決定木とアンサンブル学習

中田和秀

東京科学大学 工学院 経営工学系

機械学習入門

https://www.nakatalab.iee.e.titech.ac.jp/text/nakata.html

概要

ここでは最初に単純な機械学習法である決定木について説明する。その後、複数の決定木を組み合わせることにより複雑で高精度な予測を行うアンサンブル学習について説明をする。

目次：

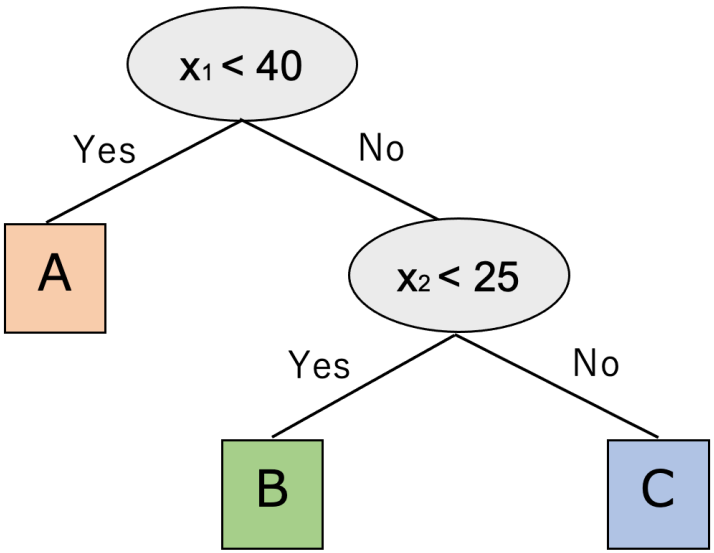
記号の使い方：

- $A := B$ は、B で A を定義する、B を A に代入することを意味する
- $[n]$ は n までのインデックスの集合を表し $[n] := \{1, 2, \cdots, n\}$

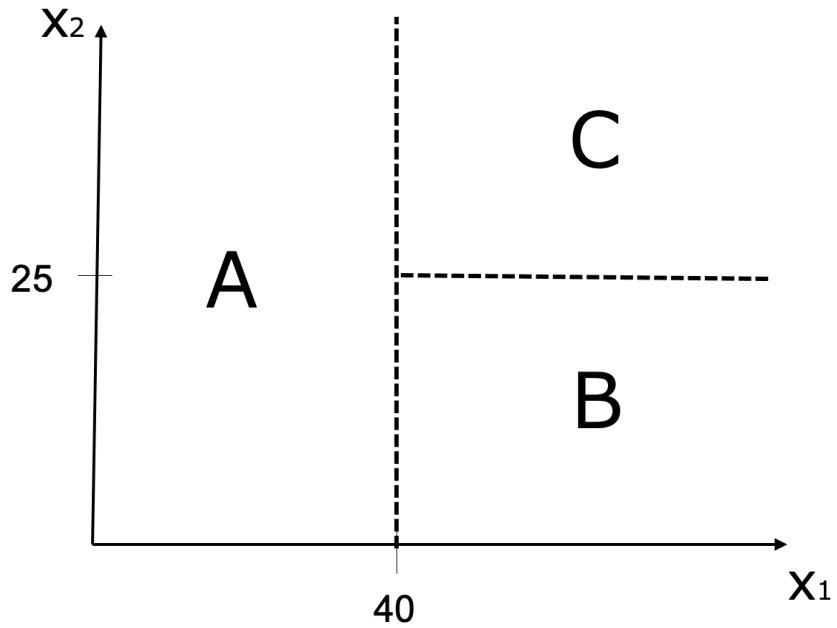
決定木

予測器

- 何かの条件に従って分岐する
- 適当な深さまで、上の作業を再帰的に繰り替えす
- 分岐が終わったとき、その領域に入ったものに一つの値を返す。



決定木



上のような領域分割に相当する

決定木の学習

学習：

訓練データに対し、できるだけ正確に判別・回帰を行う決定木を構築する

- 離散的な構造（木）を決める必要があり、パラメタ $\boldsymbol{\theta}$ は連続量ではないため、微分情報を使えない。
- 何かの基準に対して、凸性も期待できない。

→ 最適な決定木を構築するのは困難

貪欲法 (Greedy Algorithm) を用いて、近似解を計算する。

- 各ノードにおいて、そこにあるデータの２分割を最適化する。
- 各ノードでは、「一つ」の特徴量を使って分岐ルールを決める。

  例： $\boldsymbol{x} \in \mathbb{R}^n$ に対し、

  もし $x_i \geq c$ ならば右の子ノード、そうでなければ左の子ノード

分岐ルールでは、使う特徴量 i と閾値 c を決める必要がある

分岐ルール

分岐ルールの決め方

- 分岐に使う候補となる特徴量 i を順に選ぶ。　　$i \in [n]$
- 閾値の候補 c を順に選ぶ。
  1. ノードに含まれるデータの特徴量 i を抜き出し、昇順に並び替える。

$$\alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_k$$

  2. 中間の値を閾値 c の候補とする。

$$c := \frac{\alpha_1 + \alpha_2}{2}, \frac{\alpha_2 + \alpha_3}{2}, \cdots, \frac{\alpha_{k-1} + \alpha_k}{2}$$

- 特徴量 i と閾値 c の組に対し、
  「分割の評価値」が最も良かったものを分岐ルールとして採用する。

分岐ルールの候補数は $n(k-1)$

「分割の評価値」は回帰と判別に分けて、次スライド以降で説明

回帰木

あるノードにあるデータの予測値：　　ノードにあるデータの集合を $\mathcal{D}$ とする

$$\text{平均}\quad \hat{y} := \frac{1}{|\mathcal{D}|}\sum_{d\in\mathcal{D}} y_d$$

評価値

２乗誤差を使うことが多い

$$\sum_{d\in[D]} (y_d - \hat{y}_d)^2$$

集合 $\mathcal{D}_1, \mathcal{D}_2$ に分けたときの評価値：

$$\sum_{d\in\mathcal{D}_1} (y_d - \hat{y}_1)^2 + \sum_{d\in\mathcal{D}_2} (y_d - \hat{y}_2)^2$$

