Predictive Analytics:
Ensemble of Gradient-Boosted Decision Trees

Dr. Josif Grabocka

ISMLL, University of Hildesheim

Business Analytics

Predictive Analytics - Example

- For N existing bank customers and $M = 23$ features, i.e. given $X \in \mathbb{R}^{N \times 23}$ and ground truth $Y \in \{0, 1\}^N$

$Y_{:}$	Default credit card payment (Yes = 1, No = 0)
$X_{:,1}$	Amount of the given credit (NT dollar)
$X_{:,2}$	Gender (1 = male; 2 = female).
$X_{:,3}$	Education (1=graduate; 2=univ.; 3 = high school; 4 = others).
$X_{:,4}$	Marital status (1 = married; 2 = single; 3 = others).
$X_{:,5}$	Age (year)
$X_{:,6} - X_{:,11}$	Past Delays (-1=duly, . . . , 9=delay of nine months)
$X_{:,12} - X_{:,17}$	Amount of bill statements
$X_{:,18} - X_{:,23}$	Amount of previous payments

Table 1: Yeh, I. C., & Lien, C. H. (2009).

- Goal: Estimate the default of a new $(N + 1)$-th customer, i.e. given $X_{N+1,:} \in \mathbb{R}^{23}$, estimate $Y_{N+1} =?$

Problem Definition

- *Data dimensions*: N instances having M features
- *Features*: $x \in \mathbb{R}^{N \times M}$ and *Target*: $y \in \mathbb{R}^N$
- A *prediction model*: having parameters $\theta \in \mathbb{R}^K$ is $f : \mathbb{R}^M \times \mathbb{R}^K \to \mathbb{R}$

$$\hat{y}_n := f(x_n, \theta)$$

- *Loss function*: $\mathcal{L}(y_n, \hat{y}_n) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$
- *Regularization*: $\Omega(\theta) : \mathbb{R}^K \to \mathbb{R}$
- *Objective function*:

$$\underset{\theta}{\operatorname{argmin}} \sum_{n=1}^{N} \mathcal{L}(y_n, \hat{y}_n) + \Omega(\theta)$$

Prediction Models and Loss Functions

- **Prediction model**:
  - Linear model, i.e. $\hat{y}_n = \sum_{m=1}^{M} \theta_m X_{n,m}$
  - Non-linear models, e.g.: Neural Networks, Kernel-space representation, Decision Trees
- **Loss Function**:
  - Regression (target is real-values $y_n \in \mathbb{R}$), e.g. least-squares:

$$\mathcal{L}(y_n, \hat{y}_n) := (y_n - \hat{y}_n)^2$$

  - Binary Classification $y_n \in \{0, 1\}$, e.g. logistic loss:

$$\mathcal{L}(y_n, \hat{y}_n) := -y_n \log(\hat{y}_n) - (1 - y_n) \log(1 - \hat{y}_n)$$

Multi-class loss - Softmax

- Re-express targets $y_n \in \{1, \ldots, C\}$ as one-vs-all, i.e.

$$y_{n,c} := \begin{cases} 1 & y_n = C \\ 0 & y_n \neq C \end{cases}$$

- Learn model parameters per class $\theta \in \mathbb{R}^{C \times K}$
- Estimations expressed as probabilities among classes

$$\hat{y}_{n,c} = \frac{e^{f(x_n,\theta_c)}}{\sum_{q=1}^{C} e^{f(x_n,\theta_q)}}$$

- Logloss:

$$\mathcal{L}(y_{n,:}, \hat{y}_{n,:}) := -\sum_{c=1}^{C} y_{n,c} \log(\hat{y}_{n,c})$$

Classification and Regression Trees (CART)

A prediction model $\hat{y}_n := f(x_n, \theta)$ can be also a tree:

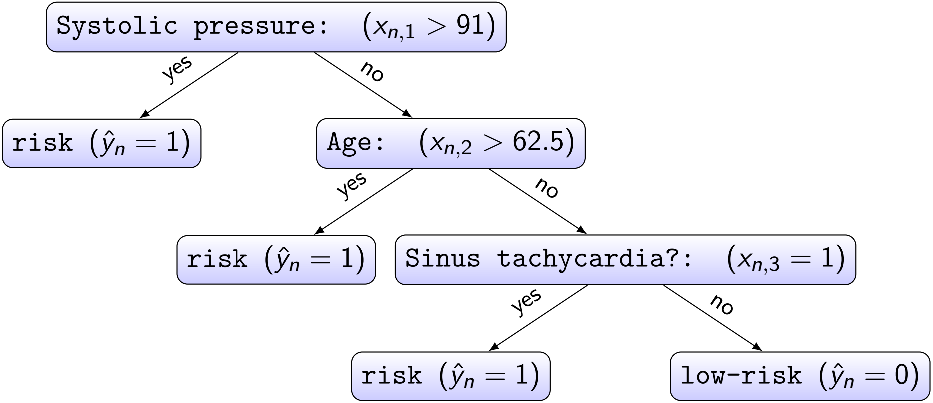


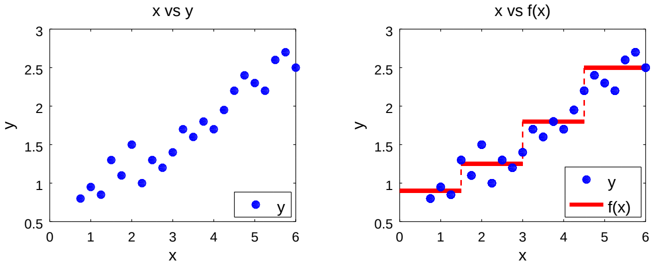
Figure 1: San Diego Medical Center

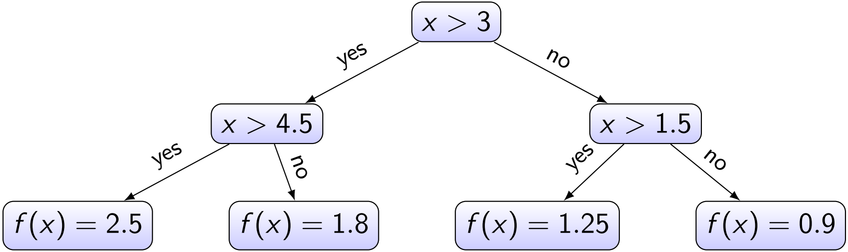
Prediction Model of a Decision Tree

- A tree having T leaves outputs the weights $w \in \mathbb{R}^T$.
- Let $q : \mathbb{R}^M \rightarrow \{1, \ldots, T\}$ denote the leaf index $q(x_n)$ where instance x_n belongs to, then
- The prediction model of a tree is:

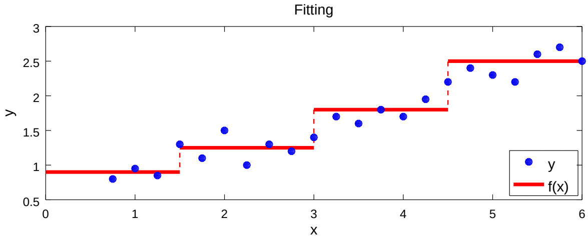
$$f(x_n) = w_{q(x_n)}$$

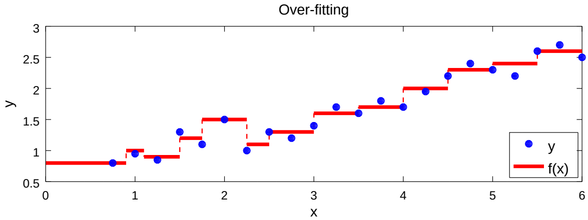
Decision Tree as a Step-wise Function





Tree Over-fitting





Tree over-fits if too many steps (nodes) and high jumps (large leaf weights)

Tree Regularization

- *Note:* Too many steps ≈ Too many leaves (T)
- *Note:* Too large step jumps ≈ Too large leaves' output values (w)
- Penalize the number of leaves and leaves' weights, e.g.:

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

Boosting

- Can weak learners (single trees) be combined to create more expressive models?
  - Jean de La Fontaine: "All power is weak unless united" (1668)

- Unite single trees into an ensemble of k trees
- The estimation is aggregated over the individual trees' predictions:

$$\begin{aligned}\hat{y}_n^{(1)} &:= f^{(1)}(x_n), \;\; \hat{y}_n^{(2)} := \hat{y}_n^{(1)} + f^{(2)}(x_n), \;\ldots \\ \hat{y}_n^{(k)} &:= \hat{y}_n^{(k-1)} + f^{(k)}(x_n) = \sum_{l=1}^{k} f^{(l)}(x_n)\end{aligned}$$

Boosted Ensemble Loss

- Add one tree at a time to the ensemble (greedy strategy)
- The loss created as a result of adding the contribution of the k-th tree is:

$$\underset{f^{(k)}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} \mathcal{L}^{(k)}(Y, \hat{y}_n^{(k-1)} + f^{(k)}(x_n)) \right] + \Omega(f^{(k)})$$

$$:= \underset{f^{(k)}}{\operatorname{argmin}} \left[\sum_{n=1}^{N} \mathcal{L}_n^{(k)} \right] + \Omega(f^{(k)})$$

- How to find the optimal k-th tree $f^{(k)}$?

Tailor Approximation

Remember Tailor Expansion (2nd degree):

$$F(x+\Delta x) \approx F(x) + \frac{dF(x)}{dx}\Delta x + \frac{1}{2}\frac{d^2F(x)}{dx^2}\Delta x^2$$

In our case $F := \mathcal{L}^{(k)}$ and $\Delta x = f^{(k)}$

$$\mathcal{L}_n^{(k)} \approx \mathcal{L}_n^{(k-1)} + \frac{\partial \mathcal{L}_n^{(k)}}{\partial \hat{y}_n^{(k-1)}} f^{(k)}(x_n) + \frac{1}{2}\frac{\partial^2 \mathcal{L}_n^{(k)}}{\partial \left(\hat{y}_n^{(k-1)}\right)^2}\left(f^{(k)}(x_n)\right)^2$$

$$\mathcal{L}_n^{(k)} \approx \mathcal{L}_n^{(k-1)} + G_n f^{(k)}(x_n) + \frac{1}{2} H_n \left(f^{(k)}(x_n)\right)^2$$

$$\text{where } G_n := \frac{\partial \mathcal{L}_n^{(k)}}{\partial \hat{y}_n^{(k-1)}}, \quad H_n := \frac{\partial^2 \mathcal{L}_n^{(k)}}{\partial \left(\hat{y}_n^{(k-1)}\right)^2}$$

Rewrite Objective

Since $\mathcal{L}_n^{(k-1)}$ is constant w.r.t. $f^{(k)}$, then rewrite objective as:

$$\underset{f^{(k)}}{\operatorname{argmin}} \sum_{n=1}^{N} \left[G_n f^{(k)}(x_n) + H_n \left(f^{(k)}(x_n) \right)^2 \right] + \Omega(f^{(k)})$$

with regularization:

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

Rewrite objective in terms of leaves

- Remember $f^{(k)}(x) := w_{q(x)}$ (previous slide).
- Let indices of all instances belonging into the j-th leaf be $I_j := \{n \mid q(x_n) = j\}$.

Then, the objective in terms of leaves' weights is:

$$\underset{w_1,\ldots,w_T}{\operatorname{argmin}} \sum_{n=1}^{N} \left[G_n w_{q(x_n)} + \frac{1}{2} H_n w^2_{q(x_n)} \right] + \gamma T + \frac{\lambda}{2} \sum_{j=1}^{T} w_j^2$$

$$\underset{w_1,\ldots,w_T}{\operatorname{argmin}} \sum_{j=1}^{T} \left[\left(\sum_{n \in I_j} G_n \right) w_j + \frac{1}{2} \left(\lambda + \sum_{n \in I_j} H_n \right) w_j^2 \right] + \gamma T$$

