QBUS6850

Lecture 8

Extreme Gradient Boosting

BUSINESS SCHOOL

QBUS6850 Team

THE UNIVERSITY OF SYDNEY

THE UNIVERSITY OF SYDNEY

- Topics covered
  - Review: Bagging and Boosting
  - Brief History of Gradient Boosting
  - Gradient Boosting for Regression
  - Gradient Boosting for Classification
  - Relationship between Adaboost and Gradient Boosting
  - XGBoost package

References

- References
  - The classic paper initialising the idea: J. Friedman, Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.
  - The key paper https://arxiv.org/pdf/1603.02754.pdf

THE UNIVERSITY OF SYDNEY

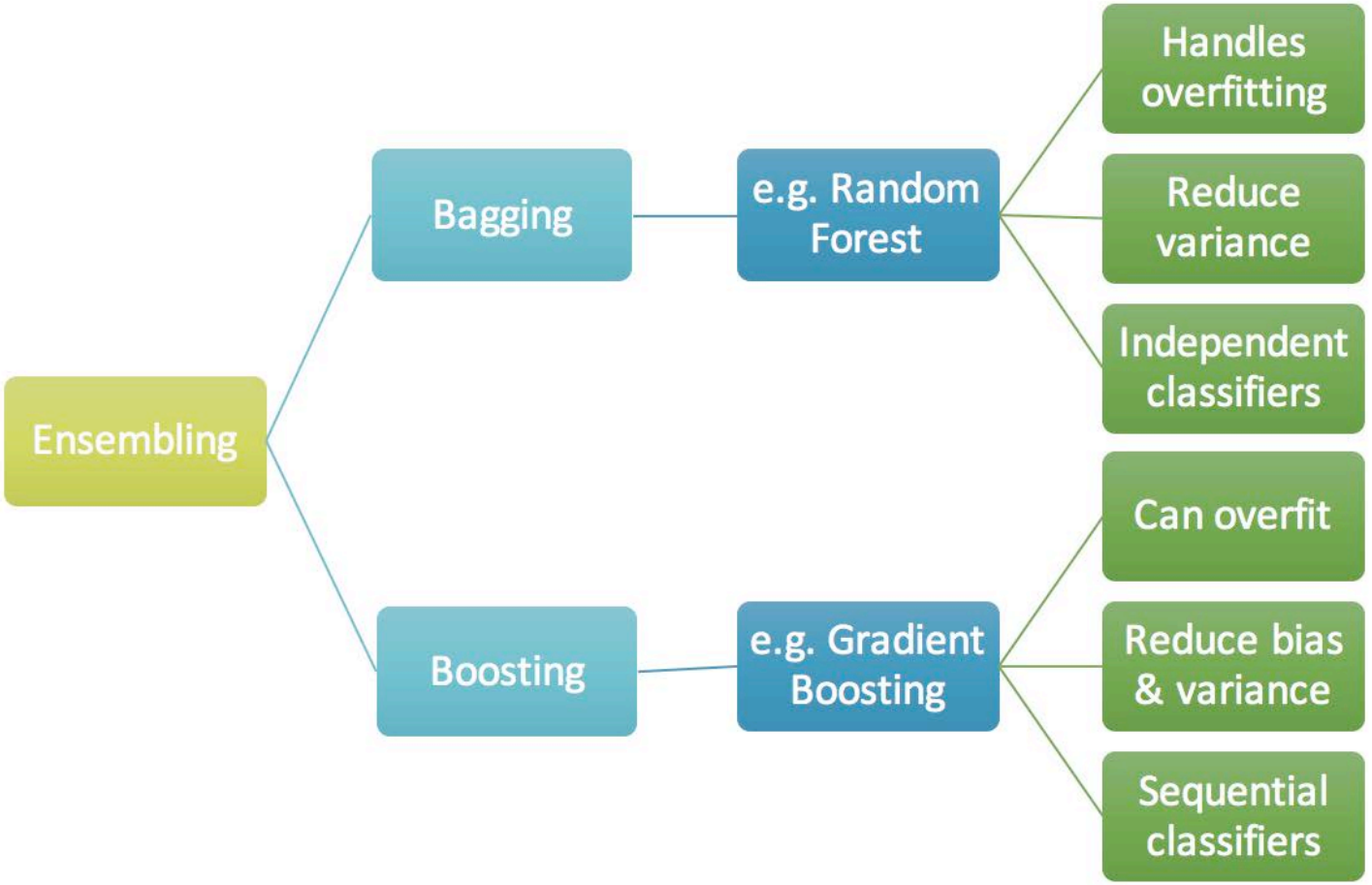
Learning Objectives

- Understand what Gradient Boosting is and the goals of using it
- Understand how XGBoost is implemented
- Understand why XGBoost must be apart of your machine learning experience
- Understand how you can start using XGBoost on your own machine learning projects
- Understand how to use the XGBoost package(s) in Python

