Decision Tree

2020年11月16日 13:40

Pros: good explanations (vs. Deep NNs)

Cons: · Big & Deep trees
· Overfit easily.
· Hard switch paths

Formalization:

$f: \{-1,1\}^n \to [-1,1]$

Two functions are close if: $\mathop{\mathbb{E}}_{x\sim D}[f(x)-g(x)]^2 \le \varepsilon$.

Fourier Basis: $\chi_S(x) = \prod_{i\in S} x_i$ for $S \subset [n]$

2^n of them

construct an orthonormal basis for Boolean function

$\Rightarrow f(x) = \sum_{S\subseteq[n]} \hat{f}_S \chi_S(x)$.

where $\hat{f}_S = \langle f(x), \chi_S(x)\rangle \overset{\text{def}}{=} \mathop{\mathbb{E}}_{x\sim D}[f(x)\cdot\chi_S(x)]$

Def L_1 norm $L_1(f) = \sum_S |\hat{f}_S|$.

Sparsity $L_0(f) = |\{S \mid \hat{f}_S \neq 0\}|$

Parseval's identity $\mathop{\mathbb{E}}_{x\sim D}[f(x)^2] = \sum_S \hat{f}_S^2$

Theorem

For any decision tree T with s leaf nodes, there exists a degree-$\log(\frac{s}{\varepsilon})$ and sparsity-$(\frac{s}{\varepsilon})^2$ function h that 2ε-approximates T.

Step 1. Truncate the tree at depth $\log(\frac{s}{\varepsilon})$.

- There are $2^{\log\frac{s}{\varepsilon}} = \frac{s}{\varepsilon}$ nodes
- By union bound, at most $\frac{\varepsilon}{s}\cdot s = \varepsilon$ leaves cut

$\Rightarrow$ Now assume T height $\le \log(\frac{s}{\varepsilon})$

Step 2. T with s leaves represented by f: $L_1(f) = s$ and degree $\log(\frac{s}{\varepsilon})$

- A tree with s leaf nodes can be represented by 最小项之和
- Each 最小项 $L_1 \le 1$

$\Rightarrow L_1(f) \le s$

$\Rightarrow$ Degree $\le$ height $= \log(\frac{s}{\varepsilon})$

Step 3. For f s.t. $L_1 \le s$. deg $\le \log(\frac{s}{\varepsilon})$. $\exists\, h$ with $L_0 \le \frac{s^2}{\varepsilon}$, deg $= \log(\frac{s}{\varepsilon})$ s.t.

$$\mathop{\mathbb{E}}_{X \sim D} (h(x) - f(x))^2 \le \varepsilon$$

Let h include all terms in

$$\Lambda = \{ s \mid |\hat{f}_s| \ge \frac{\varepsilon}{L_1(f)} \}$$

h has terms at most

$$\frac{L_1(f)}{\varepsilon / L_1(f)} = \frac{L_1(f)^2}{\varepsilon} \le \frac{s^2}{\varepsilon}$$

Meanwhile,

$$\sum_{S \notin \Lambda} (\hat{f}_S)^2 \le \max \hat{f}_S \cdot \sum_{S \notin \Lambda} |\hat{f}_S| \le \frac{\varepsilon}{L_1(f)} \cdot L_1(f) = \varepsilon$$

KM algorithm.

- prune less promising branches
- f_α the sum of all Fourier basis started by α

e.g. $f_{110} = \hat{f}_{x_1x_2x_4} \cdot x_1x_2x_4 + \hat{f}_{x_1x_2} \cdot x_1x_2$

Algo.

Def Coef(α):

If $\mathbb{E}(f_\alpha^2) \ge \theta^2$: (sampling, $\theta \triangleq \varepsilon$)

If $|\alpha| = n$:

outputs α

Else:

Coef($\alpha 0$), Coef($\alpha 1$)

LMN algorithm

LMN algorithm

Take m random uniform samples for f

Estimate:

$$\hat{f}_S \triangleq \langle f, \chi_S \rangle = \mathop{\mathbb{E}}_{x \sim D} [f(x)\chi_S(x)] \simeq \frac{\sum_{i=1}^{m} f(x_i)\chi_S(x_i)}{m}$$

Compressed Sensing

$y = Ax + e$

y: measurement; e: error

x is s e in χ_S basis $(\frac{s^2}{\epsilon})$.