ただし、$\hat{y}_1 := \dfrac{1}{|\mathcal{D}_1|}\displaystyle\sum_{d\in\mathcal{D}_1} y_d$，　$\hat{y}_2 := \dfrac{1}{|\mathcal{D}_2|}\displaystyle\sum_{d\in\mathcal{D}_2} y_d$

分類木

あるノードにあるデータの予測値 $\hat{y}$： 多数決で決める

評価値

次のような不純度が使われる。

クラス k の割合を r_k とする

- 誤判別率： $1 - \max_k r_k$
- Gini 係数： $\sum_k r_k(1 - r_k)$
- クロスエントロピー誤差： $-\sum_k r_k \log r_k$

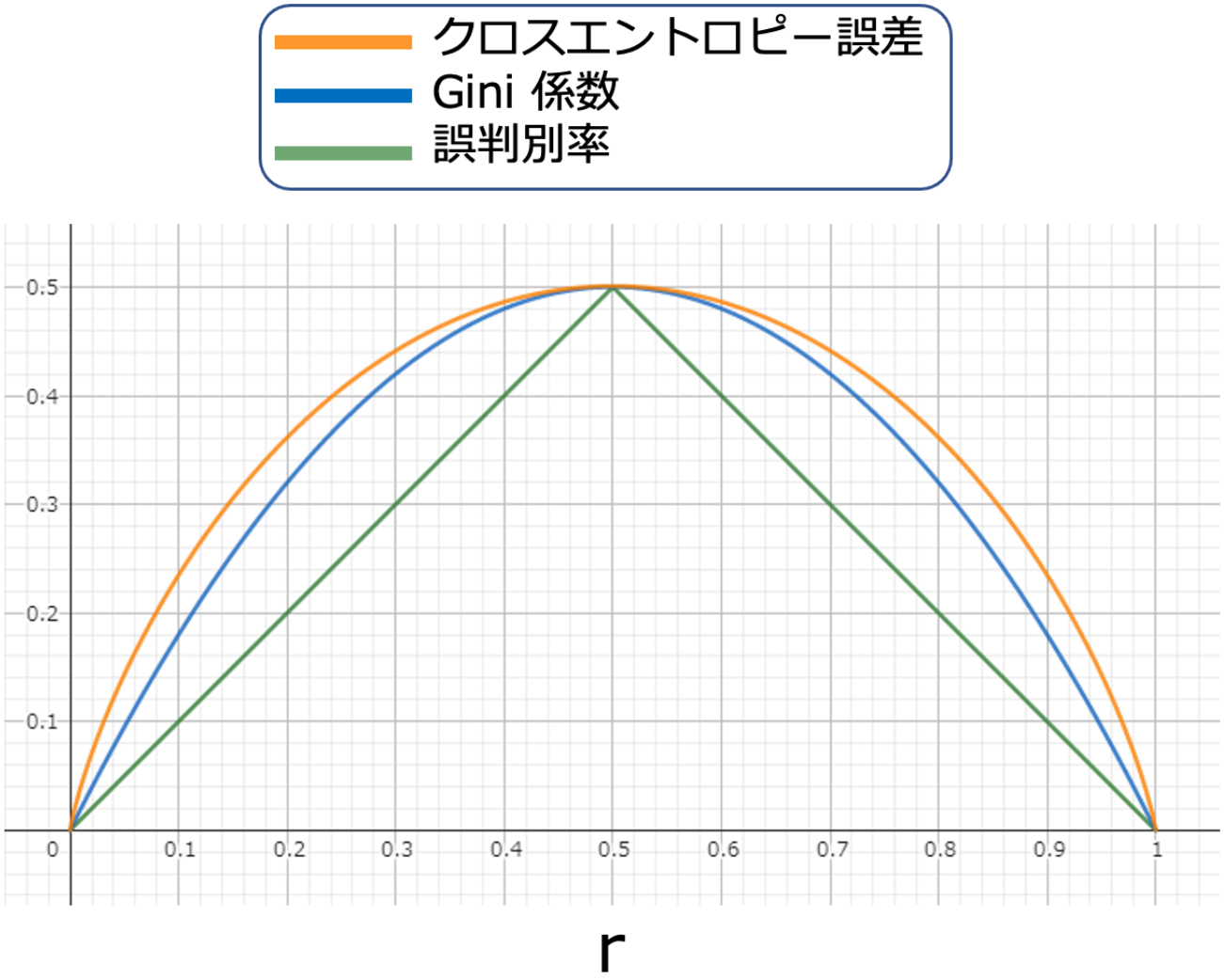
ノードにあるデータの集合を $\mathcal{D}$ としたとき、このデータの不純度を $\mathrm{imp}_{\mathcal{D}}$ とする。

集合 $\mathcal{D}_1$, $\mathcal{D}_2$ に分けたときの評価値：

$$\frac{|\mathcal{D}_1|}{|\mathcal{D}_1| + |\mathcal{D}_2|} \times \mathrm{imp}_{\mathcal{D}_1} + \frac{|\mathcal{D}_2|}{|\mathcal{D}_1| + |\mathcal{D}_2|} \times \mathrm{imp}_{\mathcal{D}_2}$$

