Introduction to Machine Learning

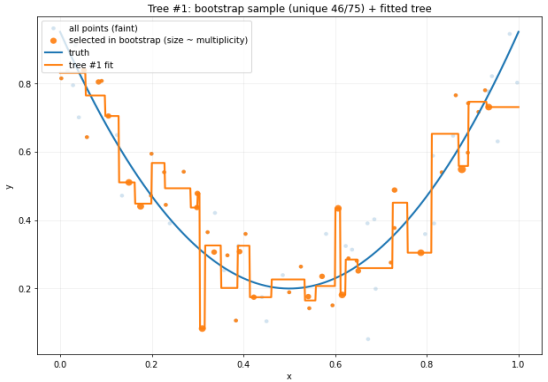
Lecture 16:

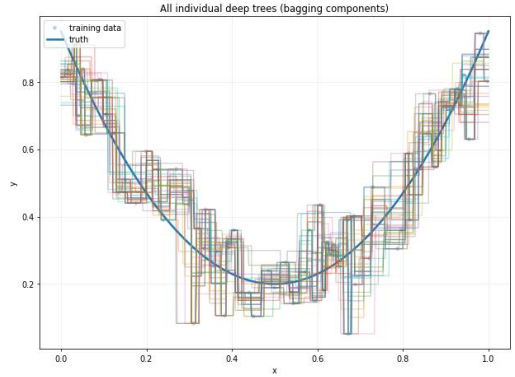
Boosting: AdaBoost, Gradient Boosting

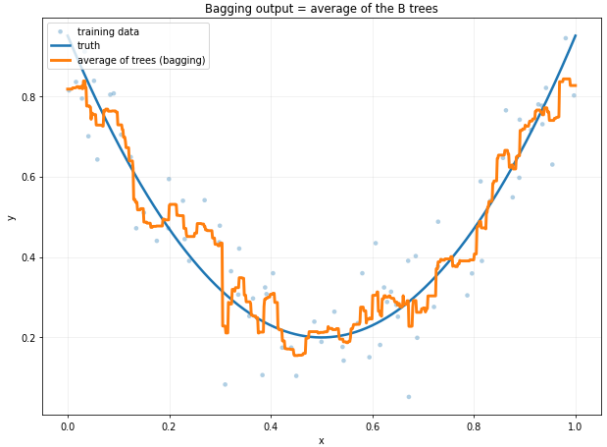
Prof. Subrahmanya Swamy Peruru
Department of Electrical Engineering
IIT Kanpur

