第14回 決定木分析

Code

1. 決定木の概要

決定木分析（decision tree analysis）は、ある変数を他の変数から予測するためのif-thenルールを求める方法です。予測対象の変数を目的変数、予測に用いる変数を説明変数、得られるif-thenルールを**決定木**といいます。また目的変数には量的変数・質的変数のどちらもとることができ、

- 目的変数が量的変数の場合 : 決定木回帰
- 目的変数が質的変数の場合 : 決定木分類

と異なる名前でよばれます。今回は、決定木分類のデモを行い、決定木を求める仕組みを解説します。

2. 決定木分類のデモ

2.1 デモデータ

`data` ディレクトリの `split_demo.csv` をデモデータに用います。このデータは、標本サイズが100で3変数のデータです。今回は、 `x` , `y` を説明変数、 `label` を目的変数とした決定木分類を行ってみます。

Hide

```
dat <- read.csv("./data/split_demo.csv", fileEncoding = "utf-8")
dat$label <- as.factor(dat$label)
head(x = dat, n = 5)
```

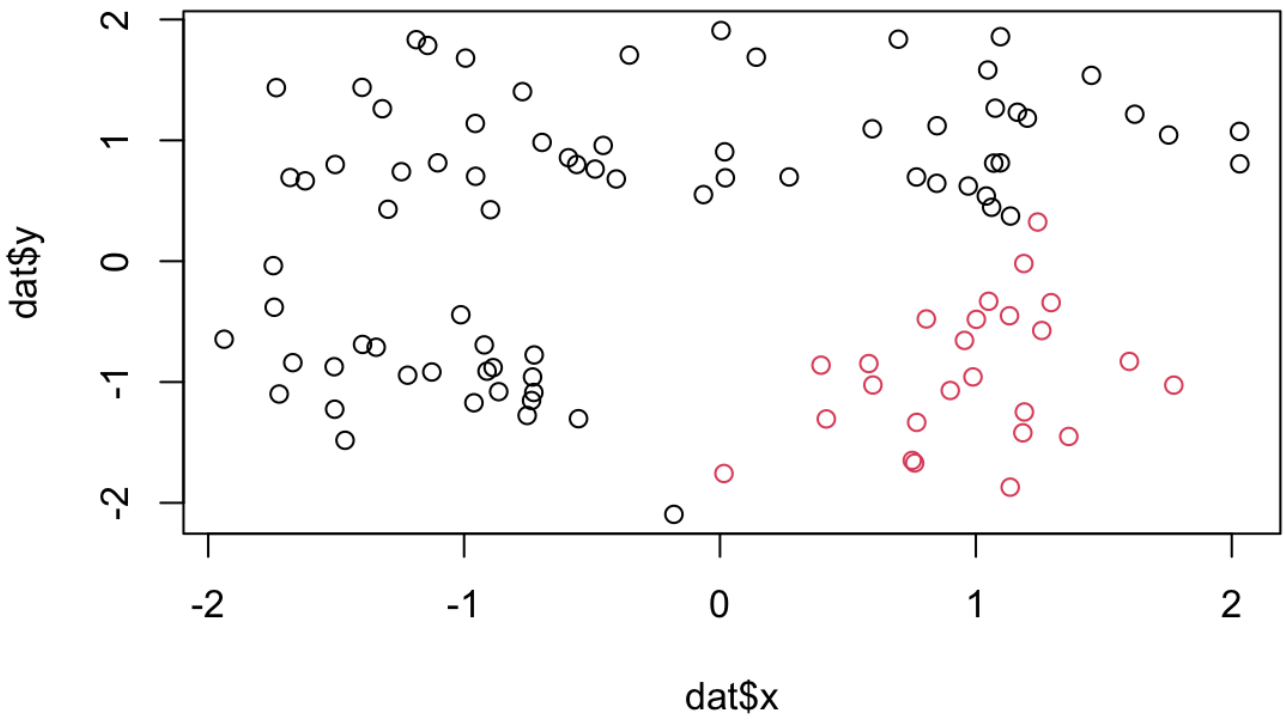
	x <dbl>	y <dbl>	label <fctr>
1	1.6208878	1.2160687	1
2	0.1418463	1.6881527	1
3	1.1352569	0.3735209	1
4	1.4515459	1.5378424	1
5	1.0407771	0.5394458	1