不純度

2 クラスの場合の r_i と不純度の関係



Gini 係数を使うことが多い。

アンサンブル学習

アンサンブル学習：

複数の弱学習器から 1 つの学習器を生成する手法

例：2 値判別の場合

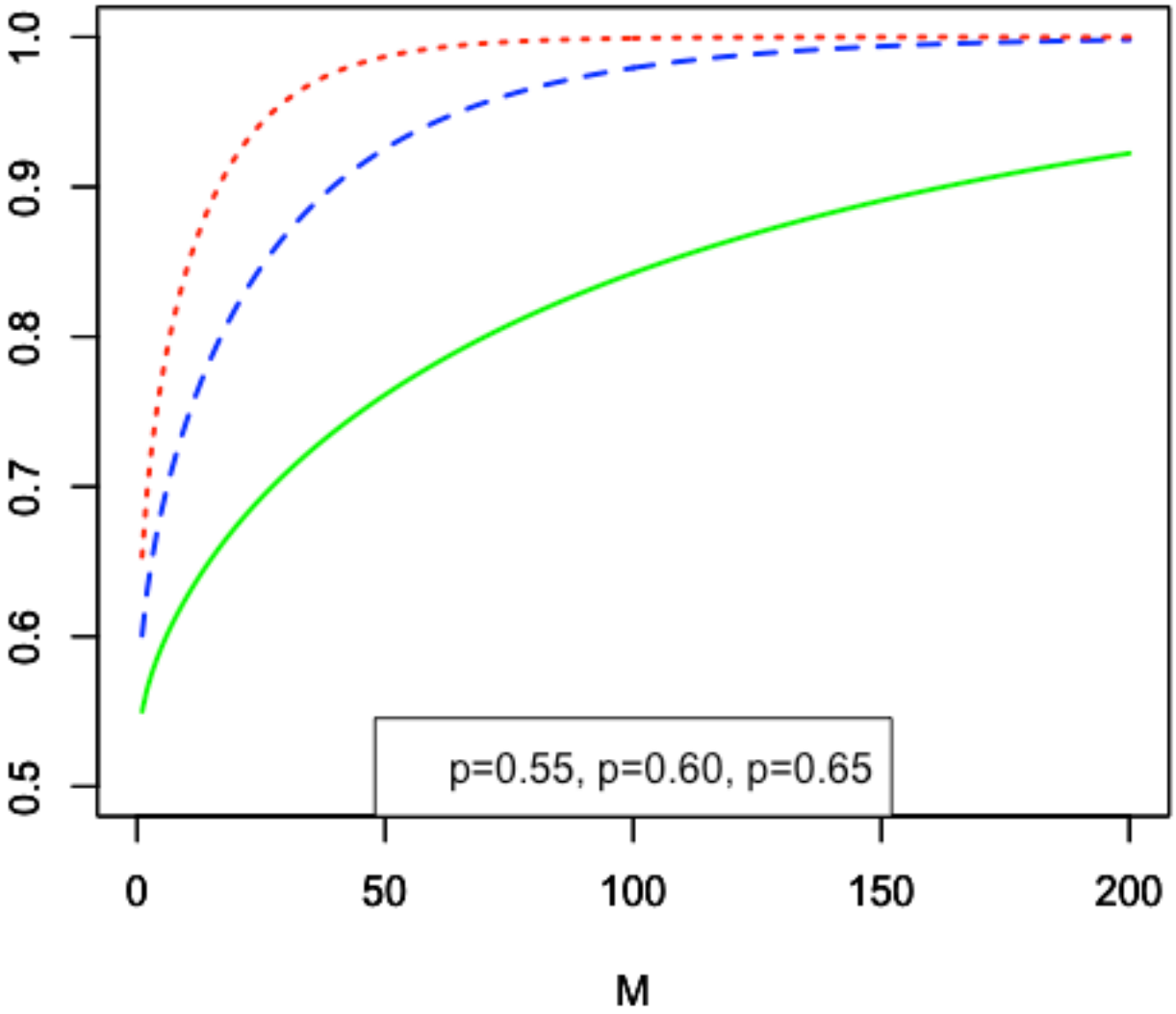
多数決で決める

- M 個の独立な弱学習器がある
- 各学習器の正解率を p とする　$(0.5 < p < 1)$
- 正解が導ける確率は中心極限定理を使うと $\Phi\left(\sqrt{M}\dfrac{p-0.5}{\sqrt{p(1-p)}}\right)$ 程度

  ただし、Φ は標準正規分布に対する分布関数
- M が大きくなるにつれ正解率は上昇する

多数決による正解率

横軸：学習器数、　縦軸：正解率



回帰の場合

平均を取る　$F(\boldsymbol{x}) := \dfrac{1}{M}\displaystyle\sum_{m=1}^{M} f_m(\boldsymbol{x})$

- $f_m(\boldsymbol{x})$ $(m \in [M])$ は独立な弱学習器
- 弱学習器 f_m $(m \in [M])$ の分散を $\mathbb{V}[f]$ とする。
- M が大きくなるほど $F(\boldsymbol{x})$ の分散は小さくなる。

$$\mathbb{V}[F] = \frac{1}{M}\mathbb{V}[f]$$

ただし、判別でも回帰でも独立な予測器をたくさん用意することは難しい

バギング

ブートストラップ法によって学習データを生成することで、多様な予測器を構築

バギング

1. $m := 1$
2. $\mathcal{D} := \{\boldsymbol{x}_d, y_d\}_{d \in [D]}$ からのサンプリング（復元抽出）によって、新しい訓練データを $\mathcal{D}_m$ を生成
3. 訓練データ $\mathcal{D}_m$ を使って学習し、 弱学習器 $f_m(\boldsymbol{x})$ を構築
4. $m < M$ ならば、$m := m + 1$ として (2) に戻る