Bagging reduces variance by averaging

A single deep tree

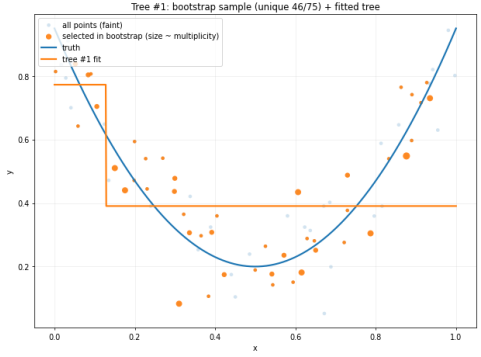


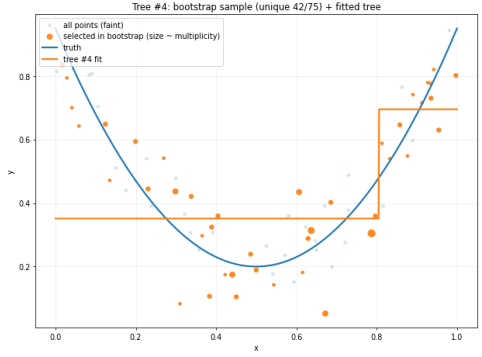


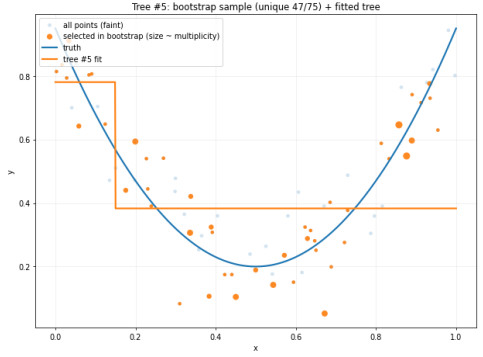


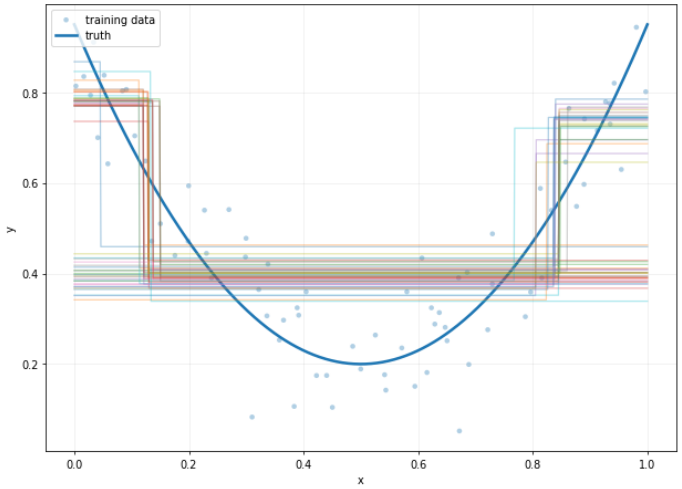
- Each tree uses different subsets of data through bootstrapping (sampling with replacement)
- Learns multiple strong (**deep**) noisy trees
- All trees are learnt in **parallel**

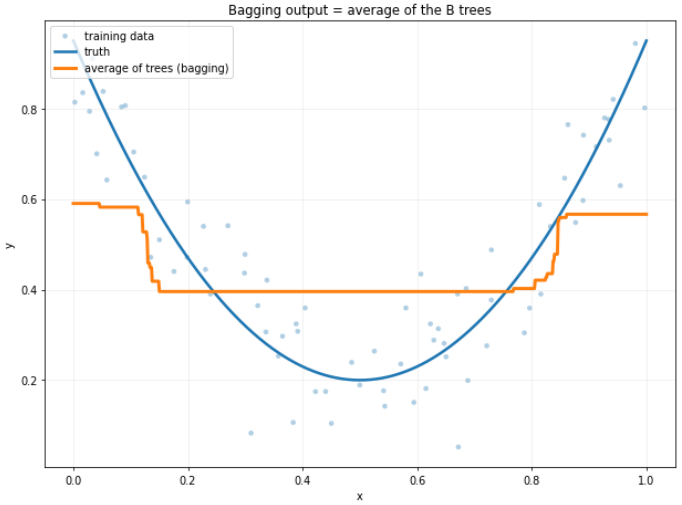
Can I use shallow trees instead of deep trees?









