

# 1 | 回帰とは

回帰とは解析対象となる数値と要因となる数値の関係に数学関数を当てはめる解析手法のこと。

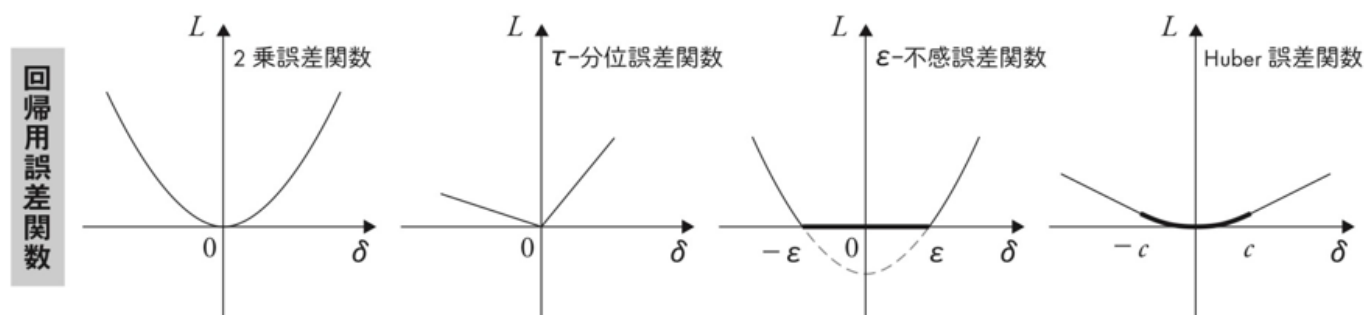
## 回帰分析の例

$$\min_{\omega, b} \sum_{i=1, \dots, N} |d_i - f(x_i)|^2$$

$$\text{where } f(X) = X\omega + b$$

## 回帰で使用する誤算関数の分類

回帰用の誤差関数は様々なものがある。



# 2 | 分類とは何か

離散的な目的変数を持つ回帰として解釈できるが、誤差関数については回帰と同様の物を使用することができない。

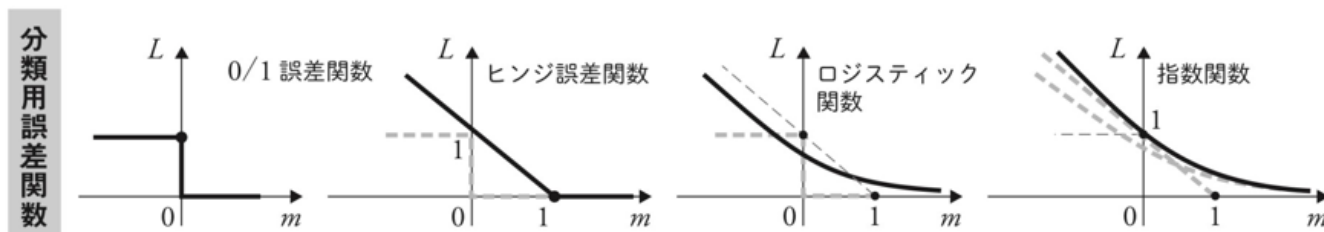
## 分類問題の例

$$\min \sum_{i=1, \dots, N} m_i$$

$$\text{where } f(x_i) = \omega x_i + b$$

$$\text{and } e^{-f(x_i)d_i}$$

ここで \$m\$ はマージンと呼ばれ、後述の "AdaBoost" の誤差関数として採用されている。



多数の組み合わせを持つ分類誤差関数がある理由は回帰と同等にバイアス・バリエーションのトレードに対応するためです。

## 3 | 統計モデルと代表的なアルゴリズム

---

### 線形モデル

#### 一般化線形モデル

$$y = g(\beta_0 x_0 + \beta_1 x_1 + \cdots + \beta_K x_K)$$

$g()$ はリンク関数

#### 一般化加法モデル

$$y = \beta_0 \phi(x_0) + \beta_1 \phi(x_1) + \cdots + \beta_K \phi(x_K)$$

## 4 | 機械学習モデルと代表的なアルゴリズム

---

機械学習は予測精度を高めるために柔軟性の高いモデルを想定していることが多い。予めデータの分布や統計モデルを仮定しない代わりに誤差関数の設計を工夫している。

### サポートベクターマシン(SVM)による分類問題

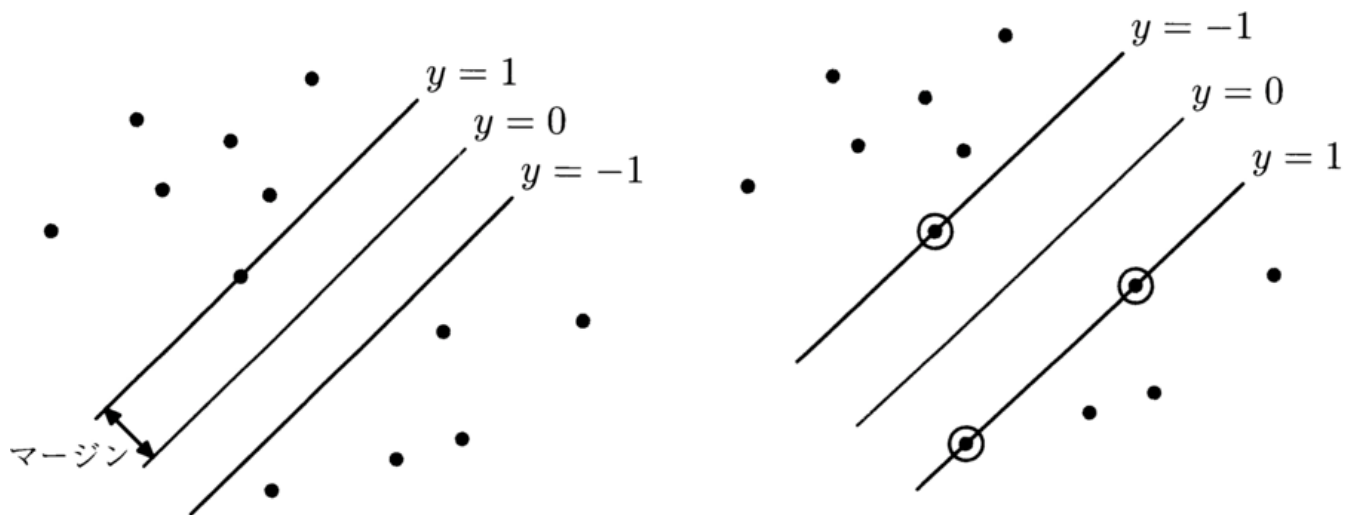
#### ハードSVM

特徴量のデータとしては、 $\{x_i\}_{i=1, \dots, N}$ が与えられ、教師データとしては $\{0, 1\}$ の2値で表される目的変数、 $\{y_i\}_{i=1, \dots, N}$ が与えられているとする。次の線形モデルを用いて2値分類問題を解くことを考える。