M 個の弱学習器の結果をまとめた予測器を出力

- 判別：M 個の弱学習器の多数決
- 回帰：M 個の弱学習器の平均

※ 例えば決定木で弱学習器を構築する

ランダムフォレスト

学習データをサンプリングするだけでなく、特徴量もサンプリングすることによって、各学習器の相関を低くする

ランダムフォレスト

1. $m := 1$
2. $\mathcal{D} := \{\boldsymbol{x}_d, y_d\}_{d \in [D]}$ からのサンプリング（復元抽出）によって、新しい訓練データを $\mathcal{D}_m$ を生成
3. 学習に利用する特徴量をランダムに選択
4. 訓練データ $\mathcal{D}_m$ に対し、上で選んだ特徴量のみを使って学習し、決定木 $f_m(\boldsymbol{x})$ を構築
5. $m < M$ ならば、$m := m + 1$ として (2) に戻る

M 個の弱学習器の結果をまとめた予測器を出力

- 判別：M 個の弱学習器の多数決
- 回帰：M 個の弱学習器の平均

ブースティング

- 予測性能が向上するように弱学習器を「逐次」追加する
- 新しい弱学習器を学習する際、既に得られた予測器の「不正確さ」を反映

弱学習器を $f_m(\boldsymbol{x})$ $(m \in [M])$ としたとき、

予測器： $F(\boldsymbol{x}) := \sum_{m=1}^{M} f_m(\boldsymbol{x})$

バギングの弱学習器が「並列」だとすると、ブースティングの弱学習器は「直列」

フレームワーク

通常の学習

$$\min_{f \in \mathcal{F}} \frac{1}{D} \sum_{d \in [D]} L(y_d, f(\boldsymbol{x}_d))$$

$\mathcal{F} := \{f(\boldsymbol{x}; \boldsymbol{\theta}) \mid \exists \boldsymbol{\theta}\}$　予測器の集合

ブースティング

1. $F_0(\boldsymbol{x}) := 0,\ m = 1$
2. 次で表される学習を行い f_m を構築。
$$\min_{f_m \in \mathcal{F}} \sum_{d \in [D]} L(y_d, F_{m-1}(\boldsymbol{x}_d) + f_m(\boldsymbol{x}_d))$$
3. $F_m(\boldsymbol{x}) := F_{m-1}(\boldsymbol{x}) + f_m(\boldsymbol{x})$
4. 終了条件を満たさなければ、$m := m + 1$ として (2) へ

以下では、AdaBoost と勾配ブースティングについて説明。

AdaBoost

２値判別問題を考える。

$$\{\boldsymbol{x}_d, y_d\}_{d\in[D]} \qquad \boldsymbol{x}_d \in \mathbb{R}^n,\ y_d \in \{+1, -1\}$$

誤差関数

指数損失を用いる

$$L(y, \hat{y}) := \exp\{-y\hat{y}\}$$

予測器

弱学習器： $\alpha_i f_i(\boldsymbol{x})$ $\quad (f_i(\boldsymbol{x}) \in \{+1, -1\},\ \alpha_i \in \mathbb{R})$

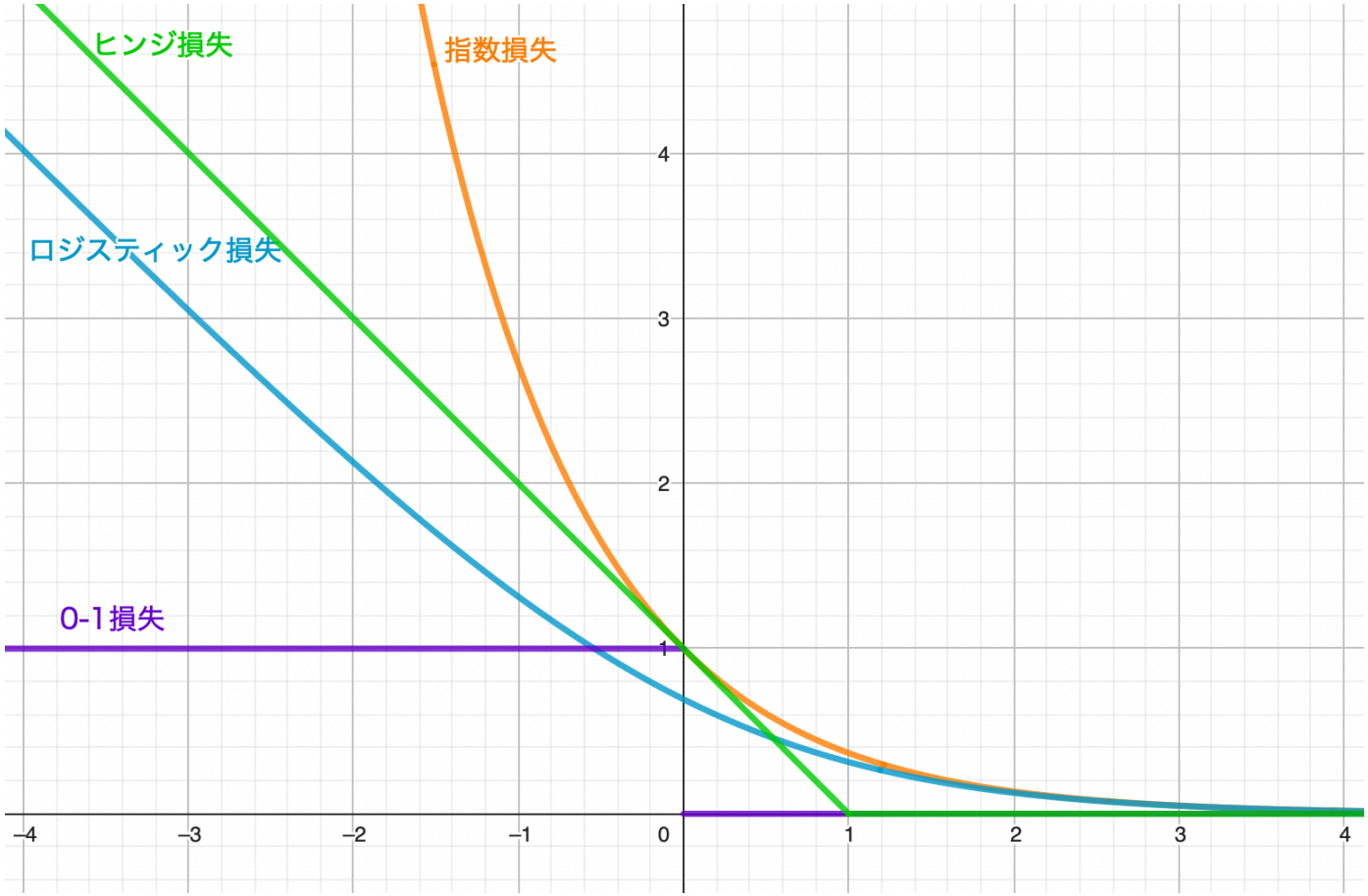
※ $f_i(\boldsymbol{x})$ は $\{+1, -1\}$ を返す予測器であることに注意