Review: Bagging and Boosting

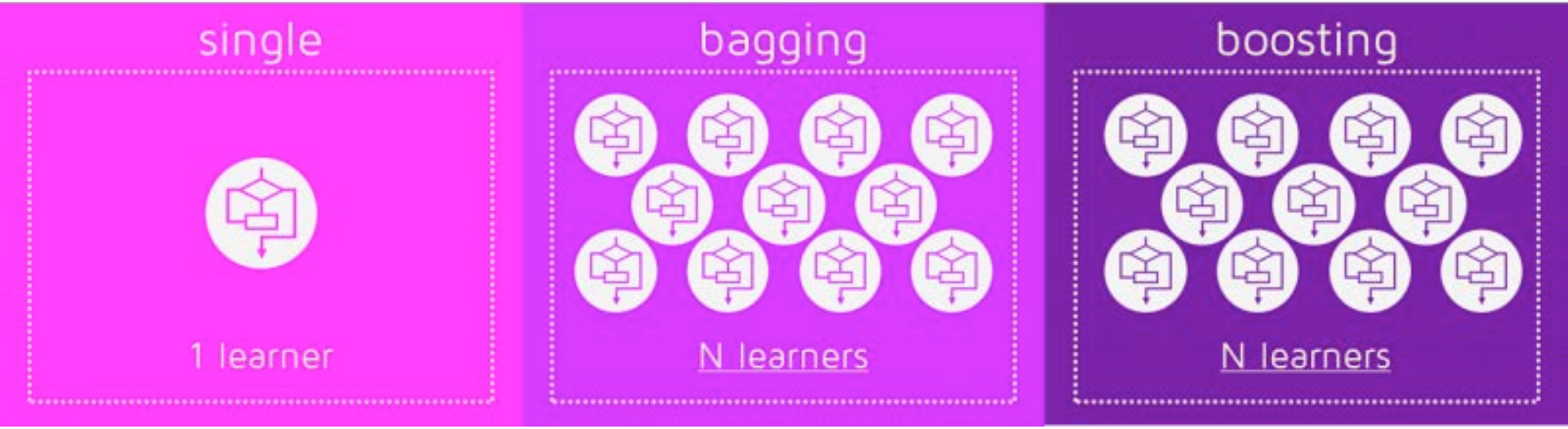
Review: Bagging and Boosting

https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d



Review: Bagging and Boosting

https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

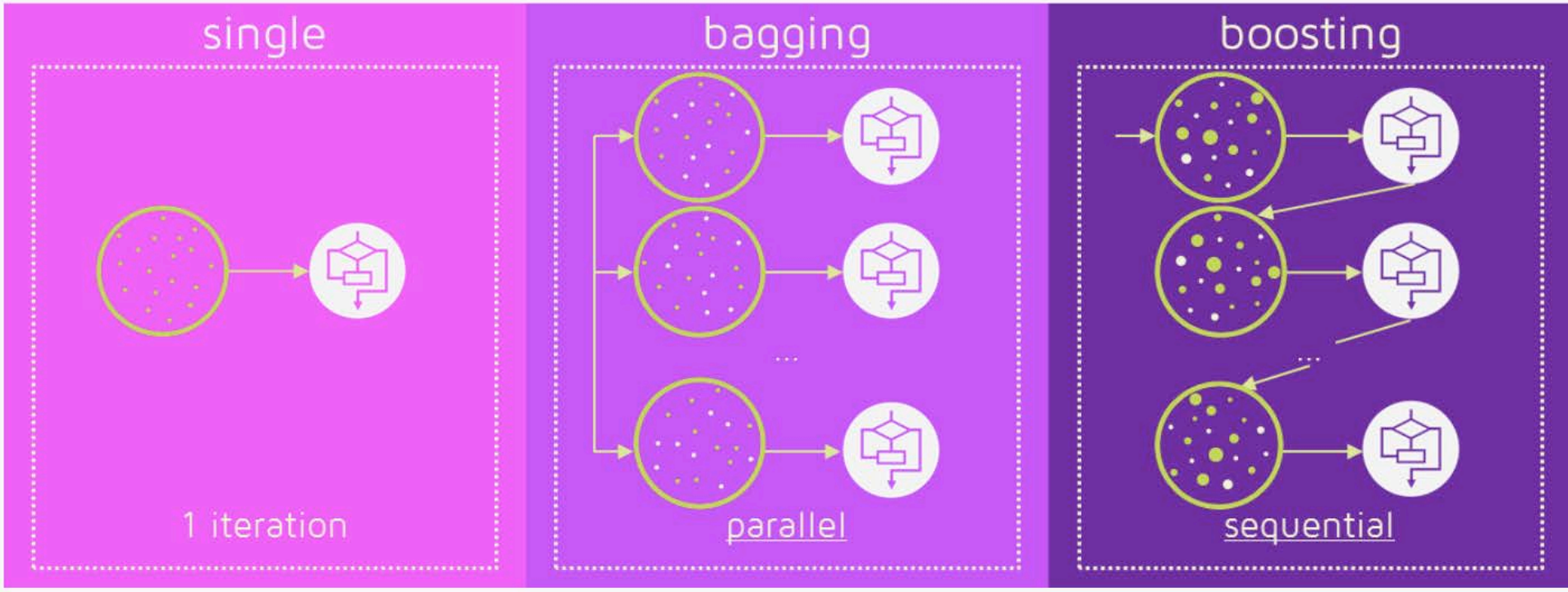


Review: Bagging and Boosting

https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

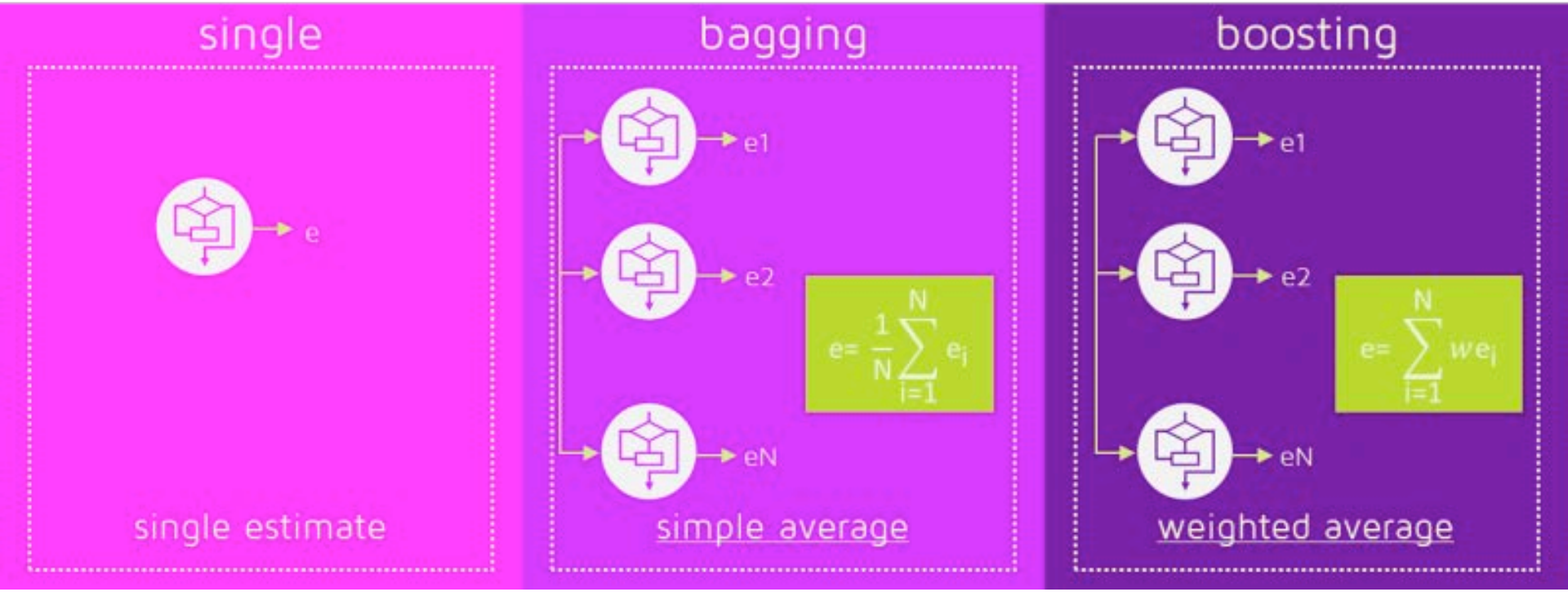
Review: Bagging and Boosting

https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/



Review: Bagging and Boosting

https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/



Review: Bagging and Boosting

https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/

Review: Bagging and Boosting

Similarities	Differences
Both are ensemble methods to get N learners from 1 learner…	… but, while they are built independently for Bagging, Boosting tries to add new models that do well where previous models fail.
Both generate several training data sets by random sampling…	… but only Boosting determines weights for the data to tip the scales in favor of the most difficult cases.
Both make the final decision by averaging the N learners (or taking the majority of them)…	… but it is an equally weighted average for Bagging and a weighted average for Boosting, more weight to those with better performance on training data.
Both are good at reducing variance and provide higher stability…	… but only Boosting tries to reduce bias. On the other hand, Bagging may solve the over-fitting problem, while Boosting can increase it.