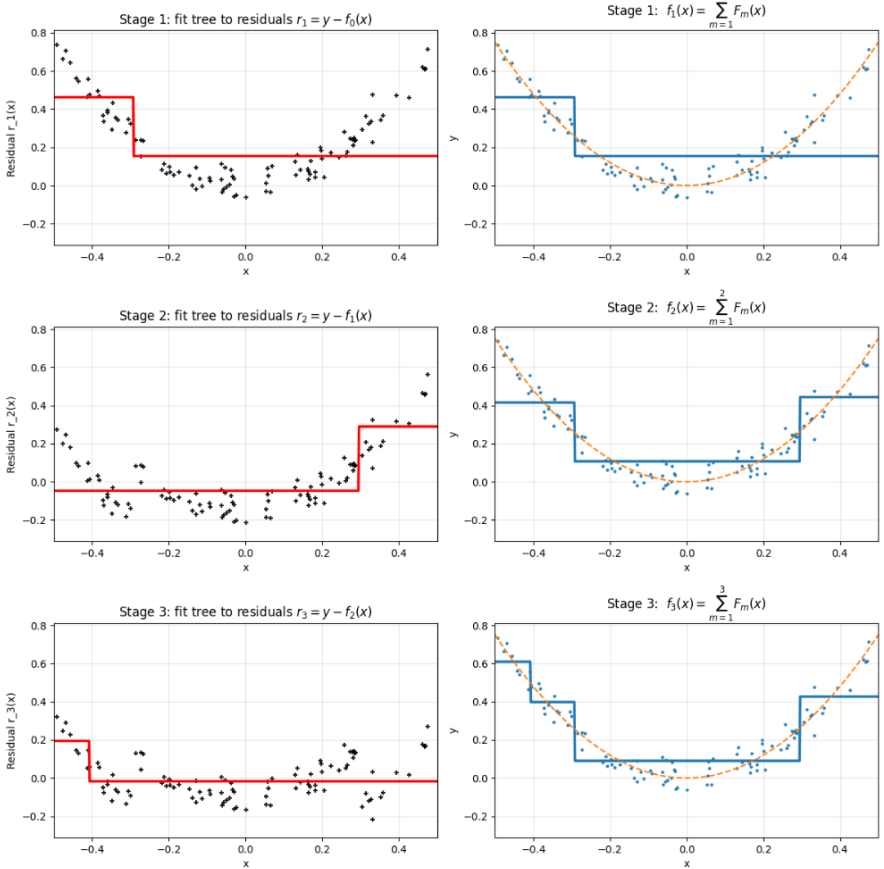


Shallow trees + **Bagging**
→ **Still underfitting**

Solution:
Shallow trees + **Boosting**

Learns sequentially by correcting previous errors

Boosting (Regression)



$F_i(\mathbf{x})$: Tree learnt at i^{th} stage

$f_i(\mathbf{x})$: Aggregate model at i^{th} stage

Least Squares Boosting

Learn weak trees that sequentially boost the previous stage's performance

A weak (shallow) tree (e.g. tree of depth 1) is learnt in each stage

Recall that depth 1 tree just

- partitions the input space into just two regions and
- predicts a constant value in each regions

$F_i(\mathbf{x})$: Tree learnt in the i^{th} stage

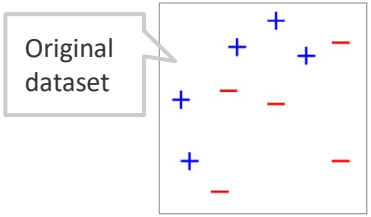
$f_i(\mathbf{x}) = \sum_{m=1}^{i} F_m(\mathbf{x})$: Aggregate model at i^{th} stage

The i^{th} stage tree $F_i(\mathbf{x})$:

- is fitted to the residual error so far: $y - f_{i-1}(\mathbf{x})$

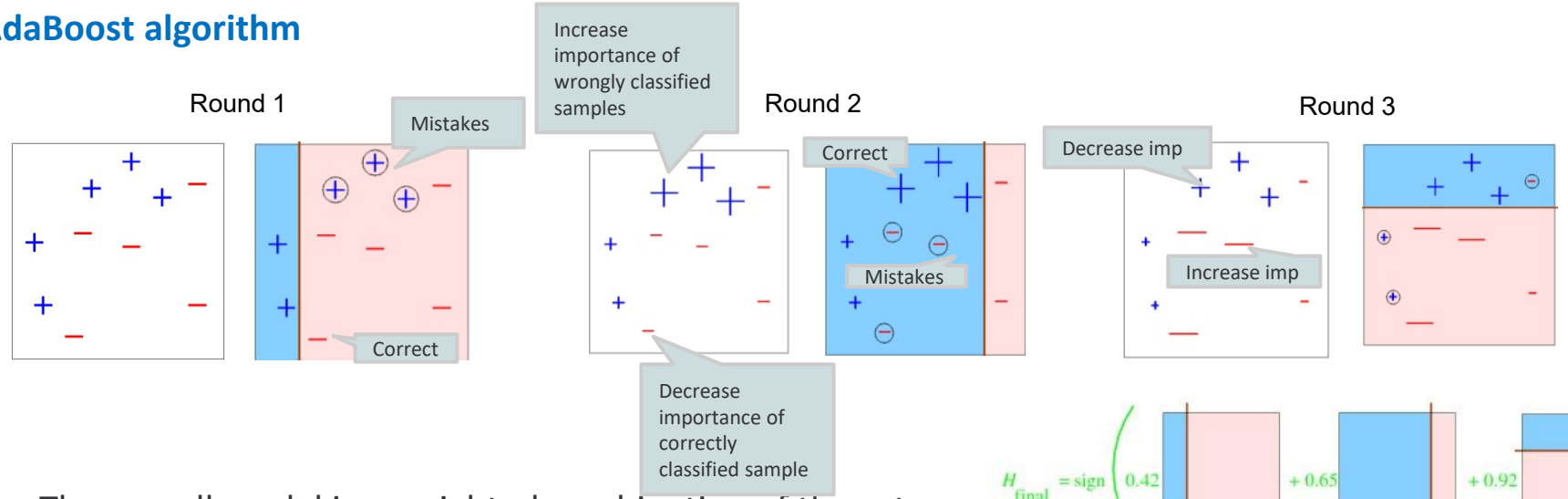
Hence, boosts the aggregate performance by adding an estimate of the residual $f_i(\mathbf{x}) = f_{i-1}(\mathbf{x}) + F_i(\mathbf{x})$