m 段階目の予測器： $\hat{y} = F_m(\boldsymbol{x}) := \sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{x})$

指数損失

$L(y, \hat{y}) := \exp(-y\hat{y})$

$y = +1$ の場合



指数損失について

$\boldsymbol{x}$ を固定して、指数損失の期待値 $\mathbb{E}[L(y, \hat{y})]$ を最小にする予測 $\hat{y}$ を考える。

最小となる $\hat{y}^*$ は、

$$\hat{y}^* = \frac{1}{2} \log \frac{P(y = +1|\boldsymbol{x})}{P(y = -1|\boldsymbol{x})}$$

オッズ比の半分

つまり、学習によって得られる予測器 $F(\boldsymbol{x})$ は、オッズ比（の半分）を予測していると捉えることができる

$F(\boldsymbol{x})$ に対し 0 を閾値として、判別をすればよい

- $P(y = +1|\boldsymbol{x}) > P(y = -1|\boldsymbol{x}) \iff \hat{y} > 0$
- $P(y = +1|\boldsymbol{x}) = P(y = -1|\boldsymbol{x}) \iff \hat{y} = 0$
- $P(y = +1|\boldsymbol{x}) < P(y = -1|\boldsymbol{x}) \iff \hat{y} < 0$

式変形

$p := P(y = +1|\boldsymbol{x})$ とすると、$P(y = -1|\boldsymbol{x}) = 1 - p$

このとき、1 階微分と 2 階微分は次の通りになる。

$$\mathbb{E}[L(y, \hat{y})] = p \exp\{-\hat{y}\} + (1 - p) \exp\{\hat{y}\}$$

$$\frac{\partial \mathbb{E}[L(y, \hat{y}])}{\partial \hat{y}} = -p \exp\{-\hat{y}\} + (1 - p) \exp\{\hat{y}\}$$

$$\frac{\partial^2 \mathbb{E}[L(y, \hat{y})]}{\partial \hat{y}^2} = p \exp\{-\hat{y}\} + (1 - p) \exp\{\hat{y}\} > 0$$

2 階微分が正より凸関数であり、1 階微分が 0 となる $\hat{y}$ が最小解である。よって、

$$\frac{\partial \mathbb{E}[L(y, \hat{y})]}{\partial \hat{y}} = -p \exp\{-\hat{y}\} + (1 - p) \exp\{\hat{y}\} = 0$$

$$\iff \hat{y} = \frac{1}{2} \log \frac{p}{1 - p}$$

学習

m 段階目の学習：

$$\min_{f_m \in \mathcal{F}, \alpha_m \in \mathbb{R}} \sum_{d \in [D]} L\left(y_d, \sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{x}_d)\right)$$

f_m と α_m の最適化を行う。

目的関数を E とおく。

$$E := \sum_{d \in [D]} L\left(y_d, \sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{x}_d)\right) = \sum_{d \in [D]} \exp\left\{-y_d \sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{x}_d)\right\}$$

$$= \exp\{-\alpha_m\} \sum_{d: y_d = f_m(\boldsymbol{x}_d)} w_d \ + \ \exp\{\alpha_m\} \sum_{d: y_d \neq f_m(\boldsymbol{x}_d)} w_d$$

ただし、 $w_d := \exp\left\{-y_d \sum_{i=1}^{m-1} \alpha_i f_i(\boldsymbol{x}_d)\right\}$

式変形

$$
\begin{aligned}
E &:= \sum_{d\in[D]} L\left(y_d, \sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{x}_d)\right) \\
&= \sum_{d\in[D]} \exp\left\{-y_d \sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{x}_d)\right\} \\
&= \sum_{d\in[D]} \exp\left\{-y_d \sum_{i=1}^{m-1} \alpha_i f_i(\boldsymbol{x}_d)\right\} \exp\left\{-y_d \alpha_m f_m(\boldsymbol{x}_d)\right\}
\end{aligned}
$$

$w_d := \exp\left\{-y_d \sum_{i=1}^{m-1} \alpha_i f_i(\boldsymbol{x}_d)\right\}$ とする

$$
\begin{aligned}
&= \sum_{d\in[D]} w_d \exp\{-y_d \alpha_m f_m(\boldsymbol{x}_d)\} \\
&= \exp\{-\alpha_m\} \sum_{d:y_d=f_m(\boldsymbol{x}_d)} w_d \;+\; \exp\{\alpha_m\} \sum_{d:y_d\neq f_m(\boldsymbol{x}_d)} w_d
\end{aligned}
$$

f_m の学習

$$\min_{f_m \in \mathcal{F}, \alpha_m \geq 0} \exp\{-\alpha_m\} \sum_{d \in [D]} w_d + \{\exp\{\alpha_m\} - \exp\{-\alpha_m\}\} \sum_{d: y_d \neq f_m(\boldsymbol{x}_d)} w_d$$