THE UNIVERSITY OF SYDNEY

Gradient Boosting

Gradient Boosting

- **Gradient boosting (GB)** is a machine learning technique for regression and classification problems
- Gradient Boosting = Gradient Descent + Boosting
- It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees
- In each ensemble step, a weak learner is used to compensate the shortcomings of existing weak learners
- The gradient information is used to define /identify the shortcomings for improvement
- The idea comes from Adaboost by Friedman et al

Gradient Boosting vs AdaBoost

- AdaBoost is the original boosting algorithm, but wasn't that efficient. Gradient Boosting was developed as a generalization of AdaBoost

- It has been observed by researchers that what AdaBoost was doing was a gradient search in decision tree space against a particular cost function (implicitly used)

- Gradient Boosting can be applied to other cost functions, some of which were more suitable to the domain than the one that was implicitly used in AdaBoost

Brief History

- Adaboost: invented in 1996 [Freund et al., 1996, Freund and Schapire, 1997]
- Interpret Adaboost as a gradient descent algorithm under a special loss function [Breiman et al., 1998, Breiman, 1999]
- Propose Gradient Boosting for any loss functions [Friedman et al., 2000, Friedman, 2001]

Freund, Y., Schapire, R. E., et al. (1996). Experiments with a new boosting algorithm. In *ICML*, volume 96, pages 148-156.
Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119-139.
Breiman, L. (1999). Prediction games and arcing algorithms. *Neural Computation*, 11(7):1493{1517.
Breiman, L. et al. (1998). Arcing classier (with discussion and a rejoinder by the author). *The Annals of Statistics*, 26(3):801{849.
Friedman, J., Hastie, T., and Tibshirani, R. (2000). Special invited paper. additive logistic regression: A statistical view of boosting. *Annals of Statistics*, pages 337-374.
Friedman, J. H. (2001). Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189-1232.

THE UNIVERSITY OF SYDNEY

Gradient Boosting Regression

GB Regression

- Given a data set $\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \ \ldots, (\mathbf{x}_N, t_N)\}$ of N data. We wish to fit a model which minimizes the squared loss over the dataset
- Suppose we already have a model $F(\mathbf{x})(= f(\mathbf{x}, \boldsymbol{\beta})$ note: in new notation we ignore the parameters), which is not perfect (or we are not happy with it)
- For example, $F(\mathbf{x}_1) = 0.8$ while $t_1 = 0.80001$; $F(\mathbf{x}_2) = 1.5$ while $t_2 = 5.4$; …
- How can we improve the current model $F(\mathbf{x})$?
- Rules to apply
  - It is not a good idea to throw away $F(\mathbf{x})$ given that we have spent resource to get it and it has done some good things e.g. for case 1
  - We will look at the modified model in term of additive way $F(\mathbf{x}) + h(\mathbf{x})$

GB Regression: Simple Solution

- That means the new model shall satisfy

$$\begin{aligned} F(\mathbf{x}_1) + h(\mathbf{x}_1) &= t_1 \\ F(\mathbf{x}_2) + h(\mathbf{x}_2) &= t_2 \\ &\vdots \\ F(\mathbf{x}_N) + h(\mathbf{x}_N) &= t_N \end{aligned} \quad \longrightarrow \quad \begin{aligned} h(\mathbf{x}_1) &= t_1 - F(\mathbf{x}_1) \\ h(\mathbf{x}_2) &= t_2 - F(\mathbf{x}_2) \\ &\vdots \\ h(\mathbf{x}_N) &= t_N - F(\mathbf{x}_N) \end{aligned}$$

We know these values

- Can we do this by another regression?

$$\mathbf{x}_n \xrightarrow{\text{Find a } h} t_n - F(\mathbf{x}_n)$$

- Input is $\mathbf{x}_n$ with target $t_n - F(\mathbf{x}_n)$ $(n = 1, 2, \dots, N)$. $h(\mathbf{x})$ can be any (simpler) model, such as a tree stump.
- If $F(\mathbf{x}) + h(\mathbf{x})$ is not good enough, consider it as the new $F(\mathbf{x})$ and find a new $F(\mathbf{x}) + h(\mathbf{x})$……
- Job is done

How to relate to Gradient Descent

- $t_n - F(\mathbf{x}_n)$ is called residuals
- Consider the loss function $L\big(t, F(\mathbf{x})\big) = \frac{1}{2}(t - F(\mathbf{x}))^2$
- We want to minimize the overall loss

$$L = \sum_{n=1}^{N} L(t_n, F(\mathbf{x}_n))$$

by adjusting $F(\mathbf{x}_1)$, $F(\mathbf{x}_2)$, …, $F(\mathbf{x}_N)$ (not the parameters inside F).

- If we treat $F(\mathbf{x}_n)$ as parameters, then

$$\frac{\partial L}{\partial F(\mathbf{x}_n)} = \frac{\partial \sum_{n=1}^{N} L(t_n, F(\mathbf{x}_n))}{\partial F(\mathbf{x}_n)} = \frac{\partial L(t_n, F(\mathbf{x}_n))}{\partial F(\mathbf{x}_n)} = F(\mathbf{x}_n) - t_n$$

- Hence the residuals are actively the negative gradients

$$t_n - F(\mathbf{x}_n) = -\frac{\partial L}{\partial F(\mathbf{x}_n)}$$

How to relate to Gradient Descent

- How did we update the improver $h(\mathbf{x})$?

$$F(\mathbf{x}_n) \coloneqq F(\mathbf{x}_n) + h(\mathbf{x}_n) \approx F(\mathbf{x}_n) + t_n - F(\mathbf{x}_n) = F(\mathbf{x}_n) - \frac{\partial L}{\partial F(\mathbf{x}_n)}$$

New F

$$F(\mathbf{x}_n) :\approx F(\mathbf{x}_n) - 1\frac{\partial L}{\partial F(\mathbf{x}_n)}$$

Gradient Descent

- For regression with squared loss

residual	⟷	Negative gradient
Fit h to residual	⟷	Fit h to negative gradient
Update F based on residual	⟷	Update F based on negative gradient

Summary

Regression with squared loss

- Start with an initial model, for example, a constant model

$$F^{(0)}(\mathbf{x}) \equiv \frac{\sum_{n=1}^{N} t_n}{N}$$

- Do the following steps until some criteria are satisfied
  - Calculate negative gradient

$$-g(\mathbf{x}_n) = -\frac{\partial L\left(t_n, F^{(m-1)}(\mathbf{x}_n)\right)}{\partial F^{(m-1)}(\mathbf{x}_n)} = t_n - F^{(m-1)}(\mathbf{x}_n)$$