$$f(x) = \omega^T \phi(x) + b$$

ここで、 $\phi()$ は特徴量空間変換関数とし、 $f(x) > 0$ であるときには $y_i = +1$ 、 $f(x) \leq 0$ であるときには $y_i = -1$ として分類することを考える。

SVMでは、データの点について $f(x) = 0$ の超平面がデータ点を分類するようにパラメータの最適化を行う。



ここで、超平面  $f(x) = 0$  から各点  $x_i$  までの距離が以下の式で与えられる。

$$\frac{|f(x)|}{\|\omega\|}$$

また、ここで分類境界と各訓練データ  $i$  の間の最短距離をマージンということにする。

すべてのデータ点を正しく分類することに興味があるので、以下の不等式が成り立つ。(分類の仕方より明らか)

$$\frac{y_i f(x_i)}{\|\omega\|} > 0$$

SVMでは、マージンの最大化問題を解くので、以下の最適化問題を解く。

$$\arg\max_{\omega, b} \frac{1}{\|\omega\|} \min_{i=1, \dots, n} [y_i (\omega^T \phi(x_i) + b)]$$

ただし、これだけだとパラメータの定数倍(スケール)について識別性がないので、境界に最も近い手について

$$y_i (\omega^T \phi(x_i) + b) = 1$$

を仮定することで、識別のための制約を与えることができる。

また、ここまでの話を総合すると、上記の制約式の中で最適化問題を  $\|\omega\|^{-1}$  の最大化問題としてとらえることができるため、最適化問題自体は以下の式に集約できる。

$$\operatorname{argmin} \frac{1}{2} \|\omega\|^2$$

$$\text{where } y_i (\omega^T \phi(x_i) + b) \geq 1 \quad i = 1, \dots, N$$

## ソフトSVM

実際には、誤差  $\delta_i = |f(x_i) - y_i|$  を最小にするのが一般的である。

式で表現すると、

$$\begin{aligned} \max [d] = \min [|w|] : & \Leftrightarrow \min \left[ \frac{1}{2} |w|^2 \right] : \Leftrightarrow \min \left[ \frac{1}{2} |w|^2 + C \sum_{i=1}^N \delta_i \right] \\ \text{s.t. } (wx_i + b)y_i & \geq 1 - \delta_i \end{aligned}$$

このSVMの原理はヒンジ誤差関数を用いることで簡単に導出が可能になる。

ヒンジ誤差関数は以下の通り。

$$m_i = f(x_i) y_i$$

$$L = \min \left[ \sum_{i=1}^n \max \{0, 1 - m_i\} \right]$$

モデルの汎化性能を上げるために誤差関数に一般化正則化項 $l_2$ を導入する。

$$L = \min \left[ C \sum_{i=1}^n \max \{0, 1 - m_i\} + \frac{1}{2} \lambda |w|^2 \right] \quad (16)$$

ここで、 $\max \{0, 1 - m_i\}$  は、誤差  $\delta_i$  を使って以下のように変形できます。

$$\max \{0, 1 - m_i\} = \min \delta_i \quad \text{s.t. } \delta_i \geq 1 - m_i, \delta_i \geq 0 \quad (17)$$

最終的に、ヒンジ誤差関数の誤差最小化により、次の式となります。

$$L = \min \left[ C \sum_{i=1}^n \delta_i + \frac{1}{2} \lambda |w|^2 \right] \quad \text{s.t. } \delta_i \geq 1 - m_i, \delta_i \geq 0 \quad (18)$$

また、非線形の問題に対応するために以下のようにカーネル関数を使用する場合もある。

$$f(x) = \sum_{j=1, \dots, N} \theta_j K(x, x_j) + b$$

## SVMの計算

ヒンジ誤差関数のパラメータについての微分を行い、勾配の計算をすることによって更新の計算式を作る。

(1)もし予測した答えが訓練データのラベルと一致している場合は、下記のような更新を実行する。

$$\text{if } y_i(\omega^T x_i + b) \geq 1 \quad \omega \leftarrow \omega - \alpha \cdot (2\lambda\omega)$$

(2)もし予測した答えが訓練データのラベルと一致していない場合は、下記のような更新を実行する。

$$\text{if } y_i(\omega^T x_i + b) < 1 \quad \omega \leftarrow \omega - \alpha \cdot (y_i x_i^T - 2\lambda\omega)$$