Optimal Tree Leaves

- Given the objective:

$$\underset{w_1,\ldots,w_T}{\operatorname{argmin}} \quad \sum_{j=1}^{T} \left[\left(\sum_{n \in I_j} G_n \right) w_j + \frac{1}{2} \left(\lambda + \sum_{n \in I_j} H_n \right) w_j^2 \right] + \gamma T$$

- Knowing that:

$$\frac{-A}{B} = \underset{x}{\operatorname{argmin}} \, Ax + \frac{1}{2} B x^2$$

- The optimal leaf weights w are:

$$w_j = -\frac{\sum\limits_{n \in I_j} G_n}{\lambda + \sum\limits_{n \in I_j} H_n}, \quad j = 1, \ldots, T$$

Ultimate Objective Function

- Given the objective:

$$\underset{w_1,\ldots,w_T}{\operatorname{argmin}} \quad \sum_{j=1}^{T} \left[\left(\sum_{n \in I_j} G_n \right) w_j + \frac{1}{2} \left(\lambda + \sum_{n \in I_j} H_n \right) w_j^2 \right] + \gamma T$$

- Knowing that:

$$\frac{-A^2}{2B} = \min_x Ax + \frac{1}{2} B x^2$$

- The final objective function is:

$$\mathcal{O}(G, H) := -\frac{1}{2} \sum_{j=1}^{T} \frac{\left(\sum\limits_{n \in I_j} G_n \right)^2}{\left(\lambda + \sum\limits_{n \in I_j} H_n \right)} + \gamma T$$

How to grow trees?

- The objective per leaf j is:

$$\mathcal{O}_j := -\frac{1}{2} \frac{\left(\sum_{n \in I_j} G_n \right)^2}{\left(\lambda + \sum_{n \in I_j} H_n \right)} + \gamma$$

- When splitting leaf j after a decision split we yield two sub-leaves $j^{(\text{Left})}$ and $j^{(\text{Right})}$
- The gain in minimizing the global objective after splitting leaf j:

$$\text{Gain}_j := \mathcal{O}_j - \left(\mathcal{O}_{j^{(\text{Left})}} + \mathcal{O}_{j^{(\text{Right})}} \right)$$

Gain of splitting a leaf

- Given: $\mathcal{O}_j := -\frac{1}{2}\frac{\left(\sum_{n \in I_j} G_n\right)^2}{\left(\lambda + \sum_{n \in I_j} H_n\right)} + \gamma, \quad \text{Gain}_j := \mathcal{O}_j - \left(\mathcal{O}_{j^{(\text{Left})}} + \mathcal{O}_{j^{(\text{Right})}}\right)$
- Derive:

$$\text{Gain}_j := \frac{1}{2}\left[\underbrace{\frac{\left(\sum_{n \in I_j^{(\text{Left})}} G_n\right)^2}{\left(\lambda + \sum_{n \in I_j^{(\text{Left})}} H_n\right)}}_{\text{Objective of left child}} + \underbrace{\frac{\left(\sum_{n \in I_j^{(\text{Right})}} G_n\right)^2}{\left(\lambda + \sum_{n \in I_j^{(\text{Right})}} H_n\right)}}_{\text{Objective of right child}} - \underbrace{\frac{\left(\sum_{n \in I_j} G_n\right)^2}{\left(\lambda + \sum_{n \in I_j} H_n\right)}}_{\text{Objective of parent}}\right] - \underbrace{\gamma}_{\text{Regularize additional leaf}}$$

Split rule search

- For each node, exhaustively visit all splitting rules:
  - For each feature $m = 1, \ldots, M$ of the data $X \in \mathbb{R}^{N \times M}$
    - Sort the instances $n = 1, \ldots, N$ of the m-th feature $x_{:,m} \in \mathbb{N}$
    - Denote the unique sort values $\mathcal{V}_m \in \mathbb{R}^{N'}$, where $N' \leq N$
    - Generate all split rules:

$$\left[x_{:,m}; \frac{\mathcal{V}_{m,n'} + \mathcal{V}_{m,n'+1}}{2}\right], \text{ for } n' = 1, \ldots, N' - 1$$