  - Fit a (simple) model (e.g., decision tree) $h^{(m)}(\mathbf{x})$ to negative gradients $-g(\mathbf{x}_n)$ for $n = 1, 2, \ldots, N$
  - Construct the new overall model

$$F^{(m)}(\mathbf{x}) \coloneqq F^{(m-1)}(\mathbf{x}) + \rho h^{(m)}(\mathbf{x}) \text{ where } \rho = 1$$

Summary

Regression with any differentiable loss function

- Start with an initial model, for example, a constant model

$$F^{(0)}(\mathbf{x}) \equiv \frac{\sum_{n=1}^{N} t_n}{N}$$

- Do the following steps until some criteria are satisfied
  - Calculate negative gradient at iteration m

$$-g(\mathbf{x}_n) := -\frac{\partial L\left(t_n, F^{(m-1)}(\mathbf{x}_n)\right)}{\partial F^{(m-1)}(\mathbf{x}_n)}$$

  - Fit a (simple) model (e.g., decision tree) $h^{(m)}(\mathbf{x})$ to negative gradients $-g(\mathbf{x}_n)$ for $n = 1, 2, \ldots, N$
  - Construct the new overall model

$$F^{(m)}(\mathbf{x}) := F^{(m-1)}(\mathbf{x}) + \rho h^{(m)}(\mathbf{x}) \text{ where } \rho = 1$$

THE UNIVERSITY OF SYDNEY

Gradient Boosting Classification

Gradient Boosting (Classification)

- GB for regression with any loss function.
- Classification is also a regression problem with a specially defined loss function, such as the cross entropy over the data
- Review: what is the cross entropy loss function? Consider the case of three classes "1", "2", "3" for single datum $\mathbf{x}_n$ with three models $F_1(\mathbf{x})$, $F_2(\mathbf{x})$, $F_3(\mathbf{x})$

$$L(t_n, F(\mathbf{x}_n)) = -\left[t_{n1}\log\left(P_1(\mathbf{x}_n)\right) + t_{n2}\log\left(P_2(\mathbf{x}_n)\right) + t_{n3}\log\left(P_2(\mathbf{x}_n)\right)\right]$$

where

$$P_j(\mathbf{x}_n) = \frac{e^{F_j(\mathbf{x}_n)}}{e^{F_1(\mathbf{x}_n)} + e^{F_2(\mathbf{x}_n)} + e^{F_3(\mathbf{x}_n)}} \quad j = 1,2,3$$

- What is $\mathbf{t} = (t_{n1}, t_{n2}, t_{n3})$?

$\mathbf{x}_n$ is class 1	$\mathbf{t} = (t_{n1}, t_{n2}, t_{n3}) = (1,0,0)$
$\mathbf{x}_n$ is class 2	$\mathbf{t} = (t_{n1}, t_{n2}, t_{n3}) = (0,1,0)$
$\mathbf{x}_n$ is class 3	$\mathbf{t} = (t_{n1}, t_{n2}, t_{n3}) = (0,0,1)$

Gradient Boosting (Classification)

- It can be proved that, for $j = 1,2,3$, [exercise!]

$$-\boldsymbol{g}_j(\mathbf{x}_n) = -\frac{\partial L(t_n, F(\mathbf{x}_n))}{\partial F_j(\mathbf{x}_n)} = t_{nj} - \frac{e^{F_j(\mathbf{x}_n)}}{e^{F_1(\mathbf{x}_n)} + e^{F_2(\mathbf{x}_n)} + e^{F_3(\mathbf{x}_n)}} = t_{nj} - P_j(\mathbf{x}_n)$$

- Fit (simple) models (e.g., shallow decision trees) $h_j(\mathbf{x})$ to negative gradients $-g_j(\mathbf{x}_n)$ for $n = 1, 2, \dots, N$ and $j = 1,2,3$

- There are three models $F_1(\mathbf{x}_n)$, $F_2(\mathbf{x}_n)$, $F_3(\mathbf{x}_n)$ to be updated

$$F_1^{new}(\mathbf{x}_n) \coloneqq F_1(\mathbf{x}_n) + \rho h_1(\mathbf{x}_n)$$
$$F_2^{new}(\mathbf{x}_n) \coloneqq F_2(\mathbf{x}_n) + \rho h_2(\mathbf{x}_n)$$
$$F_3^{new}(\mathbf{x}_n) \coloneqq F_3(\mathbf{x}_n) + \rho h_3(\mathbf{x}_n)$$

- We can start from initial $F_1(\mathbf{x}) = F_2(\mathbf{x}) = F_3(\mathbf{x}) = 0$. See `Lecture08_Example01.py`

Algorithm Summary

Classification with the cross-entropy loss function

- Start with initial models, for example, constants

$$F_j^{(0)}(\mathbf{x}) \equiv 0, \qquad j = 1,2,\dots,K$$

- Do the following steps until some criteria are satisfied
  - Calculate negative gradient at iteration m

$$-g_j(\mathbf{x}_n) = -\frac{\partial L\left(\mathbf{t}_n, F^{(m-1)}(\mathbf{x}_n)\right)}{\partial F_j^{(m-1)}(\mathbf{x}_n)} = t_{nj} - P_j^{(m-1)}(\mathbf{x}_n, \boldsymbol{\beta})$$

  - Fit (simple) model (e.g., decision tree) $h_j^{(m)}(\mathbf{x})$ to negative gradients $-g_j(\mathbf{x}_n)$ for $n = 1,2,\dots,N$ and all $j = 1,2,\dots,K$
  - Construct the new overall model

$$F_j^{(m)}(\mathbf{x}) := F_j^{(m-1)}(\mathbf{x}) + \rho h_j^{(m)}(\mathbf{x}) \text{ where } \rho = 1$$

Classification Example (K=3)

	x1	x2	x3	x4	t		
0	5.4	3.4	1.7	0.2	1	0	0
1	6.5	3	5.2	2	0	0	1
2	4.3	3	1.1	0.1	1	0	0
3	6.6	3	4.4	1.4	0	1	0
4	4.9	3.1	1.5	0.1	1	0	0
5	5	3.2	1.2	0.2	1	0	0
6	5.5	2.5	4	1.3	0	1	0
7	5.7	2.9	4.2	1.3	0	1	0
8	5.1	2.5	3	1.1	0	1	0
9	4.8	3.4	1.6	0.2	1	0	0
10	6.7	3.1	5.6	2.4	0	0	1
11	6.5	3	5.8	2.2	0	0	1

Part of Iris dataset: https://archive.ics.uci.edu/ml/datasets/iris three classes Setosa, Versicolour, Virginica

Classification Example (K=3)

Step 1: Initial Models and Initial Probability (How from F to P?)