## 実行結果について

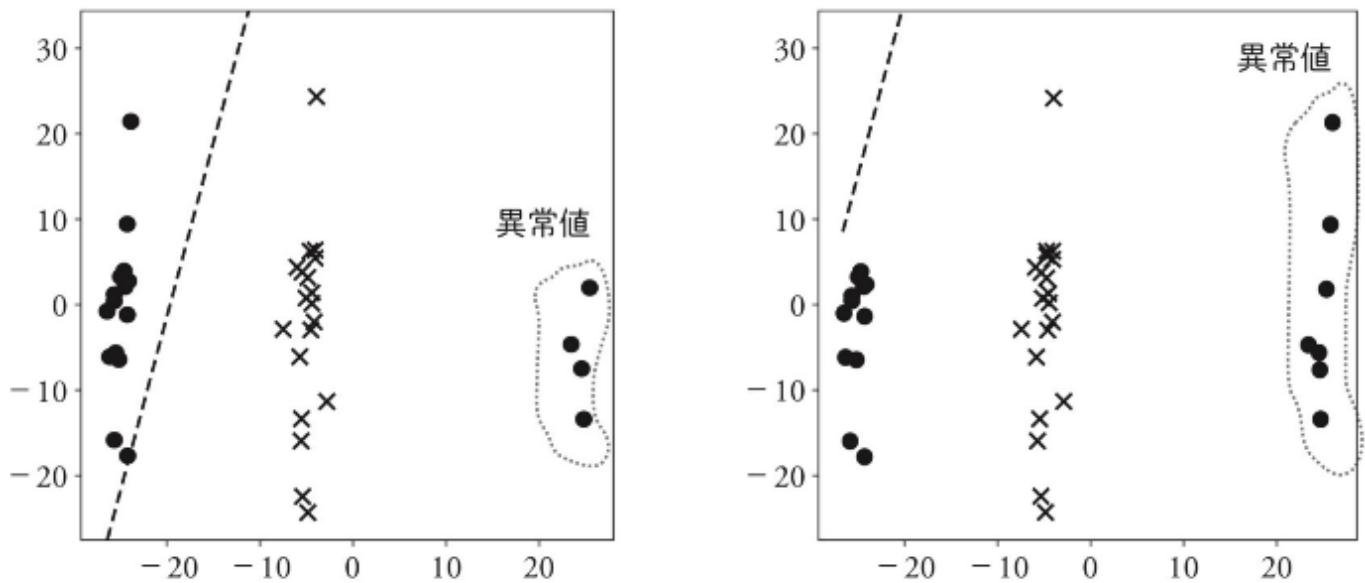


図 1.12 ヒンジ誤差関数を用いた SVM の 2 値分類問題

異常値を増やすと、境界を引くのが難しくなってしまう。

### 発展的な内容

汎化機能においてはヒンジ誤差関数よりもランブ誤差関数を用いた方が良い。

$$L = \min \sum_{i=1, \dots, N} \min 1, \max(0, 1 - m_i)$$

文献[10]で最小化アルゴリズムの解法が紹介されている。

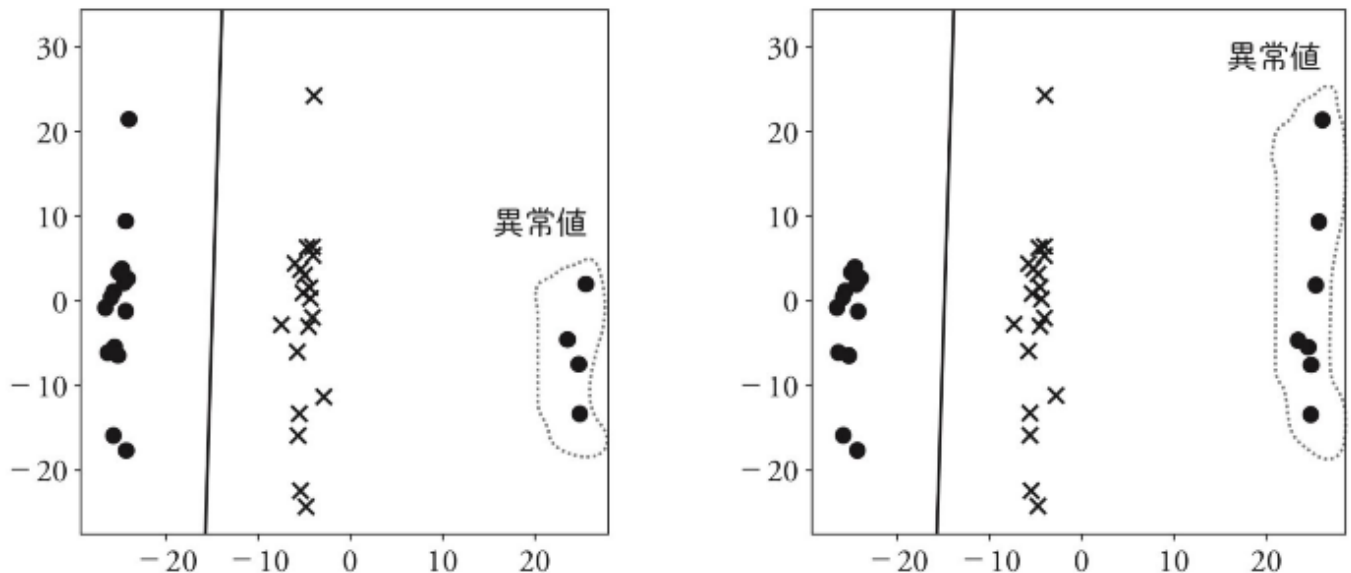


図 1.13 ランブ誤差関数を用いた SVM の 2 値分類問題

結果的に、異常値が多少増えても境界線が変化しないことがわかる。異常検知や予測の中で機械学習モデルの汎化機能に深く関連があることを覚えておく必要がある。

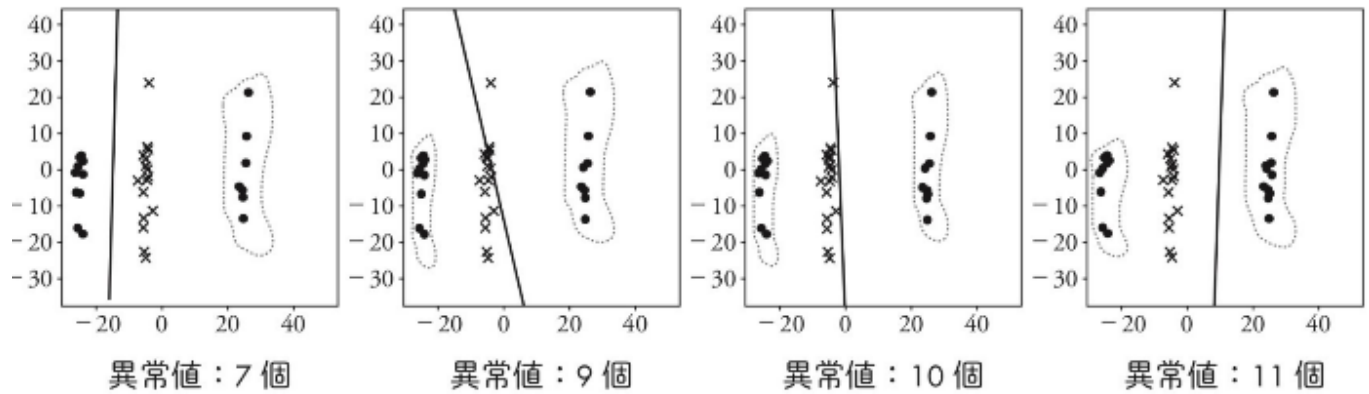


図 1.14 ランプ誤差関数を用いた SVM における汎化機能の検証

## SVMによる回帰

SVMの回帰では $\epsilon$ -不感誤差関数を利用する。

誤差関数の式は以下の通り。

$$L = \begin{cases} 0, & |\delta_i| < \epsilon \\ |\delta_i| - \epsilon = \xi, & |\delta_i| \geq \epsilon \end{cases} \quad (25)$$