- Select the split rule that maximizes the gain

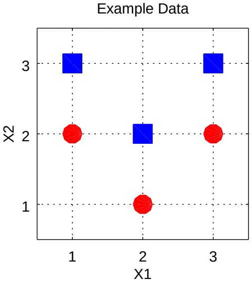
$$\underset{\substack{\left[x_{:,m}; \frac{\mathcal{V}_{m,n'} + \mathcal{V}_{m,n'+1}}{2}\right] \\ \forall m \in \{1,\ldots,M\} \\ \forall n' \in \{1,\ldots,|\mathcal{V}_{m,:}|-1\}}}{\text{argmin}} \quad \mathcal{O}_j - \left(\mathcal{O}_{j^{(\text{Left})}} + \mathcal{O}_{j^{(\text{Right})}}\right)$$

where

$$I_j^{(\text{Left})} = \left\{ n \mid x_{n,m} < \frac{\mathcal{V}_{m,n'} + \mathcal{V}_{m,n'+1}}{2} \right\}$$

$$I_j^{(\text{Right})} = \left\{ n \mid x_{n,m} > \frac{\mathcal{V}_{m,n'} + \mathcal{V}_{m,n'+1}}{2} \right\}$$

Exercise



n	x_1	x_2	y
1	1	2	0
2	2	1	0
3	3	2	0
4	1	3	1
5	2	2	1
6	3	3	1

- Learn an ensemble of 2 trees to estimate:
  - Limit maximum depth of trees to two.
  - Use logistic loss
  - Set $\gamma = 1$, $\lambda = 1$.

Exercise - Step 1: Gradients and Hessians

- Before building each tree compute the gradients and Hessians:

$$
\begin{aligned}
\mathcal{L}_n &= -y_n \log(\sigma(\hat{y}_n)) - (1 - y_n) \log(1 - \sigma(\hat{y}_n)) \\
G_n &= \frac{\partial \mathcal{L}_n}{\partial \hat{y}_n} = \sigma(\hat{y}_n) - y_n \\
H_n &= \frac{\partial^2 \mathcal{L}_n}{\partial (\hat{y}_n)^2} = \frac{\partial G_n}{\partial \hat{y}_n} = \sigma(\hat{y}_n)(1 - \sigma(\hat{y}_n))
\end{aligned}
$$

- Remember the prediction model of a boosted ensemble:

$$\hat{y}_n^{(k)} = \hat{y}_n^{(k-1)} + f^{(k)}(x_n)$$

- For the first tree, assume $\hat{y}_n^{(0)} = 0$, yielding

$$\hat{y}_n^{(1)} = f^{(1)}(x_n)$$

Exercise - Step 1: Gradients and Hessians (II)

- Knowing
  $\sigma(\hat{y}_n) = \left(1 + e^{-\hat{y}_n}\right)^{-1}, \quad G_n = \sigma(\hat{y}_n) - y_n, \quad H_n = \sigma(\hat{y}_n)(1 - \sigma(\hat{y}_n))$
- Compute once before growing each tree:

n	X_1	X_2	y	$\hat{y}^{(0)}$	$\sigma(\hat{y}^{(0)})$	G	H
1	1	2	0	0	0.5	0.5	0.25
2	2	1	0	0	0.5	0.5	0.25
3	3	2	0	0	0.5	0.5	0.25
4	1	3	1	0	0.5	-0.5	0.25
5	2	2	1	0	0.5	-0.5	0.25
6	3	3	1	0	0.5	-0.5	0.25

Exercise - Step 2: Enumerate split rules

- For first feature $m = 1$
  - Unique sorted values $\mathcal{V}_1 = \{1, 2, 3\}$
  - Rules $[x_{:,1}; 1.5]$ and $[x_{:,1}; 2.5]$
- For second feature $m = 2$:
  - Unique sorted values $\mathcal{V}_2 = \{1, 2, 3\}$
  - Rules $[x_{:,2}; 1.5]$ and $[x_{:,2}; 2.5]$
- In the beginning there is only the root $j = 1$, where:
  - All instances belong to the root: $I_1 = \{1, 2, 3, 4, 5, 6\}$
- Which rule $[x_{:,1}; 1.5]$, $[x_{:,1}; 2.5]$, $[x_{:,2}; 1.5]$, $[x_{:,2}; 2.5]$ maximizes the gain of splitting the root?