	$F_1^{(0)}$	$F_2^{(0)}$	$F_3^{(0)}$
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0

$$P_j(\mathbf{x}_n) = \frac{e^{F_j(\mathbf{x}_n)}}{e^{F_1(\mathbf{x}_n)} + e^{F_2(\mathbf{x}_n)} + e^{F_3(\mathbf{x}_n)}}$$

$$P_3(\mathbf{x}_5) = \frac{e^{F_3(\mathbf{x}_5)}}{e^{F_1(\mathbf{x}_5)} + e^{F_2(\mathbf{x}_5)} + e^{F_3(\mathbf{x}_5)}}$$

$$= \frac{e^0}{e^0+e^0+e^0} = \frac{1}{1+1+1}$$

	$P_1^{(0)}$	$P_2^{(0)}$	$P_3^{(0)}$
0	0.333333	0.333333	0.333333
1	0.333333	0.333333	0.333333
2	0.333333	0.333333	0.333333
3	0.333333	0.333333	0.333333
4	0.333333	0.333333	0.333333
5	0.333333	0.333333	0.333333
6	0.333333	0.333333	0.333333
7	0.333333	0.333333	0.333333
8	0.333333	0.333333	0.333333
9	0.333333	0.333333	0.333333
10	0.333333	0.333333	0.333333
11	0.333333	0.333333	0.333333

Classification Example (K=3)

Step 2: Calculate the negative gradient

t			$P_1^{(0)}$	$P_2^{(0)}$	$P_3^{(0)}$
1	0	0	0.333333	0.333333	0.333333
0	0	1	0.333333	0.333333	0.333333
1	0	0	0.333333	0.333333	0.333333
0	1	0	0.333333	0.333333	0.333333
1	0	0	0.333333	0.333333	0.333333
1	0	0	0.333333	0.333333	0.333333
0	1	0	0.333333	0.333333	0.333333
0	1	0	0.333333	0.333333	0.333333
0	1	0	0.333333	0.333333	0.333333
1	0	0	0.333333	0.333333	0.333333
0	0	1	0.333333	0.333333	0.333333
0	0	1	0.333333	0.333333	0.333333

$-g_j(\mathbf{x}_n) = t_{nj} - P_j(\mathbf{x}_n)$

Negative gradients

$-g_1$	$-g_2$	$-g_3$
0.666667	-0.3333…	-0.333333
-0.3333…	-0.3333…	0.666667
0.666667	-0.3333…	-0.333333
-0.3333…	0.666667	-0.333333
0.666667	-0.3333…	-0.333333
0.666667	-0.3333…	-0.333333
-0.3333…	0.666667	-0.333333
-0.3333…	0.666667	-0.333333
-0.3333…	0.666667	-0.333333
0.666667	-0.3333…	-0.333333
-0.3333…	-0.3333…	0.666667
-0.3333…	-0.3333…	0.666667

Classification Example (K=3)

Step 3a: Modeling from $\mathbf{X}$ to $-g_1$ [Model 1]

$\mathbf{X}$

0	5.4	3.4	1.7	0.2
1	6.5	3	5.2	2
2	4.3	3	1.1	0.1
3	6.6	3	4.4	1.4
4	4.9	3.1	1.5	0.1
5	5	3.2	1.2	0.2
6	5.5	2.5	4	1.3
7	5.7	2.9	4.2	1.3
8	5.1	2.5	3	1.1
9	4.8	3.4	1.6	0.2
10	6.7	3.1	5.6	2.4
11	6.5	3	5.8	2.2

h1

$-g_1$: Negative gradient

0.666667	-0.3333…	-0.333333
-0.3333…	-0.3333…	0.666667
0.666667	-0.3333…	-0.333333
-0.3333…	0.666667	-0.333333
0.666667	-0.3333…	-0.333333
0.666667	-0.3333…	-0.333333
-0.3333…	0.666667	-0.333333
-0.3333…	0.666667	-0.333333
-0.3333…	0.666667	-0.333333
0.666667	-0.3333…	-0.333333
-0.3333…	-0.3333…	0.666667
-0.3333…	-0.3333…	0.666667

Classification Example (K=3)

Build a decision tree $h_1(\mathbf{x})$ of depth 1 from $\boldsymbol{X}$ to $-g_1$: [Regression Problem!]

$$h_1^{(1)}(\mathbf{x}) = \begin{cases} 0.667 & \text{if } x_4 \leq 0.65 \\ -0.333 & \text{if } x_4 > 0.65 \end{cases}$$

0	5.4	3.4	1.7	0.2
1	6.5	3	5.2	2
2	4.3	3	1.1	0.1
3	6.6	3	4.4	1.4
4	4.9	3.1	1.5	0.1
5	5	3.2	1.2	0.2
6	5.5	2.5	4	1.3
7	5.7	2.9	4.2	1.3
8	5.1	2.5	3	1.1
9	4.8	3.4	1.6	0.2
10	6.7	3.1	5.6	2.4
11	6.5	3	5.8	2.2

$h_1(\mathbf{x})$ →

0.666667
-0.333333
0.666667
-0.333333
0.666667
0.666667
-0.333333
-0.333333
-0.333333
0.666667
-0.333333
-0.333333

Classification Example (K=3)

Step 3b: Modeling from $\mathbf{X}$ to $-g_2$ [Model 2]

$\mathbf{X}$

0	5.4	3.4	1.7	0.2
1	6.5	3	5.2	2
2	4.3	3	1.1	0.1
3	6.6	3	4.4	1.4
4	4.9	3.1	1.5	0.1
5	5	3.2	1.2	0.2
6	5.5	2.5	4	1.3
7	5.7	2.9	4.2	1.3
8	5.1	2.5	3	1.1
9	4.8	3.4	1.6	0.2
10	6.7	3.1	5.6	2.4
11	6.5	3	5.8	2.2

h2

$-g_2$: Negative gradient

0.666667	-0.3333…	-0.333333
-0.3333…	-0.3333…	0.666667
0.666667	-0.3333…	-0.333333
-0.3333…	0.666667	-0.333333
0.666667	-0.3333…	-0.333333
0.666667	-0.3333…	-0.333333
-0.3333…	0.666667	-0.333333
-0.3333…	0.666667	-0.333333
-0.3333…	0.666667	-0.333333
0.666667	-0.3333…	-0.333333
-0.3333…	-0.3333…	0.666667
-0.3333…	-0.3333…	0.666667

Classification Example (K=3)

Build a decision tree $h_2(\mathbf{x})$ of depth 1 from $\boldsymbol{X}$ to $-g_2$: [Regression Problem!]

$$h_2^{(1)}(\mathbf{x}) = \begin{cases} 0.667 & \text{if } x_2 \leq 2.95 \\ -0.222 & \text{if } x_2 > 2.95 \end{cases}$$