Boosting (Classification)



- Consider a Binary classification problem with each input having 2 features.
- A weak model like a simple decision stump (tree of depth 1)

AdaBoost algorithm



- The overall model is a weighted combination of these trees
- Two questions to understand:
  - How to increase/decrease the importance of a sample?
  - How to decide the weight of each tree in the overall model?

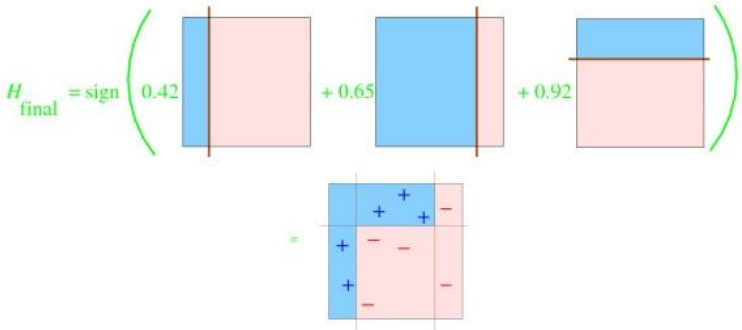


Fig credits: Prof. Piyush Rai

AdaBoost (Adaptive Boosting)

How to increase/decrease importance to a sample?

- Use a weighed loss function $\sum_{i=1}^{N} w_i \, \mathbf{1}(y_i \neq F(\mathbf{x}_i))$
- Exponentially increase weights of wrongly classified samples

How to decide the weight of a tree in the overall model $\sum_{m=1}^{M} \alpha_m \, F_m(\mathbf{x})$?

- Based on classification error of that tree (smaller the error, larger the weight)

Initialize to equal sample weights: $w_i = \frac{1}{N}, \quad i = 1, \ldots, N$

For $m = 1, \ldots, M$:

- Train a weak learner (e.g., a decision stump) that minimizes the weighted misclassification error

$$F_m = \text{argmin}_F \sum_{i=1}^{N} w_i \, \mathbf{1}(y_i \neq F(\mathbf{x}_i))$$

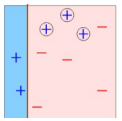
- Compute weighted classification error of the trained weak learner: $\varepsilon_m = \sum_{i=1}^{N} w_i \, \mathbf{1}(y_i \neq F_m(\mathbf{x}_i))$
- Compute tree weight (smaller the error, larger the tree weight):

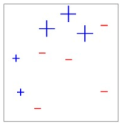
$$\alpha_m = \frac{1}{2} \log \frac{1 - \varepsilon_m}{\varepsilon_m}$$

- Update sample weights

If $y_i = F_m(\mathbf{x}_i)$ (correctly classified): $w_i \leftarrow w_i e^{-\alpha_m}$

If $y_i \neq F_m(\mathbf{x}_i)$ (misclassified): $w_i \leftarrow w_i e^{+\alpha_m}$





- Normalize weights so that $\sum_i w_i = 1$.

Final classifier after all the stages: $f(\mathbf{x}) = \text{sgn}(\sum_{m=1}^{M} \alpha_m \, F_m(\mathbf{x}))$

Weak Learning Requirement:

Each weak learner must perform better than random guessing (i.e., achieve weighted error strictly less than 50%)

$$\varepsilon_m < \frac{1}{2}$$

AdaBoost Derivation

AdaBoost as Forward Stage-wise Additive Modeling

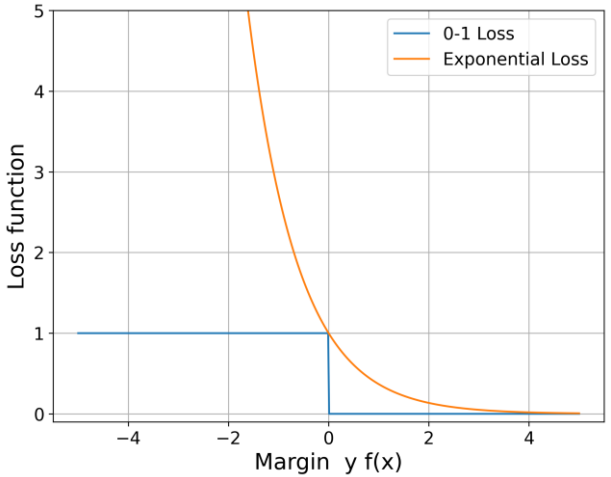
Additive model: An ensemble as a weighted sum of weak learners:

$$f_M(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m F_m(\mathbf{x}), \qquad F_m(\mathbf{x}) \in \{-1, +1\}$$

Prediction based on ensemble score: $\text{sgn}\big(f_M(\mathbf{x})\big)$

Use exponential loss:

$$L(f) = \sum_{i=1}^{n} \exp\left(-y_i f(\mathbf{x}_i)\right)$$



Forward stage-wise update: $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \alpha_m F_m(\mathbf{x})$

At stage m: Learn the best m^{th} stage learner keeping the previous stage $f_{m-1}(\mathbf{x})$ fixed

$$(\alpha_m, F_m) = \arg\min_{\alpha, F} \sum_{i=1}^{n} \exp\left(-y_i (f_{m-1}(\mathbf{x}_i) + \alpha F(\mathbf{x}_i))\right)$$

Factoring previous ensemble in the objective:

$$\sum_{i=1}^{n} \underbrace{\exp(-y_i f_{m-1}(\mathbf{x}_i))}_{w_i^{(m)} = \exp(-y_i f_{m-1}(\mathbf{x}_i))} \exp(-\alpha y_i F(\mathbf{x}_i))$$

Weights emerge naturally from exponential loss.

Equivalence to minimizing Weighted error

Since $F(\mathbf{x}_i) \in \{-1, +1\}$, the objective can be written as:

$$\sum_i w_i^{(m)} e^{-\alpha y_i F(\mathbf{x}_i)} = e^{-\alpha} \sum_{y_i = F(\mathbf{x}_i)} w_i^{(m)} + e^{\alpha} \sum_{y_i \neq F(\mathbf{x}_i)} w_i^{(m)}$$

Normalizing the weights by dividing with $\sum_i w_i^{(m)}$ gives:

$$= e^{-\alpha} \sum_{y_i = F(\mathbf{x}_i)} \frac{w_i^{(m)}}{\sum_i w_i^{(m)}} + e^{\alpha} \sum_{y_i \neq F(\mathbf{x}_i)} \frac{w_i^{(m)}}{\sum_i w_i^{(m)}}$$

ϵ_m weighted misclassification error

$$= e^{-\alpha}(1 - \epsilon_m) + e^{\alpha}\, \epsilon_m$$

$$= e^{-\alpha} + \epsilon_m (e^{\alpha} - e^{-\alpha})$$

For fixed $\alpha > 0$, minimizing objective over F is equivalent to

$$F_m = \arg\min_F \epsilon_m$$

Optimizing the weight α_m

Now minimize the objective over α:

$$e^{-\alpha}(1-\epsilon_m)+e^{\alpha}\epsilon_m$$

Set derivative to zero:

$$(1-\epsilon_m)\,(-e^{-\alpha})+\epsilon_m\ e^{\alpha}=0$$

$$\epsilon_m e^{\alpha}=(1-\epsilon_m)e^{-\alpha}$$

$$e^{2\alpha}=\frac{1-\epsilon_m}{\epsilon_m}$$

$$\boxed{\alpha_m=\frac{1}{2}\log\frac{1-\epsilon_m}{\epsilon_m}}$$

So:

$$\text{smaller } \epsilon_m \ \Rightarrow\ \text{larger } \alpha_m$$

More accurate weak learners get larger influence in the ensemble.

Gradient Boosting

Forward Stage-wise Additive Modeling

Additive model:

$$f_M(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m F_m(\mathbf{x})$$

Stage-wise construction:

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \alpha_m F_m(\mathbf{x})$$

Optimization at stage m:

$$(\alpha_m, F_m) = \arg\min_{\alpha, F} \sum_{i=1}^{n} L\left(y_i, (f_{m-1}(\mathbf{x}_i) + \alpha F(\mathbf{x}_i))\right)$$

AdaBoost:
For exponential loss $L(y, f(\mathbf{x})) = e^{-yf(\mathbf{x})}$ and $F(\mathbf{x}) \in \{-1, +1\}$, this yields AdaBoost in closed form.

