_i;\theta))^2$$

$$\theta_2 = arg\min_{\theta} \sum_{i=1}^{N} (-2(y_i - F_0(x_i) - \alpha_1 g(x_i;\theta_1)) - g(x_i;\theta))^2$$

Fitting on residuals!?

Step 2 : $$\alpha_2 = arg\min_{\alpha} \sum_{i=1}^{N} (y_i - F_1(x_i) - \alpha g(x_i;\theta_2))^2$$

Step 3 : $$F_2(x) = F_1(x) + \alpha_2 g(x_i;\theta_2)$$

Gradient Boosting

- $F_H(x)$ can be obtained by solving

Step 1 : $\theta_H = arg\min_{\theta} \sum_{i=1}^{N} \left(-\frac{dL_2}{dF_{H-1}} - g(x_i; \theta) \right)^2$

$$\theta_H = arg\min_{\theta} \sum_{i=1}^{N} (-2(y_i - F_{H-1}(x_i)) - g(x_i; \theta))^2$$

$$\theta_H = arg\min_{\theta} \sum_{i=1}^{N} (-2(y_i - F_{H-2}(x_i) - \alpha_{H-1} g(x_i; \theta_{H-1})) - g(x_i; \theta))^2$$

Fitting on residuals!?

Step 2 : $\alpha_H = arg\min_{\alpha} \sum_{i=1}^{N} (y_i - F_{H-1}(x_i) - \alpha g(x_i; \theta_H))^2$

Step 3 : $F_H(x) = F_{H-1}(x) + \alpha_H g(x_i; \theta_H)$

Gradient Boosting

- What is the significance of Gradient Boosting?
- Seemingly, it is equal to basic boosting in L_2 loss function.
- However, If we use L_1, or Huber loss function, GB has a more general applications.
- That’s why we call the boost(or step) as the negative gradient, the generalized or pseudo residuals.

Gradient Boosting

- L_1 loss function and its derivative.

$$L_1 = \sum_{i=1}^{N} |y_i - F(x_i)|, \qquad \frac{dL_1}{dF(x)} = sign(y - F(x))$$

This loss function is robust to outliers.

- Huber loss function and its derivative
  please, refer to https://en.wikipedia.org/wiki/Huber_loss

- You can customize your own loss function.

Gradient Boosting

- You should specify the size of tree for each increment.
- Heuristically, $4 \leq d \leq 8$.
- The depth, d, reflect the order of an interaction!
- If a tree has 3 depth, then the tree includes not only main effect, but also up to third interactions.

Gradient Boosting

- In classification problem, a classifier learns in continuous spaces. For examples, soft-max mapping,

$$p_k(x) = \frac{e^{F_k(x)}}{\sum_l e^{F_l(x)}} \iff F_k(x) = \log p_k(x) - \frac{1}{K} \sum_{k=1}^{K} y_k \log p_k(x)$$

- With the cross-entropy function, and its derivative,

$$L(\{y_k, F_k(x)\}_1^K) = -\sum_{k=1}^{K} y_k \log p_k(x)$$

$$\frac{dL(\{y_k, F_k(x)\}_1^K)}{dF_k(x)} = p_k(x) - y_k$$

Extended Gradient Boosting

- Stochastic Gradient Booting
  > Subsampled data without replacement is used to fit a increment function. This do lighter computations.
  > A bit of resistance on overfitting

- Regularized Gradient Booting
  > Charge penalties onto the number of trees.

$$F_m(x) = F_{m-1}(x) + \nu_{penalty}\alpha_m g(x_i; \theta_m)$$

XGBoost

- Gradient boosting needs burdensome computations which makes it difficult to apply GB models into Big data.

- T. Chen wrote in his paper how to improve computation ability by adjusting algorithms to build a tree.

- Refer to "https://gentlej90.tistory.com/87"

XGBoost

- XGBoost supports not only subsampling rows and columns, but also the regularized version of GB.
- Related to this, there are many parameters on XGBoost.
- **Unfortunately, XGBoost is very sensitive to tuning hyperparameters.**
- https://xgboost.readthedocs.io/en/latest/tutorials/param_tuning.html

LightGBM

- Structural difference.

- For more details, refer to https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

CatBoost

- Feature engineering for categorical predictors.
- For more details, refer to https://towardsdatascience.com/catboost-vs-light-gbm-vs-xgboost-5f93620723db

Partial Dependence Plot

- Boosting models have high prediction performances, generally more than models based on bagging, and, moreover, more than neural network based models if data has structural form.

- But, a Gradient Boosting machine are inherently a black box model.

- However, via PDP, we can interpret the machine indirectly.