5 rows

`label` 別に色分けした散布図をかいて、標本点の分布を確認しておきましょう。

Hide

```
plot(dat$x, dat$y, col = dat$label)
```



問題：この散布図から、決定木分類によって得られそうなif-thenルールを予め想定してみましょう。

2.2 決定木分類の計算

決定木分類は `rpart` パッケージの `rpart` 関数で計算することができます。指定する引数は以下のとおりです。

- `formula` ：目的変数と説明変数を指定する。
- `data` ：決定木を作るために用いるデータ。
- `method` ：回帰か分類か。回帰の場合は `anova` 、分類の場合は `class` を指定する。
- `control` ：得られる決定木の複雑さをコントロールするための引数（**ハイパーパラメータ**）
  - `maxdepth` ：ここに渡した数より深い決定木は作らない。
  - `minsplit` ：ここに渡した数より所属する標本点の個数が少ないノードでは、これ以上分割を試みない。
  - `minbucket` ：終端ノードに所属する標本点の個数は、すくなくともここに渡した数より大きくなるようにする。

`control` の説明にはわからない単語もあると思いますが、3.2で詳しく説明します。

Hide

```
library(rpart)
result <- rpart(formula = label ~ x + y,
                data = dat,
                method = "class",
                control = rpart.control(maxdepth = 2,
                                        minsplit = 20,
                                        minbucket = 5))
result
```

```
n= 100

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 100 25 1 (0.75000000 0.25000000)
  2) x< 0.3325615 55  1 1 (0.98181818 0.01818182) *
  3) x>=0.3325615 45 21 2 (0.46666667 0.53333333)
    6) y>=0.3488362 21  0 1 (1.00000000 0.00000000) *
    7) y< 0.3488362 24  0 2 (0.00000000 1.00000000) *
```

result には得られた決定木が出力されていますが、分かりづらいと思います。 partykit パッケージの as.party 関数でわかりやすい形に書き換えておきましょう。

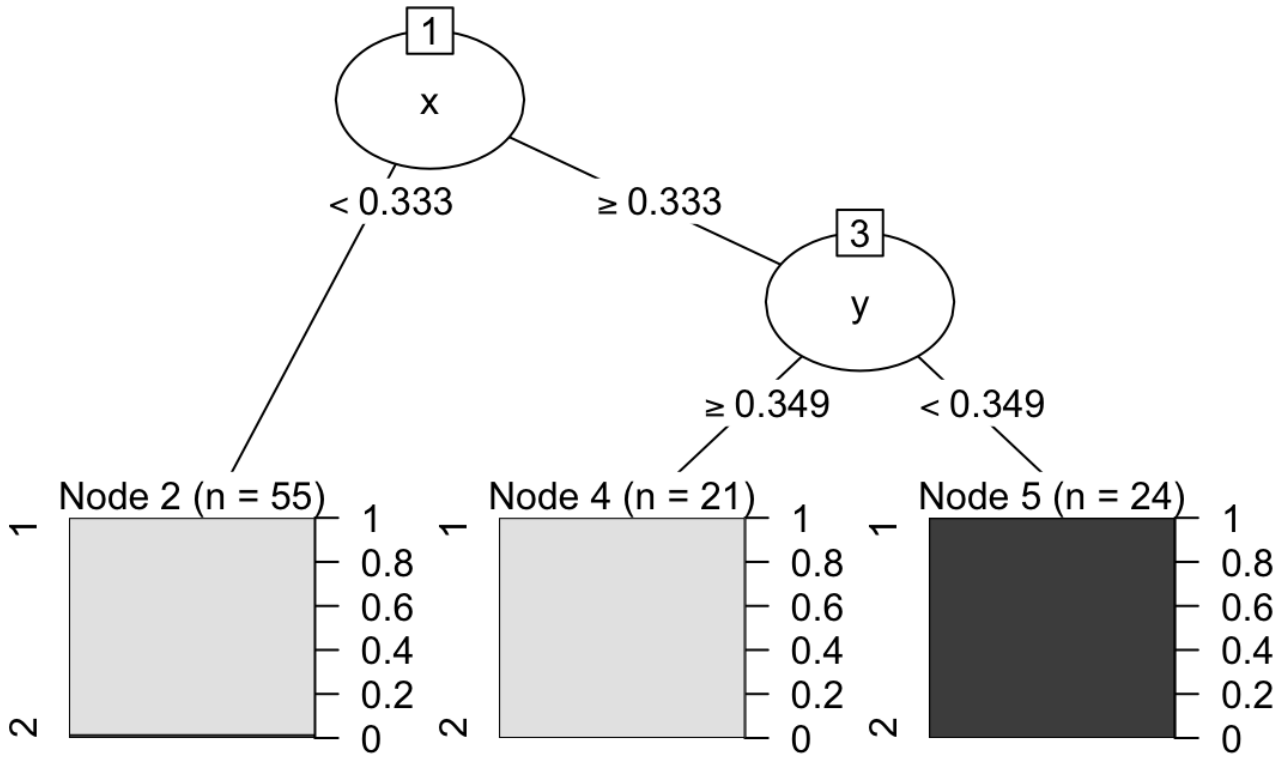
Hide

```
library(partykit)
```

```
要求されたパッケージ grid をロード中です
要求されたパッケージ libcoin をロード中です
要求されたパッケージ mvtnorm をロード中です
```