For a general loss function , exact stage-wise optimization is typically intractable.

Solution: Gradient Boosting
Approximate the stage-wise minimization via gradient descent in function space.

Gradient in Function Space

At the training points, the function f is represented as:

$$\mathbf{f} = (f(\mathbf{x}_1), \dots, f(\mathbf{x}_N)) \in \mathbb{R}^N.$$

With empirical risk:

$$\mathcal{L}(\mathbf{f}) = \sum_{i=1}^{N} L\left(y_i, f(\mathbf{x}_i)\right).$$

Gradient:

This allows us to compute gradients of the empirical risk with respect to the vector of function values:

$$\nabla \mathcal{L}(\mathbf{f}) = \left(\frac{\partial L(y_1, f(\mathbf{x}_1))}{\partial f(\mathbf{x}_1)}, \dots, \frac{\partial L(y_N, f(\mathbf{x}_N))}{\partial f(\mathbf{x}_N)} \right).$$

Weak learner F_m is designed to approximate the negative gradient vector.

Gradient Boosting Principle

Compute the gradient at $f = f_{m-1}$:

$$g_{im} = \left.\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right|_{f=f_{m-1}}$$

Fit weak learner to negative gradients (pseudo residuals) $r_{im} = -g_{im}$:

$$F_m = \arg\min_F \sum_{i=1}^{N} (r_{im} - F(\mathbf{x}_i))^2$$

Update:

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \alpha_m F_m(\mathbf{x})$$

(Optionally choose α_m by line search.)

Repeat the procedure for M stages.

Least Squares Boosting (Regression)

Squared loss:

$$L(y_i, f(\mathbf{x}_i)) = \frac{1}{2}(y_i - f(\mathbf{x}_i))^2$$

Gradient:

$$\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)} = -(y_i - f(\mathbf{x}_i))$$

Pseudo-residual:

$$r_{im} = y_i - f_{m-1}(\mathbf{x}_i)$$

Algorithm:

- Compute residuals:

$$r_{im} = y_i - f_{m-1}(\mathbf{x}_i)$$

- Fit weak regressor F_m to the residual r_{im}
- Update:

$$f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + F_m(\mathbf{x})$$

Conclusion: Residual fitting = gradient boosting with squared loss.

Unified View of Boosting

Gradient Boosting Template

1. Compute pseudo-residuals:
2. $g_{im} = -\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}$
3. Fit weak learner F_m to g_{im}
4. Update:
5. $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \alpha_m F_m(\mathbf{x})$

Loss	Pseudo-residual	Algorithm
$(y-f)^2$	$y-f$	Least Squares Boosting
$\exp(-yf)$	ye^{-yf}	AdaBoost

Takeaway: Different loss ⇒ Different boosting algorithm.

Exercise: Derive gradient boosting updates for absolute loss and logistic (log-loss).

Modern Gradient Boosting Systems

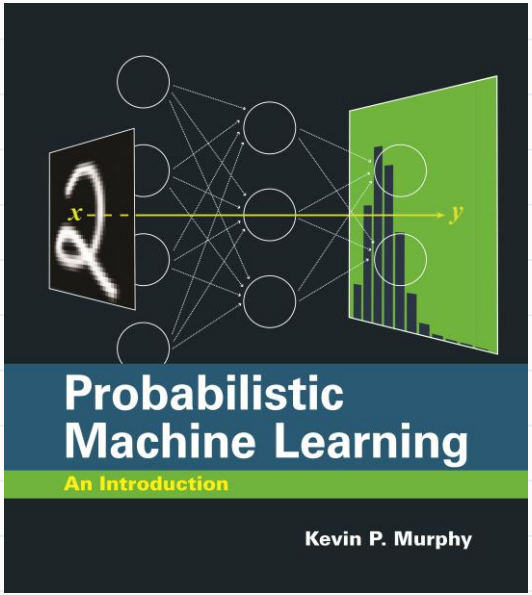
XGBoost – Regularized, second-order (Newton) tree boosting

LightGBM – Fast, leaf-wise tree boosting for large-scale data

Reference

Chapter 18.5

of “Probabilistic Machine Learning: An Introduction” book by Kevin P. Murphy



Thank You