Partial Dependence Plot

- Let data X be $X = [X_s, X_c]$.
- Our interest is to understand how X_s has a effect on responses, marginalizing X_c.
- We call this as partial dependence of $f(X)$ on X_s.

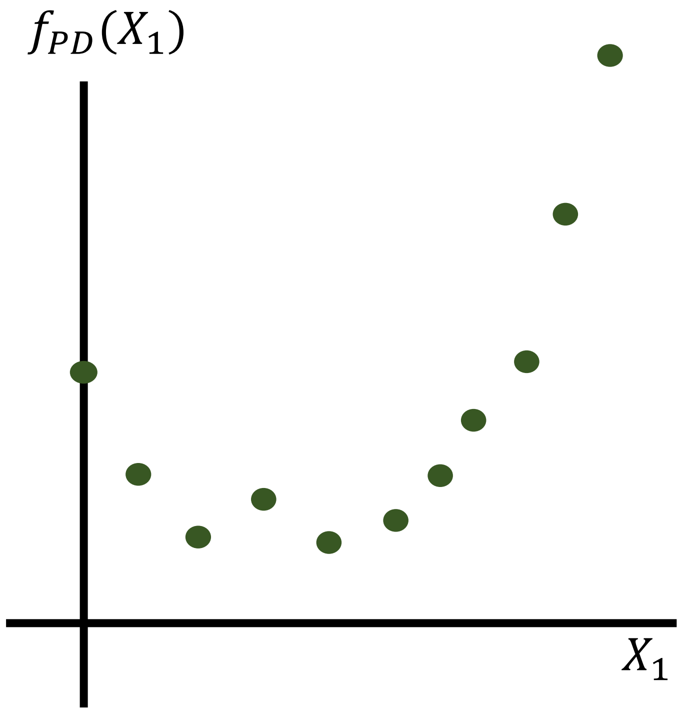
$$f(X_s) = E_{X_c}[f(X_s, X_c)] \approx \frac{1}{N}\sum_{i=1}^{N} f\left(X_s, x_{i_C}\right)$$

- Above equation implies that after taking X_c into account, the effects of X_s onto $f(X)$ will be represented.

Partial Dependence Plot

- Let 3 predictors, X_1, X_2, and X_3, exist.
- X_1 : 0~10, X_2 : M, F, and X_3 : -10.4 ~ 22.5.
- We want to draw PDP of X_1.

X_1	X_2	X_3	$f(X)$	$f_{PD}(X_1)$
$X_1 = 0$	M	-10.4	120	$f_{PD}(X_1) = 134$
	M	-9.5	132	
	....	....	....	
	F	22.5	150	
$X_1 = 1$	M	-10.4	130	$f_{PD}(X_1) = 93$
	M	-9.5	100	
	....	....	....	
	F	22.5	50	
....	....	....	....	....
$X_1 = 10$	M	-10.4	220	$f_{PD}(X_1) = 234$
	M	-9.5	232	
	....	....	....	
	F	22.5	250	



Partial Dependence Plot

- Assume that X_c and X_s have pure additive effects.

$$f(X) = h(X_s) + h(X_c), \qquad E_{X_c}[f(X)] = \int \big(h(X_s) + h(X_c)\big) f(X_c) dX_c$$

$$E_{X_c}[f(X)] = h(X_s) + const$$

- Assume that X_c and X_s have pure multiplicative effects.

$$f(X) = h(X_s)h(X_c), \qquad E_{X_c}[f(X)] = \int \big(h(X_s)h(X_c)\big) f(X_c) dX_c$$

$$E_{X_c}[f(X)] = h(X_s) \times const$$

Partial Dependence Plot

- PDP is different to simply ignoring X_c.
- Ignoring X_c means that

$$f(X_s) = E_{X_c}[f(X_s, X_c)|X_s]$$

$$f(X) = h(X_s) + h(X_c), \qquad E_{X_c}[f(X)] = \int \big(h(X_s) + h(X_c)\big) f(X_c|X_s) dX_c$$

$$E_{X_c}[f(X)] = h(X_s) + \int h(X_c) f(X_c|X_s) dX_c$$

$$f(X) = h(X_s)h(X_c), \qquad E_{X_c}[f(X)] = \int \big(h(X_s)h(X_c)\big) f(X_c|X_s) dX_c$$

$$E_{X_c}[f(X)] = h(X_s)\int h(X_c) f(X_c|X_s) dX_c$$

Partial Dependence Plot

- In case that X_s and X_c are independent,

$$E_{X_c}[f(X_s, X_c)] = E_{X_c}[f(X_s, X_c)|X_s].$$

- PDP is reasonable when the interactions between X_s and X_c are deficit.

- In other words, PDP might be unreliable if there are strong interactions.