- f_m は $\sum_{d: y_d \neq f_m(\boldsymbol{x}_d)} w_d$ にのみ関係
- α_m に依存せず f_m を最適化できる

重み付き 0–1 誤差最小化

$$\min_{f_m \in \mathcal{F}} \sum_{d \in [D]} w_d \; L_{0,1}(y_d, f_m(\boldsymbol{x}_d))$$

$y_d, f_m(\boldsymbol{x}_d) \in \{+1, -1\}$, 決定木で学習できる

0–1 誤差関数： $L_{0,1}(y, \hat{y}) := \begin{cases} 0 & (y = \hat{y}) \\ 1 & (y \neq \hat{y}) \end{cases}$

- 判別に失敗しているデータを優先的に判別する弱学習器

α_m の学習

$$E = \exp\{-\alpha_m\} \sum_{d:y_d=f_m(\boldsymbol{x}_d)} w_d + \exp\{\alpha_m\} \sum_{d:y_d\neq f_m(\boldsymbol{x}_d)} w_d$$
$$= w_0 \exp\{-\alpha_m\} + w_1 \exp\{\alpha_m\}$$

ただし、$w_0 := \sum_{d:y_d=f_m(\boldsymbol{x}_d)} w_d$, $w_1 := \sum_{d:y_d\neq f_m(\boldsymbol{x}_d)} w_d$

これを最小にする α_m^* は、

$$\alpha_m^* = \frac{1}{2} \log \frac{w_0}{w_1}$$

正常に学習ができている場合、$w_0 > w_1$ より $\alpha_m^* > 0$

式変形

$$E := \ w_0 \exp\{-\alpha_m\} \ + \ w_1 \exp\{\alpha_m\}$$

より、

$$\frac{\partial E}{\partial \alpha_m} = -w_0 \exp\{-\alpha_m\} + w_1 \exp\{\alpha_m\}$$
$$\frac{\partial^2 E}{\partial \alpha_m^2} = +w_0 \exp\{-\alpha_m\} + w_1 \exp\{\alpha_m\}$$

2 階微分が正より凸関数であり、1 階微分が 0 となる $\hat{y}$ が最小解である。よって、

$$\begin{aligned} & - w_0 \exp\{-\alpha_m\} + w_1 \exp\{\alpha_m\} = 0 \\ \iff & \quad \exp 2\alpha_m = \frac{w_0}{w_1} \\ \iff & \quad \alpha_m = \frac{1}{2} \log \frac{w_0}{w_1} \end{aligned}$$

w_d の更新

$$w_d := \exp\left\{-y_d \sum_{i=1}^{m} \alpha_i f_i(\boldsymbol{x}_d)\right\} \text{ の計算}$$

$$d \in [D], \quad w_d := \begin{cases} w_d \exp\{-\alpha_m\} & (y_d = f_m(\boldsymbol{x}_d)) \\ w_d \exp\{\alpha_m\} & (y_d \neq f_m(\boldsymbol{x}_d)) \end{cases}$$

w_d を定数倍しても学習は変わらないため、次のように更新

$$d \in [D], \quad w_d := \begin{cases} w_d & (y_d = f_m(\boldsymbol{x}_d)) \\ w_d \exp\{2\alpha_m\} & (y_d \neq f_m(\boldsymbol{x}_d)) \end{cases}$$

- 判別に失敗したデータに対して重みを大きくする
- 全体的に学習がうまく行っていると α_m は大きくなるため、補正も大きくなる

AdaBoost

AdaBoost

1. $w_d := 1/D \ (d \in [D]), \quad m := 1$
2. 重み付き決定木で学習し f_m を構築。

   $\{\boldsymbol{x}_d, y_d, w_d\}_{d\in[D]}, \qquad \min_{f_m \in \mathcal{F}} \sum_{d\in[D]} w_d \, L_{0,1}(y_d, f_m(\boldsymbol{x}_d))$

   また、最適値を A とする
3. $w_0 := \sum_{d:y_d = f_m(\boldsymbol{x}_d)} w_d, \quad w_1 := \sum_{d:y_d \neq f_m(\boldsymbol{x}_d)} w_d, \quad \alpha_m = \frac{1}{2} \log \frac{w_0}{w_1}$
4. $w_d := w_d \exp\{2\alpha_m\} \quad (y_d \neq f_m(\boldsymbol{x}_d))$
5. 終了条件を満たせば $F(\boldsymbol{x}) := \sum_{i\in[m]} \alpha_m f_m(\boldsymbol{x})$ を出力。

   そうでなければ、$m := m + 1$ として (2) に戻る。

勾配ブースティング

学習 $\min_{f_m \in \mathcal{F}} \sum_{d \in [D]} L(y_d, F_{m-1}(\boldsymbol{x}_d) + f_m(\boldsymbol{x}_d))$ が簡単に出来ない場合

$L(y, \hat{y})$ に対する１階微分 $\dfrac{\partial L}{\partial \hat{y}}$、２階微分 $\dfrac{\partial^2 L}{\partial \hat{y}^2}$ を使う

- 一次近似

  $$L(y_d, F_{m-1}(\boldsymbol{x}_d) + f_m(\boldsymbol{x}_d)) \approx L(y_d, F_{m-1}(\boldsymbol{x}_d)) + g_d f_m(\boldsymbol{x}_d)$$