Hide

```
plot(as.party(result))
```



問題 : 得られた決定境界を散布図上に図示してください。

2.3 変数重要度

各説明変数が目的変数の予測にどれだけ寄与したのかを数値を用いて表現する方法に**変数重要度**（variable importance）があります。詳しくは3.4節で説明しますが、ひとまず値を参照する方法を紹介します。

Hide

```
result$variable.importance
```

```
      y       x 
23.56768 16.33636 
```

3. 決定木分類の仕組み

3.1 Giniエントロピー

決定木分類の仕組みを理解するうえで大切になる**Giniエントロピー**を解説します。Giniエントロピーは、データに含まれるラベルの個数の均等さをあらわす数値のことで、すべてのラベルが同じ数含まれているとき最も大きな値をとるように定義されています。

Giniエントロピーの定義を述べます。データに L 種類のラベルがあり、それぞれ $p_1, \cdots, p_K$ の割合で含まれているとします。このとき、Giniエントロピーは

$$I = 1 - \sum_{l=1}^{K} p_l^2$$

と定義されます。

問題 : 2種類のラベル $l = 1, 2$ を考えます。データに各ラベルが含まれる割合 p_1, p_2 が以下のようになっているとき、Giniエントロピーがいくらになっているかを計算してください。

1. $p_1 = 0.5, p_2 = 0.5$
2. $p_1 = 0.8, p_2 = 0.2$
3. $p_1 = 1.0, p_2 = 0.0$

解答 : 1. $I = 1 - (0.5^2 + 0.5^2) = 0.5$ 2. $I = 1 - (0.8^2 + 0.2^2) = 0.32$ 3. $I = 1 - (1^2 + 0^2) = 0$

3.2 決定木を求める仕組み

決定木分類で決定木を求める仕組みを説明します。決定木を求めるときに大切なのは、Giniエントロピーが最も変化するようなルールを見つけるというアイディアです。

データ D を説明変数 x が t 以上の値をもつデータ D_1 と t 未満の値をもつデータ D_2 に分割したとします。このとき、Giniエントロピーの変化量を次のように計算することができます。

$$\Delta I(x, t) = I(D) - \left(\frac{n(D_1)}{n(D)} I(D_1) + \frac{n(D_2)}{n(D)} I(D_2) \right)$$

ここで、$n(D)$ はデータ D に含まれる標本サイズです。決定木分類では、この変化量が最も大きくなるような説明変数 x と閾値 t のペアを逐次的に探していきます。

なお、得られる決定木が複雑になりすぎないように、以下のようなハイパーパラメータを予め設定します。

- `maxdepth` ：ここに渡した数より深い決定木は作らない。
- `minsplit` ：ここに渡した数より所属する標本点の個数が少ないノードでは、これ以上分割を試みない。
- `minbucket` ：終端ノードに所属する標本点の個数は、すくなくともここに渡した数より大きくなるようにする。