誤差関数を最小化するための学習即ち以下の通り(過学習を防ぐために分類問題と同じように一般化正則化項を導入している)

$$L = \min \left[ C \sum_{i=1}^n \max \{0, |\delta| - \epsilon\} + \frac{1}{2} |w|^2 \right] \quad (26)$$

この誤差関数を見ればわかる通り、通常の解析勾配計算手法が適用できないので、Tensorflowの最適化沿る場が利用されている。

2乗誤差を用いた回帰の結果と比較してみる。これだけだと何とも言えない気がするが、誤差関数収束値が高いほうが汎化性能が高いと考えることができる。

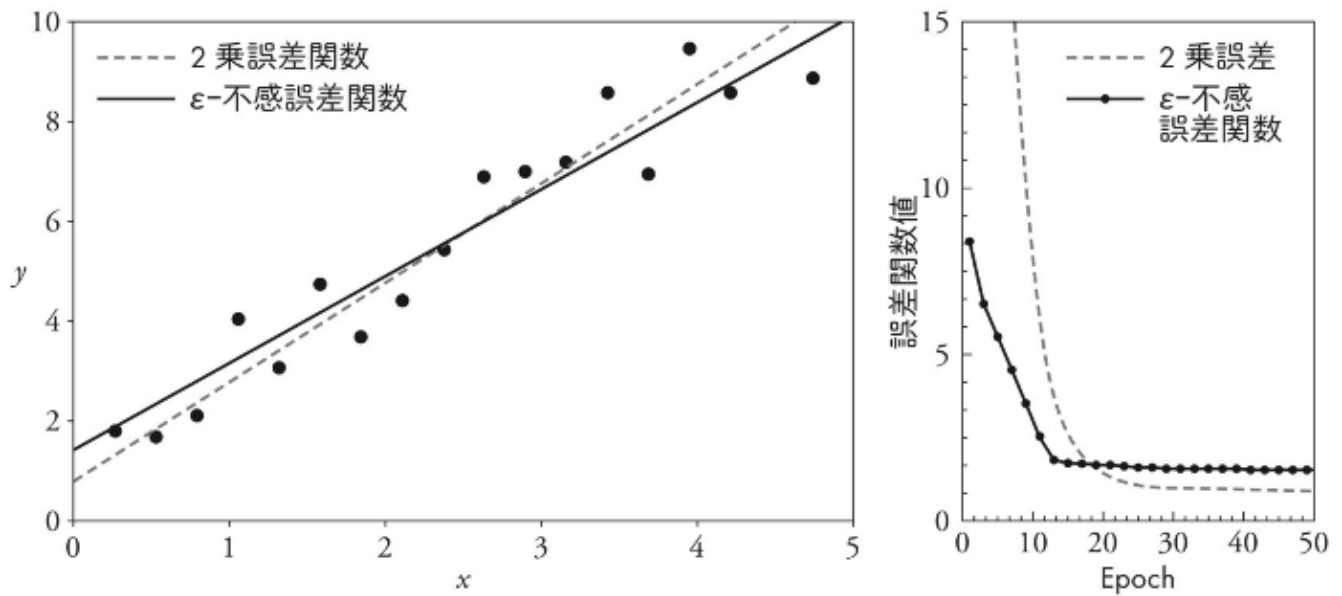
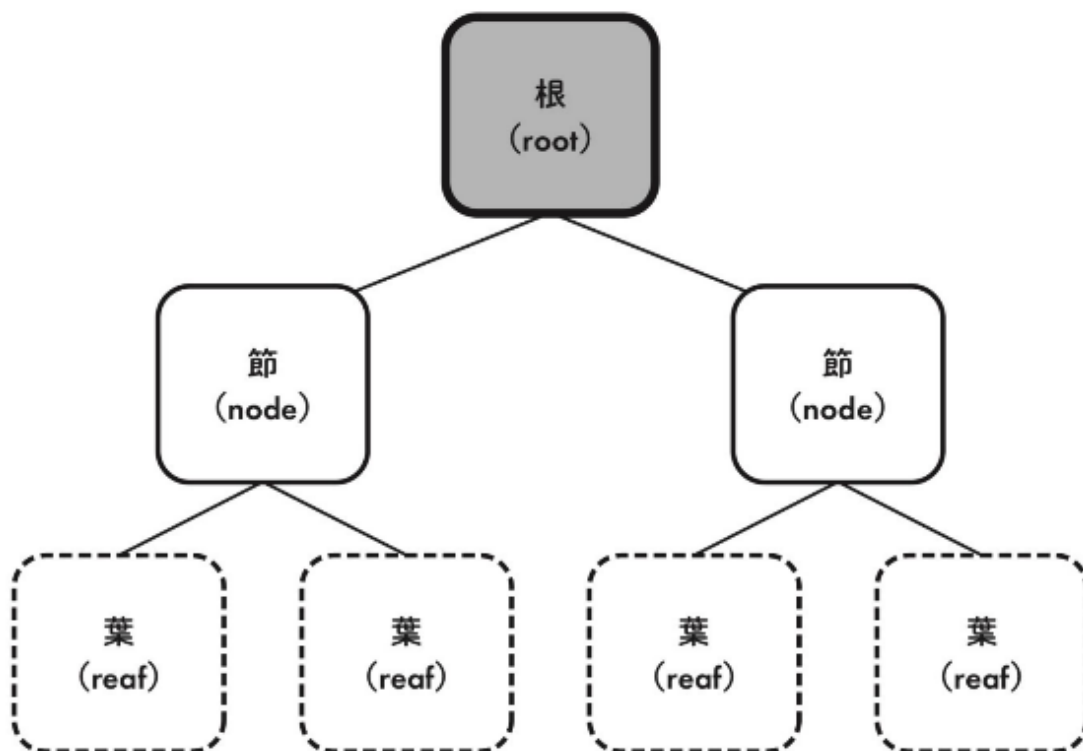