How to pick random varibles ?

- Gini index

If labels in n classes.

$$Gini = 1 - \sum_{i=1}^{n} P_i^2 \quad \in [0,1)$$

$P_i = Pr[$ left branch & result $= i]$

$Gini = Pr[left] \cdot Gini(left) + Pr[right] \cdot Gini(right)$

Gini index $= 0 \rightarrow$ all in one class

pick the varible with smallest Gini

Overfitting $\Longrightarrow$ Random Forest (Most practical)

- Bagging

$(X_1, y_1), \cdots, (X_n, y_n)$ instead of use it directly,

we sample it n times with replacement $\Rightarrow X_b \quad b = 1, 2, \cdots, B$

Each Element: miss probability $(1 - \frac{1}{n})^n \simeq \frac{1}{e}$

Get B trees (forest) using B dataset

$$\Rightarrow \hat{f}(X_{test}) = \frac{1}{B} \sum_{b=1}^{B} f_b(X_{test})$$

- Feature Bagging (ensemble)

Also do bagging on features

Each tree only use a random subset of features

Forced to use all features to learn !

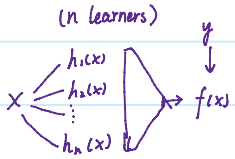
- Boosting

(n learners)

Given a set of features to learn:

- Boosting

Combine weak learners to build a strong one → other possible methods:
- Stacked Learner



Modify the sample likelihood of data with wrong prediction

Framework: for $t=1$ to T

construct distribution D_t on $\{1,\cdots,m\}$

Find classifier h_t (weak), error $\varepsilon_t = Pr_{i\sim D_t}[h_t(x_i)\neq y_i]$

Output final classifier H_{final}

e.g. Adaboost. $D_1(i)=\frac{1}{m}$

Given $D_t, h_t \Rightarrow D_{t+1}(i)= \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$

Z_t = normalize const

$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$

- $H_{final} = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$

Theorem: write $\varepsilon_t = \frac{1}{2} - \gamma_t$

$\Rightarrow Error_{train}(H_{final}) \leq \prod_t 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \leq e^{-2\sum_t \gamma_t^2}$

Proof:

$$Error = \frac{1}{m}\sum_i \begin{cases} 1 & \text{if } y_i \neq H_{final}(x_i) \\ 0 & \text{else} \end{cases}$$

$$= \frac{1}{m}\sum_i \begin{cases} 1 & y_i f(x_i) \leq 0 \\ 0 & \text{else} \end{cases}$$

$$\leq \frac{1}{m}\sum_i e^{-y_i f(x_i)} = \sum_i D_{final}(i)\prod_t Z_t = \prod_t Z_t$$

$$D_{final}(i) = \frac{1}{m}\frac{e^{-y_i\sum_t \alpha_t h_t(x_i)}}{\prod_t Z_t} = \frac{1}{m}\frac{e^{-y_i f(x_i)}}{\prod_t Z_t}$$

$$Z_t = \sum_i D_t(i)\, e^{-\alpha_t y_i h_t(x_i)}$$

$$= \varepsilon_t e^{\alpha_t} + (1-\varepsilon_t)e^{-\alpha_t}$$

$$= 2\sqrt{\varepsilon_t(1-\varepsilon_t)}$$

$\Rightarrow \varepsilon_t = \frac{1}{2} - \gamma_t \Rightarrow Z_t = \sqrt{1-4\gamma_t^2} \leq e^{-2\gamma_t^2}$

$\Rightarrow Error \leq e^{-2\sum_t \gamma_t^2}$

Generalization