0	5.4	3.4	1.7	0.2
1	6.5	3	5.2	2
2	4.3	3	1.1	0.1
3	6.6	3	4.4	1.4
4	4.9	3.1	1.5	0.1
5	5	3.2	1.2	0.2
6	5.5	2.5	4	1.3
7	5.7	2.9	4.2	1.3
8	5.1	2.5	3	1.1
9	4.8	3.4	1.6	0.2
10	6.7	3.1	5.6	2.4
11	6.5	3	5.8	2.2

$h_2(\mathbf{x})$ →

-0.222222
-0.222222
-0.222222
-0.222222
-0.222222
-0.222222
0.666667
0.666667
0.666667
-0.222222
-0.222222
-0.222222

Classification Example (K=3)

Step 3c: Modeling from $\mathbf{X}$ to $-g_3$ [Model 3]

$\mathbf{X}$

0	5.4	3.4	1.7	0.2
1	6.5	3	5.2	2
2	4.3	3	1.1	0.1
3	6.6	3	4.4	1.4
4	4.9	3.1	1.5	0.1
5	5	3.2	1.2	0.2
6	5.5	2.5	4	1.3
7	5.7	2.9	4.2	1.3
8	5.1	2.5	3	1.1
9	4.8	3.4	1.6	0.2
10	6.7	3.1	5.6	2.4
11	6.5	3	5.8	2.2

h3 →

$-g_3$: Negative gradient

0.666667	-0.3333…	-0.333333
-0.3333…	-0.3333…	0.666667
0.666667	-0.3333…	-0.333333
-0.3333…	0.666667	-0.333333
0.666667	-0.3333…	-0.333333
0.666667	-0.3333…	-0.333333
-0.3333…	0.666667	-0.333333
-0.3333…	0.666667	-0.333333
-0.3333…	0.666667	-0.333333
0.666667	-0.3333…	-0.333333
-0.3333…	-0.3333…	0.666667
-0.3333…	-0.3333…	0.666667

Classification Example (K=3)

Build a decision tree $h_3(\mathbf{x})$ of depth 1 from $\boldsymbol{X}$ to $-g_3$: [Regression Problem!]

$$h_3^{(1)}(\mathbf{x}) = \begin{cases} -0.333 & \text{if } x_4 \leq 1.70 \\ 0.667 & \text{if } x_4 > 1.70 \end{cases}$$

0	5.4	3.4	1.7	0.2
1	6.5	3	5.2	2
2	4.3	3	1.1	0.1
3	6.6	3	4.4	1.4
4	4.9	3.1	1.5	0.1
5	5	3.2	1.2	0.2
6	5.5	2.5	4	1.3
7	5.7	2.9	4.2	1.3
8	5.1	2.5	3	1.1
9	4.8	3.4	1.6	0.2
10	6.7	3.1	5.6	2.4
11	6.5	3	5.8	2.2

$h_3(\mathbf{x})$ →

-0.333333
0.666667
-0.333333
-0.333333
-0.333333
-0.333333
-0.333333
-0.333333
-0.333333
-0.333333
0.666667
0.666667

Classification Example (K=3)

Step 4: Updating $F := F + \rho h$ with $\rho = 1$ for simplicity

$$F_1^{(1)}(\mathbf{x}) := F_1^{(0)}(\mathbf{x}) + \rho h_1^{(1)}(\mathbf{x}) = h_1^{(1)}(\mathbf{x}) = \begin{cases} 0.667 & \text{if } x_4 \leq 0.65 \\ -0.333 & \text{if } x_4 > 0.65 \end{cases}$$

$$F_2^{(1)}(\mathbf{x}) := F_2^{(0)}(\mathbf{x}) + \rho h_2^{(1)}(\mathbf{x}) = h_2^{(1)}(\mathbf{x}) = \begin{cases} 0.667 & \text{if } x_2 \leq 2.95 \\ -0.222 & \text{if } x_2 > 2.95 \end{cases}$$

$$F_3^{(1)}(\mathbf{x}) := F_3^{(0)}(\mathbf{x}) + \rho h_3^{(1)}(\mathbf{x}) = h_3^{(1)}(\mathbf{x}) = \begin{cases} -0.333 & \text{if } x_4 \leq 1.70 \\ 0.667 & \text{if } x_4 > 1.70 \end{cases}$$

0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
5	0	0	0
6	0	0	0
7	0	0	0
8	0	0	0
9	0	0	0
10	0	0	0
11	0	0	0

+

0.666667	-0.222222	-0.333333
-0.333333	-0.222222	0.666667
0.666667	-0.222222	-0.333333
-0.333333	-0.222222	-0.333333
0.666667	-0.222222	-0.333333
0.666667	-0.222222	-0.333333
-0.333333	0.666667	-0.333333
-0.333333	0.666667	-0.333333
-0.333333	0.666667	-0.333333
0.666667	-0.222222	-0.333333
-0.333333	-0.222222	0.666667
-0.333333	-0.222222	0.666667

Classification Example (K=3)

Step 5: F to P again

$F_1^{(1)}$	$F_2^{(1)}$	$F_3^{(1)}$
0.666667	-0.222222	-0.333333
-0.333333	-0.222222	0.666667
0.666667	-0.222222	-0.333333
-0.333333	-0.222222	-0.333333
0.666667	-0.222222	-0.333333
0.666667	-0.222222	-0.333333
-0.333333	0.666667	-0.333333
-0.333333	0.666667	-0.333333
-0.333333	0.666667	-0.333333
0.666667	-0.222222	-0.333333
-0.333333	-0.222222	0.666667
-0.333333	-0.222222	0.666667

$$P_j(\mathbf{x}_n) = \frac{e^{F_j(\mathbf{x}_n)}}{e^{F_1(\mathbf{x}_n)} + e^{F_2(\mathbf{x}_n)} + e^{F_3(\mathbf{x}_n)}}$$

$P_1^{(1)}$	$P_2^{(1)}$	$P_3^{(1)}$
0.562116	0.231093	0.206791
0.206791	0.231093	0.562116
0.562116	0.231093	0.206791
0.320768	0.358464	0.320768
0.562116	0.231093	0.206791
0.562116	0.231093	0.206791
0.211942	0.576117	0.211942
0.211942	0.576117	0.211942
0.211942	0.576117	0.211942
0.562116	0.231093	0.206791
0.206791	0.231093	0.562116
0.206791	0.231093	0.562116

Classification Example (K=3)