図 1.16 SVM 線形回帰と 2 乗誤差関数線形回帰の比較

## アンサンブル学習

決定木など貴行増を用いた分類や回帰について紹介する。



さかさまの木のようにみえるので「決定木」とよぶ

図 1.17 決定木の構造

## 決定木について(ジニ係数を使用した分割規則)

ジニ係数は情報の不純度を表し、値が減少するように分割を実行する。

ジニ係数の計算式

$$E(t) = 1 - \sum_{i=1}^K P_i^2(C_i|t)$$

ここで、 $K$ がクラスの数、 $P_i(C_i|t)$ はノード $t$ のクラス $C_i$ の割合。

各ノードについてジニ係数を計算した後にそのノードに分割されたサンプル数で加重平均を使用して加重ジニ係数を算出している。

## 決定木の応用

次に、ブースティングとランダムフォレストについて説明する。

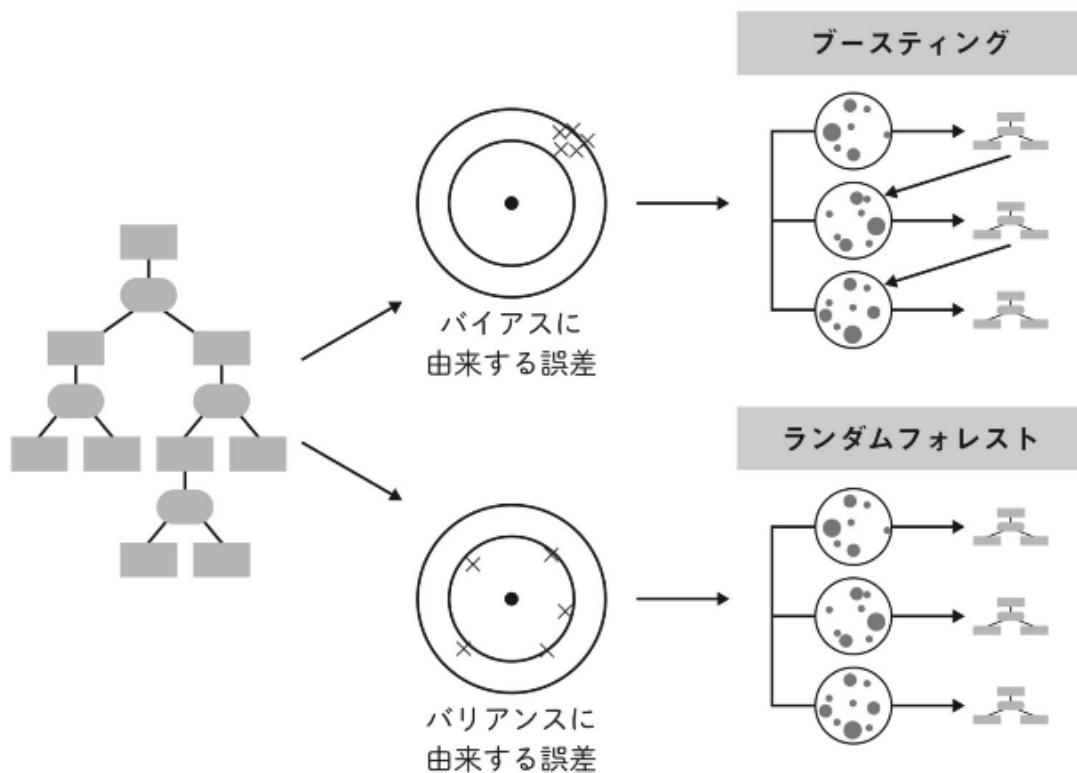


図 1.20 ブースティングとランダムフォレスト

ブースティングはバイアスに由来する誤差を低減する手法。真値との誤差を決定木によって補完する仕組み。つまり、決定木を「直列」につなぐようなイメージ。

ランダムフォレストはバリエーションに由来する誤差を低減する手法。決定木を「並列」に並べて作り、最終的な予測結果を各決定木の予測結果を最終的に統一するために多数決や平均を使用する。

各決定木の間に相関関係を設けるかどうかの違いがある。

## ランダムフォレスト

概要：それぞれの決定木が独立になるようにブートストラップと呼ばれるサンプリングを行う。具体的には、元のデータから重複ありでランダムに一部を抽出し、そのデータセットそれぞれに対して決定木を作成