  ただし、 $g_d := \left[\dfrac{\partial L}{\partial \hat{y}}\right]_{\hat{y}=F_{m-1}(\boldsymbol{x}_d)}$

- 二次近似

  $$L(y_d, F_{m-1}(\boldsymbol{x}_d) + f_m(\boldsymbol{x}_d)) \approx L(y_d, F_{m-1}(\boldsymbol{x}_d)) + g_d f_m(\boldsymbol{x}_d) + \frac{1}{2} h_d f_m(\boldsymbol{x}_d)^2$$

  ただし、 $g_d := \left[\dfrac{\partial L}{\partial \hat{y}}\right]_{\hat{y}=F_{m-1}(\boldsymbol{x}_d)}$, $h_d := \left[\dfrac{\partial^2 L}{\partial \hat{y}^2}\right]_{\hat{y}=F_{m-1}(\boldsymbol{x}_d)}$

一次近似の利用

$$L(y_d, F_{m-1}(\boldsymbol{x}_d) + f_m(\boldsymbol{x}_d)) \approx L(y_d, F_{m-1}(\boldsymbol{x}_d)) + g_d f_m(\boldsymbol{x}_d)$$

一次近似の最小化は最急降下法とみなすことができる

- 勾配が g_d、 探索方向が $f_m(\boldsymbol{x}_d)$ に相当
- $f_m(\boldsymbol{x}_d) = -g_d \qquad (d \in [D])$
- データ点だけでなく、汎化性能を上げる必要がある

何かの誤差関数 $L_?(y, \hat{y})$ の元で学習

$$\min_{f_m \in \mathcal{F}} \sum_{d \in [D]} L_?(-g_d, f_m(\boldsymbol{x}_d))$$

ステップサイズの計算も必要

$$\min_{\alpha \geq 0} \sum_{d \in [D]} L(y_d, F_{m-1}(\boldsymbol{x}_d) + \alpha f_m(\boldsymbol{x}_d))$$

一次近似の利用

勾配ブースティング（一次）

1. $F_0(\boldsymbol{x}) := 0, \ \ m := 1$
2. $g_d := \left[\dfrac{\partial L}{\partial \hat{y}}\right]_{\hat{y}=F_{m-1}(\boldsymbol{x}_d)} \qquad (d \in [D])$
3. 次の学習を行い f_m を構築。
$$\min_{f_m \in \mathcal{F}} \sum_{d \in [D]} L_?(-g_d, f_m(\boldsymbol{x}_d))$$
4. 直線探索でステップサイズ α_m を計算。
$$\min_{\alpha_m \geq 0} \sum_{d \in [D]} L(y_d, F_{m-1}(\boldsymbol{x}_d) + \alpha_m f_m(\boldsymbol{x}_d))$$
5. $F_m(\boldsymbol{x}) := F_{m-1}(\boldsymbol{x}) + \alpha_m f_m(\boldsymbol{x})$
6. 終了条件を満たせば $F_m(\boldsymbol{x})$ を出力。
そうでなければ、$m := m + 1$ として (2) へ

二次近似の利用

$$L(y_d, F_{m-1}(\boldsymbol{x}_d) + f_m(\boldsymbol{x}_d)) \simeq L(y_d, F_{m-1}(\boldsymbol{x}_d)) + g_d f_m(\boldsymbol{x}_d) + \frac{1}{2} h_d f_m(\boldsymbol{x}_d)^2$$

二次近似の最小化は Newton 法とみなすことができる。

- $f_m(\boldsymbol{x}_d) = -\dfrac{g_d}{h_d} \qquad (d \in [D])$

このとき、何かの誤差関数 $L_?(y, \hat{y})$ の元で

$$\min_{f_m \in \mathcal{F}} \sum_{d \in [D]} L_? \left(-\frac{g_d}{h_d}, f_m(\boldsymbol{x}_d) \right)$$

を行う学習の代わりに、より直接的な次の学習を考える。

$$\min_{f_m \in \mathcal{F}} \sum_{d \in [D]} g_d f_m(\boldsymbol{x}_d) + \frac{1}{2} h_d f_m(\boldsymbol{x}_d)^2$$

これは、２乗誤差 $L_2(y, \hat{y}) := (y - \hat{y})^2$ を用いた重み付き学習と等しい。

$$\min_{f_m \in \mathcal{F}} \sum_{d \in [D]} h_d \; L_2 \left(-\frac{g_d}{h_d}, f_m(\boldsymbol{x}_d) \right)$$

勾配ブースティング（二次）

勾配ブースティング（二次）

1. $F_0(\boldsymbol{x}) := 0, \ \ m := 1$
2. $g_d := \left[\dfrac{\partial L}{\partial \hat{y}}\right]_{\hat{y}=F_{m-1}(\boldsymbol{x}_d)}, \ h_d := \left[\dfrac{\partial^2 L}{\partial \hat{y}^2}\right]_{\hat{y}=F_{m-1}(\boldsymbol{x}_d)} \quad (d \in [D])$
3. 次の学習を行い f_m を構築。 $\quad L_2(y, \hat{y}) := (y - \hat{y})^2$

$$\min_{f_m \in \mathcal{F}} \sum_{d \in [D]} h_d \ L_2\left(-\frac{g_d}{h_d}, f_m(\boldsymbol{x}_d)\right)$$

4. $F_m(\boldsymbol{x}) := F_{m-1}(\boldsymbol{x}) + f_m(\boldsymbol{x})$
5. 終了条件を満たせば $F_m(\boldsymbol{x})$ を出力。
   そうでなければ、$m := m + 1$ として (2) へ