Step 6: Negative Gradient again

			$P_1^{(1)}$	$P_2^{(1)}$	$P_3^{(1)}$	$-g_1$	$-g_2$	$-g_3$
1	0	0	0.562116	0.231093	0.206791	0.437884	-0.231093	-0.206791
0	0	1	0.206791	0.231093	0.562116	-0.206791	-0.231093	0.437884
1	0	0	0.562116	0.231093	0.206791	0.437884	-0.231093	-0.206791
0	1	0	0.320768	0.358464	0.320768	-0.320768	0.641536	-0.320768
1	0	0	0.562116	0.231093	0.206791	0.437884	-0.231093	-0.206791
1	0	0	0.562116	0.231093	0.206791	0.437884	-0.231093	-0.206791
0	1	0	0.211942	0.576117	0.211942	-0.211942	0.423883	-0.211942
0	1	0	0.211942	0.576117	0.211942	-0.211942	0.423883	-0.211942
0	1	0	0.211942	0.576117	0.211942	-0.211942	0.423883	-0.211942
1	0	0	0.562116	0.231093	0.206791	0.437884	-0.231093	-0.206791
0	0	1	0.206791	0.231093	0.562116	-0.206791	-0.231093	0.437884
0	0	1	0.206791	0.231093	0.562116	-0.206791	-0.231093	0.437884

Classification Example (K=3)

Step 7: Modeling Negative Gradients again, we have the second set of basic modeler $h_1(\mathbf{x}), h_2(\mathbf{x}), h_3(\mathbf{x})$: Decision trees of depth 1

$$h_1^{(2)}(\mathbf{x}) = \begin{cases} 0.437 & \text{if } x_4 \le 0.65 \\ -0.225 & \text{if } x_4 > 0.65 \end{cases}$$

$$h_2^{(2)}(\mathbf{x}) = \begin{cases} 0.278 & \text{if } x_2 \le 2.95 \\ -0.076 & \text{if } x_2 > 2.95 \end{cases}$$

$$h_3^{(2)}(\mathbf{x}) = \begin{cases} -0.155 & \text{if } x_4 \le 1.70 \\ 0.296 & \text{if } x_4 > 1.70 \end{cases}$$

$$F_1^{(1)}(\mathbf{x}) = \begin{cases} 0.667 & \text{if } x_4 \le 0.65 \\ -0.333 & \text{if } x_4 > 0.65 \end{cases}$$

Note: the splitting points for this set of basic models are the same as the first set of basic models.

Step 8: Final models (suppose we stop here)

$$F_1(\mathbf{x}) = F_1^{(2)}(\mathbf{x}) := F_1^{(1)}(\mathbf{x}) + h_1^{(2)}(\mathbf{x}) = \begin{cases} 1.104 & \text{if } x_4 \le 0.65 \\ -0.558 & \text{if } x_4 > 0.65 \end{cases}$$

$$F_2(\mathbf{x}) = F_2^{(2)}(\mathbf{x}) := F_2^{(1)}(\mathbf{x}) + h_2^{(2)}(\mathbf{x}) = \begin{cases} 0.945 & \text{if } x_2 \le 2.95 \\ -0.298 & \text{if } x_2 > 2.95 \end{cases}$$

$$F_3(\mathbf{x}) = F_3^{(2)}(\mathbf{x}) := F_3^{(1)}(\mathbf{x}) + h_3^{(2)}(\mathbf{x}) = \begin{cases} -0.488 & \text{if } x_4 \le 1.70 \\ 0.963 & \text{if } x_4 > 1.70 \end{cases}$$

Classification Example (K=3)

Step 9: Prediction on $\mathbf{x}^* = [4.7, 3.2, 1.3, 0.2]$

$F_1(\mathbf{x}^*) = 1.104;\ F_2(\mathbf{x}^*) = -0.298;\ F_3(\mathbf{x}^*) = -0.488$

Hence the probabilities are

$$P_1(\mathbf{x}^*) = \frac{e^{F_1(\mathbf{x}^*)}}{e^{F_1(\mathbf{x}^*)} + e^{F_2(\mathbf{x}^*)} + e^{F_3(\mathbf{x}^*)}} = \frac{e^{1.104}}{e^{1.104} + e^{-0.298} + e^{-0.488}} = 0.690$$

$$P_2(\mathbf{x}^*) = \frac{e^{F_2(\mathbf{x}^*)}}{e^{F_1(\mathbf{x}^*)} + e^{F_2(\mathbf{x}^*)} + e^{F_3(\mathbf{x}^*)}} = \frac{e^{-0.298}}{e^{1.104} + e^{-0.298} + e^{-0.488}} = 0.170$$

$$P_3(\mathbf{x}^*) = \frac{e^{F_3(\mathbf{x}^*)}}{e^{F_1(\mathbf{x}^*)} + e^{F_2(\mathbf{x}^*)} + e^{F_3(\mathbf{x}^*)}} = \frac{e^{-0.488}}{e^{1.104} + e^{-0.298} + e^{-0.488}} = 0.140$$

Now this sample can be classified as Class 1.

We can do more round. See `Lecture08_Example01.py`

Gradient Tree Boosting

Advanced Consideration

- In the basic GB algorithm, we leave $\rho = 1$. Can we change that? And How?
- We have not specified what the basic model for $h(\mathbf{x})$ is, and how to fit $h(\mathbf{x})$ to the negative gradient.
- Fitting $h(\mathbf{x})$ to the negative gradient turns out to be a regression problem no matter whether GB is solving regression or classification problems
- In most of GB algorithms, people are using decision trees for $h(\mathbf{x})$.
- In the basic GB algorithm, fitting $h(\mathbf{x})$ to the negative gradient is a separate task, however this can be incorporated to the overall model fitting
- We have not considered regularisation yet!

Theoretical Explanation: Basic Gradient Boosting

- Let $\hat{F}^{(m-1)}(\mathbf{x}_n)$ be the prediction of the n-th case at the $(m-1)$-th iteration, we need to add h_m to minimise the following objective

$$L^{(m)} = \sum_{n=1}^{N} L(t_n, \hat{F}^{(m-1)}(\mathbf{x}_n) + h_m(\mathbf{x}_n)) + \Omega(h_m)$$

$\hat{F}^{(m)}(\mathbf{x}_n)$

- Using Taylor approximation, one has

All constants

$$L^{(m)} \approx \sum_{n=1}^{N} \left[L(t_n, \hat{F}^{(m-1)}(\mathbf{x}_n)) + g_n h_m(\mathbf{x}_n) + \frac{1}{2} e_n h_m^2(\mathbf{x}_n) \right] + \Omega(h_m)$$

where

$$g_n = \frac{\partial L(t_n, \hat{F}^{(m-1)}(\mathbf{x}_n))}{\partial \hat{F}^{(m-1)}(\mathbf{x}_n)} \quad e_n = \frac{\partial^2 L(t_n, \hat{F}^{(m-1)}(\mathbf{x}_n))}{\partial \hat{F}^{(m-1)}(\mathbf{x}_n) \partial \hat{F}^{(m-1)}(\mathbf{x}_n)}$$

Note: When Loss is squared error, $e_n = 1$.