する。

最終的には、それらの予測結果を平均することで回帰問題を、多数決を取ることで分類問題を解くことができる。

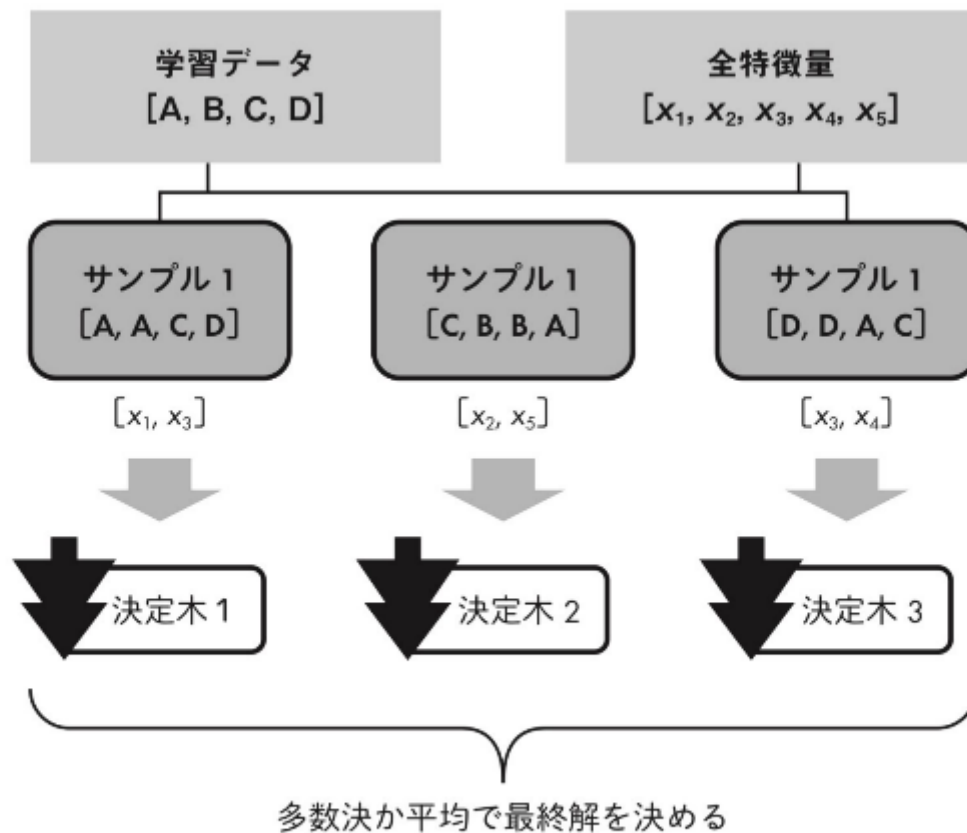


図 1.21 ランダムフォレストにおける決定木の作りかた

特徴：特徴量をランダムに選び、決定木同士の相関を弱めることによってバリエーションを下げる。(過学習を避ける) 分割にはジニ係数などの分割基準を使用する。

注意点：ノードとリーフの数を調整することによって決定木の深さを決める。木の深さと木の葉数が重要なパラメータとして最適化する必要がある。

## ブースティング(AdaBoost)

ブースティングの手法は勾配法と非勾配法の2種類ある。

### 非勾配ブースティング

概要：

1. 各データについて重み付けを行う。(初期値では $\frac{1}{N}$ を採用)
2. 識別機(決定木)を一つ作り、元データを分類してみる。
3. 誤分類されたデータがあれば、そのデータについての重みを増やす。
4. もう一度識別機(決定木)を作り、重み付けされたデータについて分類する。

これを繰り返す。

具体的な計算式、アルゴリズムは、図1.24を参考。

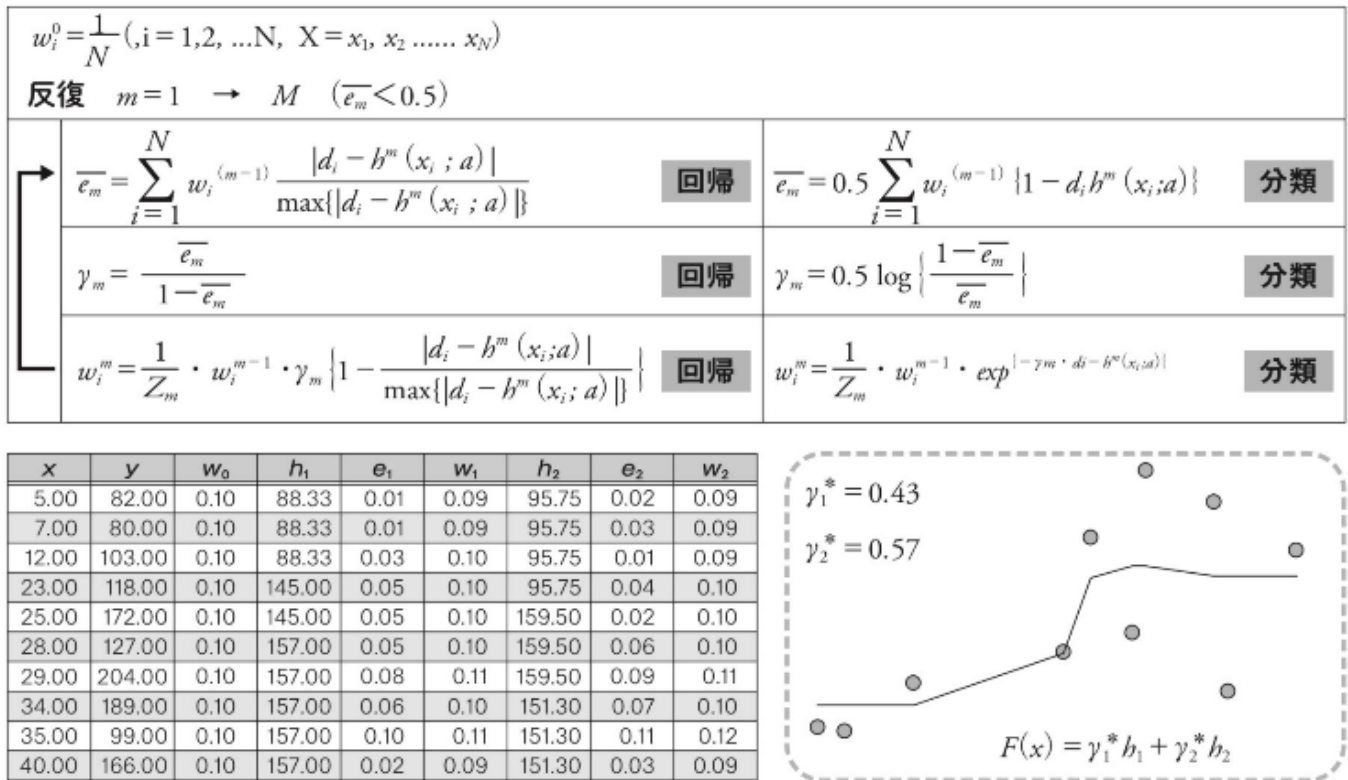


図 1.24 AdaBoost による回帰

ここで、 $\gamma_i$  は各決定木の係数。

出力の際には、分類、回帰問題に限らず、各決定木の係数と弱学習木  $h_i$  の積の総和で計算される。(分類の際にはサイン関数  $\text{sgn}(x)$  を使う。)