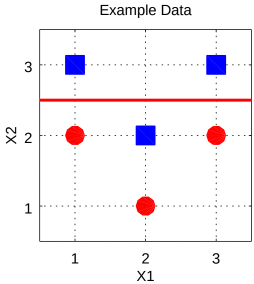
Exercise - Step 3: Best split rule = Maximal Gain

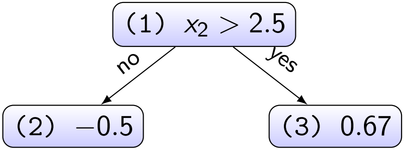
n	X_1	X_2	y	$\hat{y}^{(0)}$	$\sigma(\hat{y}^{(0)})$	G	H
1	1	2	0	0	0.5	0.5	0.25
2	2	1	0	0	0.5	0.5	0.25
3	3	2	0	0	0.5	0.5	0.25
4	1	3	1	0	0.5	-0.5	0.25
5	2	2	1	0	0.5	-0.5	0.25
6	3	3	1	0	0.5	-0.5	0.25

- Rule $[x_{:,1}; 1.5]$:
  - $I_1^{(\text{Left})} = \{1, 4\}$ and $I_1^{(\text{Right})} = \{2, 3, 5, 6\}$, thus $Gain_1 = -1$
- Rule $[x_{:,1}; 2.5]$:
  - $I_1^{(\text{Left})} = \{1, 2, 4, 5\}$ and $I_1^{(\text{Right})} = \{3, 6\}$, thus $Gain_1 = -1$
- Rule $[x_{:,2}; 1.5]$:
  - $I_1^{(\text{Left})} = \{2\}$ and $I_1^{(\text{Right})} = \{1, 3, 4, 5, 6\}$, thus $Gain_1 = -0.84$
- Rule $[x_{:,2}; 2.5]$:
  - $I_1^{(\text{Left})} = \{1, 2, 3, 5\}$ and $I_1^{(\text{Right})} = \{4, 6\}$, thus $Gain_1 = -0.41$ (best)

Our first tree with depth 1!

- The best rule we found $[x_{:,2}; 2.5]$:
  - Splits node ($j = 1$) into $I_1^{(\text{Left})} = \{1, 2, 3, 5\}$ and $I_1^{(\text{Right})} = \{4, 6\}$
  - Left child ($j = 2$) with weight $w_2 = -\frac{G_1+G_2+G_3+G_5}{H_1+H_2+H_3+H_5+\lambda} = -0.5$
  - Right child ($j = 3$) with weight $w_3 = -\frac{G_4+G_6}{H_4+H_6+\lambda} = 0.66$





- Interpretation of the outcome $y_n^{(1)} = f^{(1)}(x_n) = w_{q(x_n)}$:
  - $\sigma(\hat{y}_n^{(1)}) = \sigma(-0.5) = 0.37, \ \forall n \in \{1, 2, 3, 5\}, \ q(x_n) = 2$
  - $\sigma(\hat{y}_n^{(1)}) = \sigma(0.67) = 0.66, \ \forall n \in \{4, 6\}, \ q(x_n) = 3$

Grow the tree further

- Follow the same procedure to compute the best rules for further splitting node ($j = 2$) and ($j = 3$)
- Proceed until the maximum allowed depth is reached.
- For subsequent trees in the ensemble follow the same procedure, but note that:
  - For the first tree $\hat{y}_n^{(0)} = 0$
  - For the second tree $\hat{y}_n^{(1)} = f^{(1)}(x_n)$
  - For the third tree $\hat{y}_n^{(2)} = f^{(1)}(x_n) + f^{(2)}(x_n)$, etc ...
- Finish the exercise at home!