勾配ブースティング決定木

学習には決定木を利用することが多い

- 学習時間が短い
- 適度な自由度（表現力）がある

勾配ブースティング決定木 (Gradient Boosting Decision Tree: GBDT) と呼ぶ

ブースティングは過学習を抑える工夫が重要

- 収縮
  $0 < \gamma \ll 1$ となる学習率を導入し、$F_m(\boldsymbol{x}) := F_{m-1}(\boldsymbol{x}) + \gamma\alpha_m f_m(\boldsymbol{x})$
- 正規化項の導入
- 木の自由度を抑制
  決定木の深さや葉の数などを制限する
- 特徴選択
  ランダムフォレストのように、特徴量をランダムに選択

GBDT の実装

$F_0(\boldsymbol{x}) := 0$ の代わりに、定数 $F_0(\boldsymbol{x}) := \operatorname{argmin}_{\gamma} \sum_{d \in [D]} L(y_d, \gamma)$ を使うことが多い。

計算量の削減

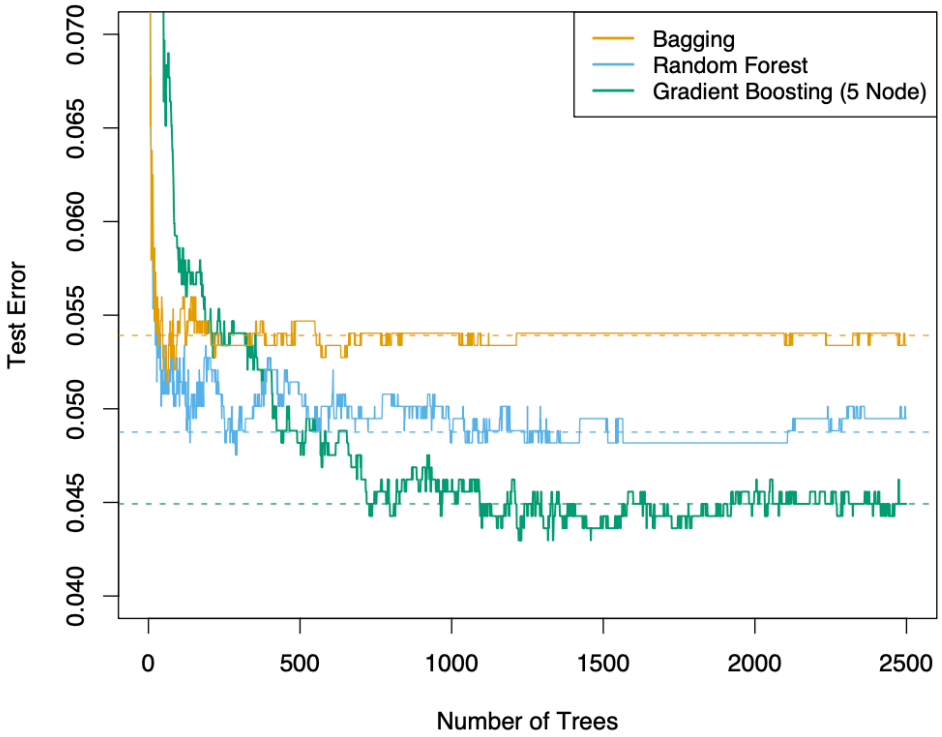
- データのスパース性の発見と利用
- 効率を上げる特徴選択
- キャッシュを利用した計算
- データセットのサンプリング　　など

ソフトウェア：

- XGBoost (eXtreme Gradient Boosting)
- LightGBM (Light Gradient Boosting Machine)
- CatBoost (Categorical Boosting)

高速に学習でき、ある程度の精度を出すことが多いため、よく使われる。

数値実験例



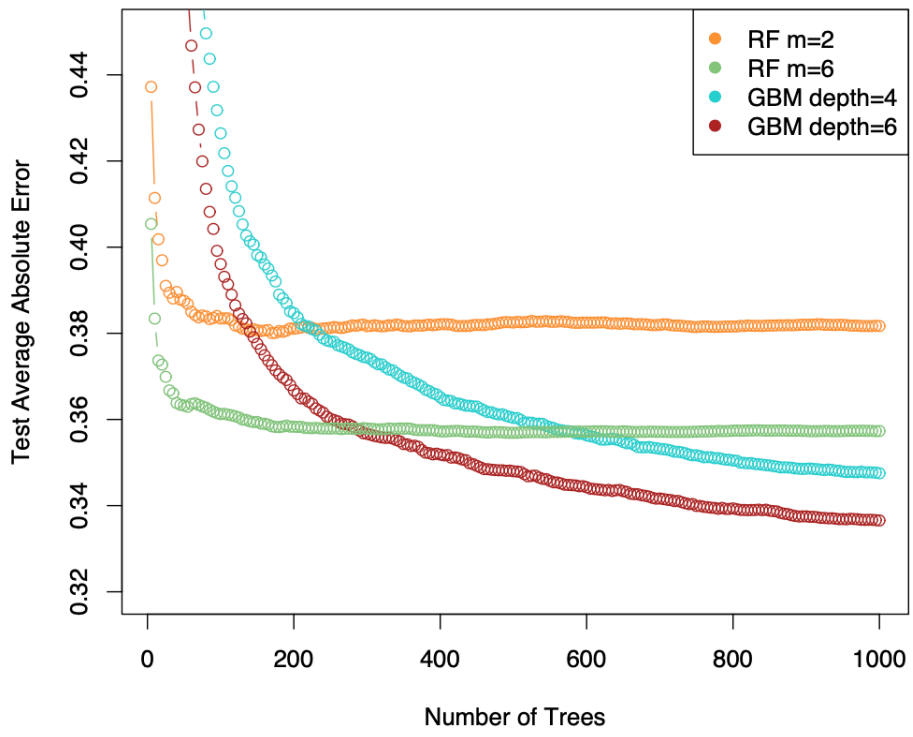


Figure: The Elements of Statistical Learning, Fig15.1, 15.3