### 勾配ブースティング(GBDT)

非勾配の手法であるAdaBoostとの違いは、誤差関数の扱い方にある。

AdaBoost: 弱学習器毎に一つの係数で各学習を調整していた。

GBDT: 各データに対する予測値の誤差に勾配を決めることで誤差学習器を更新していく。また、こちらでは比較的浅い決定木を使用する。

また、特徴的な点として、ランダムフォレストとは逆に決定木の数を増やすと過学習を起こしやすいなどの問題がある。

基本的には以下の最小化問題を解くことを考える。

$$\arg \min_{F(X)} \sum_{i=1}^N L(y_i, F(x_i))$$

ここで  $F(X)$  は回帰関数。

アルゴリズムは以下の通り。

図中の  $F(X)$  は回帰関数です。目的関数は次のように表現されます。

$$obj = \operatorname{argmin}_{F(X)} \sum_{i=1}^N L(y_i, F(x_i)) \quad (31)$$

すると、勾配は以下のように定義されます。

$$-g_m(x_i) = - \left[ \frac{\partial L(y_i, F(x_i))}{\partial F(x_i)} \right]_{F(X)=F_{m-1}(X)} \quad (32)$$

弱学習器  $h(x_i, \alpha)$  は、以下の条件を満足するように構成されます。これはランダムフォレストを実行する際に使用した分割基準（分散最小あるいは情報エントロピー最小原則）と一致しています。ここでは分散最小則で表現しましょう。

$$\alpha_m = \operatorname{argmin}_{\beta, a} \sum_{i=1}^N [-g_m(x_i) - \beta h^m(x_i; a)]^2 \quad (33)$$

続いて、更新する歩幅  $\gamma_m$  を決めます。

$$\gamma_m = \operatorname{argmin}_{\gamma, a} \sum_{i=1}^N L(y_i, F_{m-1}(X) + \gamma h^m(x_i; \alpha_m)) \quad (34)$$

$$F_m(X) = F_{m-1}(X) + \gamma h^m(x_i; \alpha_m) \quad (35)$$

これらを通じて、以下のように更新されていく。

$$\begin{aligned} r_1 &= d - F_0(X) \\ r_1 &\rightarrow h^1(x_i; a) \\ F_1(X) &= F_0(X) + h^1(x_i; a) \\ r_2 &= d - F_1(X) \\ r_2 &\rightarrow h^2(x_i; a) \\ F_2(X) &= F_1(X) + h^2(x_i; a) \\ r_3 &= d - F_2(X) \\ &\vdots \end{aligned}$$