Theory Explanation: Basic Gradient Boosting

- In basic GB, we write the objective as

$$L^{(m)} \approx \sum_{n=1}^{N} \frac{1}{2} e_n \left[h_m(\mathbf{x}_n) - \left(-\frac{g_n}{e_n} \right) \right]^2 + \Omega(h_m) + consts$$

- Minimising $L^{(m)}$ is equivalent to fitting h_m to $\left(-\frac{g_n}{e_n} \right)$

- For example, for the regression with the squared error loss function, we have $e_n = 1$. That is why we fit $h_m(\mathbf{x})$ to the negative gradient $-g(\mathbf{x}_n)$.

- For the cross entropy loss function, we no longer have $e_n = 1$, but we still fit $h_m(\mathbf{x})$ to the negative gradient $-g(\mathbf{x}_n)$.

Gradient Tree Boosting

- Using a tree model h_m in m-th iteration, Gradient Tree Boosting takes two steps.
  - Step 1: Fitting the tree h_m to $\left(-\frac{g_n}{e_n}\right)$. Keep the tree structure, i.e., the partition of the input space as J regions $R_1, R_2, \ldots, R_J$
  - Step 2: We express the tree as

$$h_m(x) = \sum_{j=1}^{J} \beta_j 1(\mathbf{x}; R_j) \quad \textbf{where } 1(\mathbf{x}; R_j) = \begin{cases} 1 & \mathbf{x} \in R_j \\ 0 & \text{otherwise} \end{cases}$$

Optimize the following, with respect to all β_j,

$$L^{(m)} \approx \sum_{n=1}^{N} \left[g_n h_m(\mathbf{x}_n) + \frac{1}{2} e_n h_m^2(\mathbf{x}_n) \right] + \frac{1}{2}\lambda \sum_{j=1}^{J} \beta_j^2 + \gamma J$$

Gradient Tree Boosting

- Consider the model we are working on

$$h_m(\mathbf{x}) = \sum_{j=1}^{J} \beta_j 1(\mathbf{x}; R_j) \quad \text{where } 1(\mathbf{x}; R_j) = \begin{cases} 1 & \mathbf{x} \in R_j \\ 0 & \text{otherwise} \end{cases}$$

- Can I say, if the sample $\mathbf{x}_n$ falls in the region R_1, then

$$h_m(\mathbf{x}_n) = \beta_1 \text{ and } h_m^2(\mathbf{x}_n) = \beta_1^2?$$

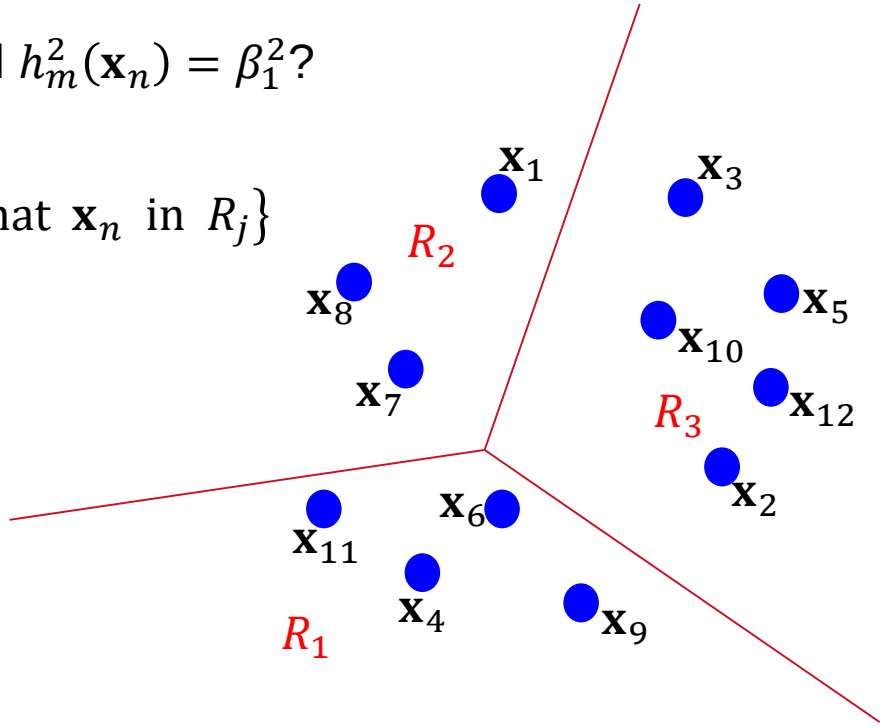
- Let us denote

$$I_j = \{n: \text{ such that } \mathbf{x}_n \text{ in } R_j\}$$

- What is the meaning of I_j?

For the example on the right, write out

I_1, I_2, I_3



Gradient Tree Boosting

- So we can re-write

$$L^{(m)} \approx \sum_{n=1}^{N} \left[g_n h_m(\mathbf{x}_n) + \frac{1}{2} e_n h_m^2(\mathbf{x}_n) \right] + \frac{1}{2}\lambda \sum_{j=1}^{J} \beta_j^2 + \gamma J$$

As

$$L^{(m)} \approx \sum_{j=1}^{J} \left[\left(\sum_{n \in I_j} g_n \right) \beta_j + \frac{1}{2} \left(\sum_{n \in I_j} e_n + \lambda \right) \beta_j^2 \right] + \gamma J$$

- This can be minimised with respect to each β_j, i.e.,

$$\min_{\beta_j} \left(\sum_{n \in I_j} g_n \right) \beta_j + \frac{1}{2} \left(\sum_{n \in I_j} e_n + \lambda \right) \beta_j^2$$

- The best β_j^* is given by [exercise!]

$$\beta_j^* = -\frac{\sum_{n \in I_j} g_n}{\sum_{n \in I_j} e_n + \lambda}$$

THE UNIVERSITY OF SYDNEY

XGBoost

XGBoost

- **XGBoost** stands for e**X**treme **G**radient **B**oosting which is an implementation aiming to achieve an efficient and scalable learning system
- XGBoost is proposed by Tianqi Chen for large-scale machine learning in 2014
- XGBoost has recently been dominating applied machine learning and It has been widely used in Kaggle competitions for structured or tabular data.
- It can be used to large-scale machine learning problems with great efficiency and accuracy

XGBoost vs Gradient Boosting

- Gradient Boosting was however plagued by a lot of ad-hoc parameters to control the growth of the decision trees in the algorithm

- In XGBoost the size of the tree and the magnitude of the weights are controlled by standard regularization parameters.

- This leads to a ‘mostly’ parameter-free optimization routine.

https://www.quora.com/What-is-the-difference-between-eXtreme-Gradient-Boosting-XGBoost-AdaBoost-and-Gradient-Boosting

pip3 install xgboost

Using XGBoost

- First you shall install xgboost in your python. From command terminal of your anaconda, issue the following command to install it

```
pip3 install xgboost
```

- `Lecture08_Example02.py` shows you all the steps you need to use XGBoost
- XGBoost can do both regression and classification. You can control the depth of a basic tree model etc. Also you can control the maximum number of leaves to be added
- Unfortunately XGBoost manages only numeric data, so you must convert your categorical or ordinal data to numeric format