最小2乗誤差関数によるGBDTのアルゴリズムは以下の通り。

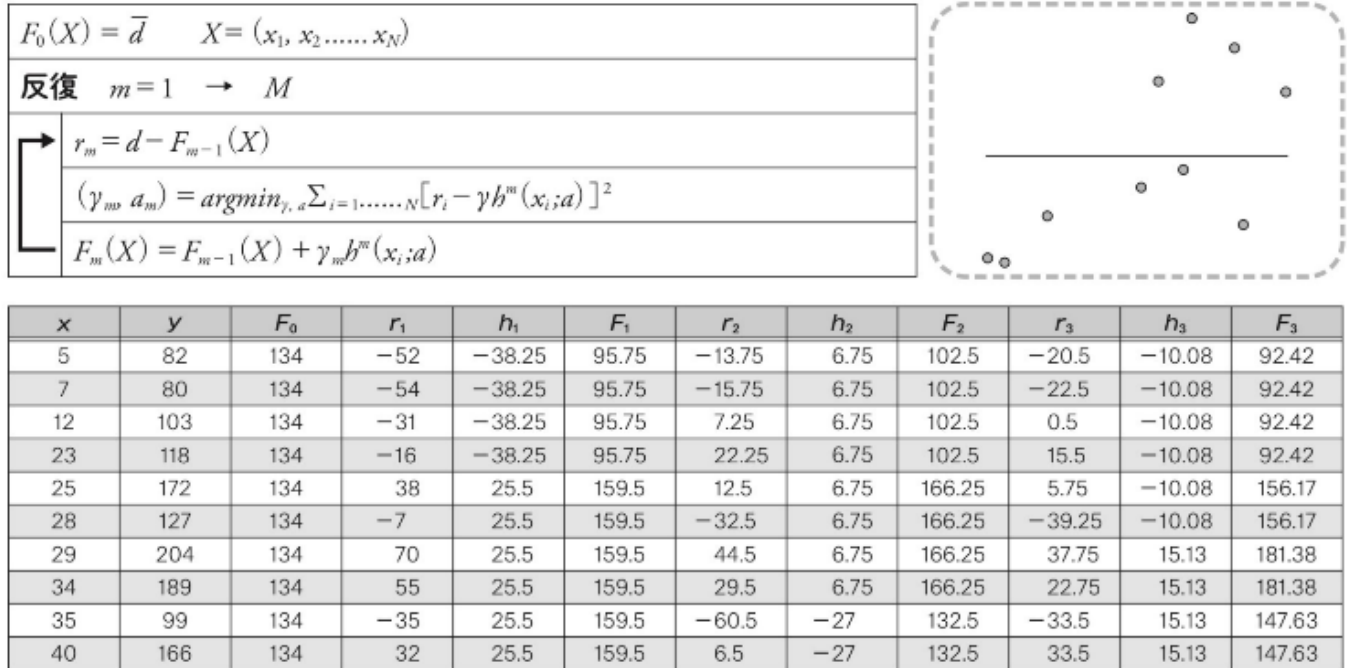


図 1.25 最小 2 乗誤差関数による GBDT のアルゴリズム

決定木のイメージ

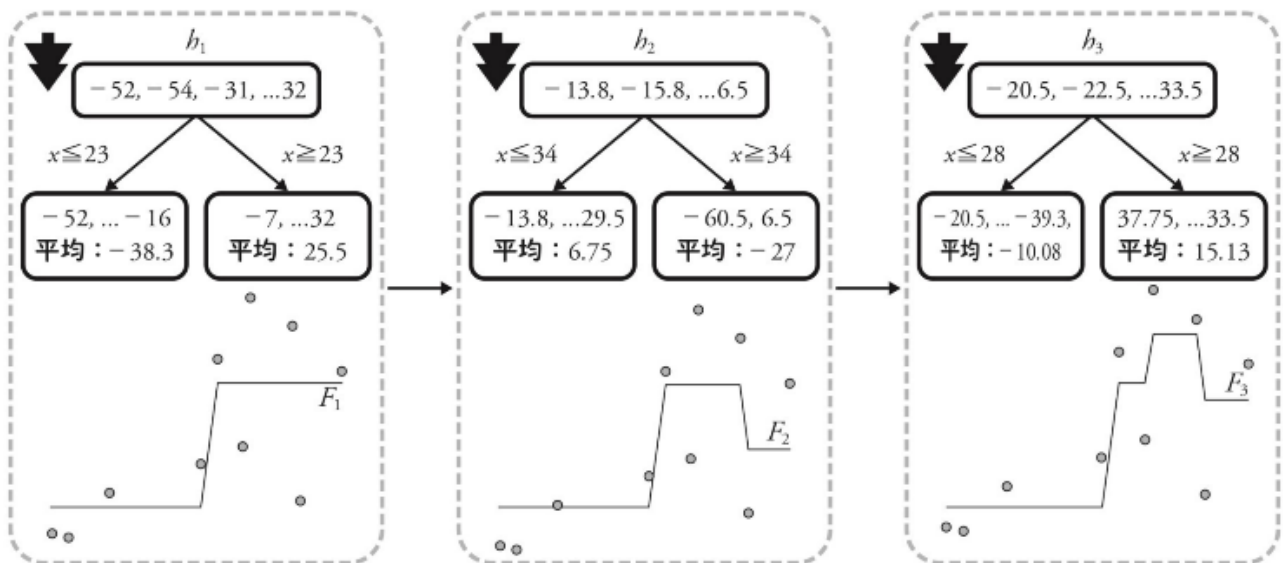


図 1.26 決定木の更新と回帰関数（勾配ブースティング）

## XGBoosting

GBDTで高くなりがちなバリエーションは汎化機能の低下及びそれに伴う予測精度の低下を引き起こすことがある。それを回避するために開発された手法がXGブーストである。

### XGBoostの誤差関数

XGBoostでは以下の誤差関数を最小化する。

$$obj^m = \sum_{i=1}^N \left[ L(y_i, F^{m-1}(x_i)) + g_i f_m(x_i) + \frac{1}{2} h_i f_m^2(x_i) \right] + \Omega(f_m) + constant \quad (38)$$

ここで  $g_i$  と  $h_i$  は以下のように定義されています。

$$g(x_i) = \left[ \frac{\partial L(y_i, F^{m-1}(x_i))}{\partial F^{m-1}(x_i)} \right], \quad h(x_i) = \left[ \frac{\partial^2 L(y_i, F^{m-1}(x_i))}{\partial \{F^{m-1}(x_i)\}^2} \right] \quad (39)$$

ちなみに  $\Omega(f_m)$  は期の複雑さを表す正則項となっており、汎化機能や予測精度の向上が見込める。

$$\Omega(f_m) = \gamma T + \frac{1}{2} \lambda \sum_{i=1}^T \{f_m^i\}^2 \quad (40)$$

### XGBoostの特徴

1.  $f(x_i)$  の解析解が導出されている。

$$f_m = -\frac{G_j^2}{H_j + \lambda}$$

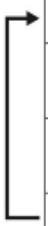
$$\text{where } G = \sum_{i=1}^N g_i \sum_{i=1}^N h_i$$

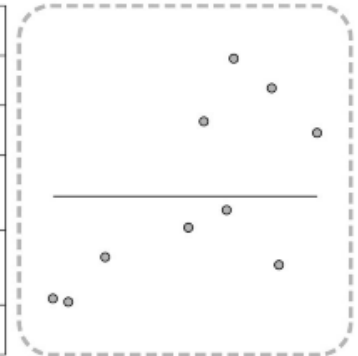
2.  $f(x_i)$  を構成する決定木の分割基準が提供される。

$$\min_T -\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T$$

$T$  は決定木のノードの数分割の基準としては目的関数が最小になるように分割する。

具体的なアルゴリズム

$F_0(X) = \bar{d} \quad X = (x_1, x_2, \dots, x_N)$	
反復 $m = 1 \rightarrow M$	
	$g_i = 2[d - F_{m-1}(X)], h_i = 2 \rightarrow G = \sum_{i=1}^N g_i, H = \sum_{i=1}^N h_i$
	$\operatorname{argmin}_T \text{Obj} = -\frac{1}{2} \sum_{i=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T \rightarrow \text{Splittree}(\text{決定木を決める})$
	$f_m = -\frac{G_j}{H_j + \lambda} \rightarrow \text{各決定木で計算を行う}$
	$F_m = F_{m-1} + \alpha f_m$



$x$	$y$	$F_0$	$g$	$h$	$f_1$	$F_1$	$g$	$h$	$f_2$	$F_2$	$g$	$h$	$f_2$	$F_3$
5.00	82.00	134.00	-104.00	2.00	-34.77	99.23	-34.46	2.00	5.49	104.72	-45.44	2.00	-9.95	94.77
7.00	80.00	134.00	-108.00	2.00	-34.77	99.23	-38.46	2.00	5.49	104.72	-49.44	2.00	-9.95	94.77
12.00	103.00	134.00	-62.00	2.00	-34.77	99.23	7.54	2.00	5.49	104.72	-3.44	2.00	-9.95	94.77
23.00	118.00	134.00	-32.00	2.00	-34.77	99.23	37.54	2.00	5.49	104.72	26.56	2.00	-9.95	94.77
25.00	172.00	134.00	76.00	2.00	23.90	157.90	28.20	2.00	5.49	163.39	17.22	2.00	-9.95	153.44
28.00	127.00	134.00	-14.00	2.00	23.90	157.90	-61.80	2.00	5.49	163.39	-72.78	2.00	-9.95	153.44
29.00	204.00	134.00	140.00	2.00	23.90	157.90	92.20	2.00	5.49	163.39	81.22	2.00	13.07	176.46
34.00	189.00	134.00	110.00	2.00	23.90	157.90	62.20	2.00	5.49	163.39	51.22	2.00	13.07	176.46
35.00	99.00	134.00	-70.00	2.00	23.90	157.90	-117.80	2.00	-21.04	136.86	-75.72	2.00	13.07	149.93
40.00	166.00	134.00	64.00	2.00	23.90	157.90	16.20	2.00	-21.04	136.86	58.28	2.00	13.07	149.93

図 1.27 XG ブースティングの計算手順