ここでノードとは、決定木のうち説明変数と閾値のペア (x, t) が書かれている部分のことです。また、終端ノードとはすべての条件をみたしたデータのうちの何割が各ラベルに所属しているかを記載する部分のことです。

例えば `maxdepth=2` , `minsplit=20` , `minbucket=5` のとき、以下のようなことに注意して説明変数と閾値のペアを探します。

- 深さが2以上になるような決定木を作らない。
- 標本サイズが20を下回るようなデータでは説明変数と閾値のペアを探さない。
- 分割するときは、分割後のデータの標本サイズがいずれも 5 以上になるようにする。

3.3 決定木についての重要な注意点

決定木分析によって得られる決定木が、必ずしも人間の直感と一致するとは限らない点には注意を払ってください。決定木分析では、**得られた決定木を信頼しすぎず、どれだけ得られた決定木に納得感があるかを分析者が反省する**ことが大切になってきます。

単純かつ有名な例としてXOR問題とよばれている分類問題があります。 `XOR_demo.csv` に対して、事前にif-thenルールを想定し、決定木分類によって得られる決定木と比較をおこなってみてください。

Hide

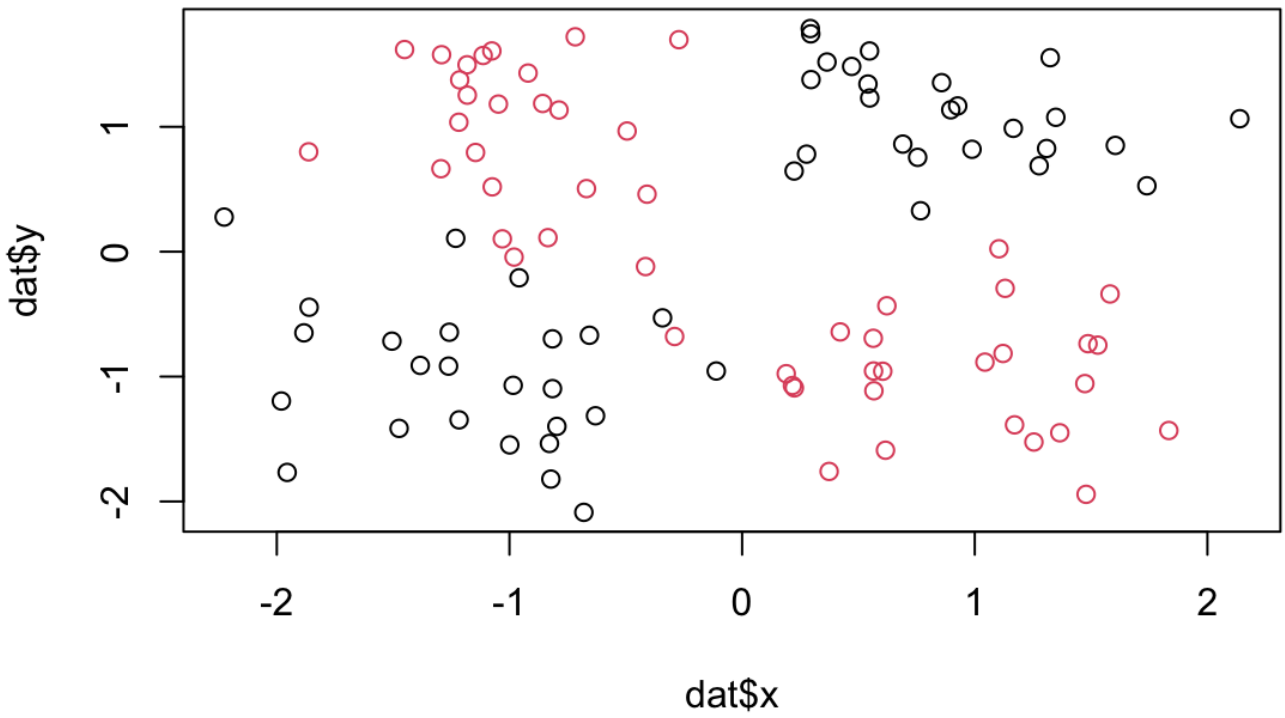
```
dat <- read.csv("./data/XOR_demo.csv", fileEncoding = "utf-8")
dat$label <- as.factor(dat$label)
head(x = dat, n = 5)
```

	x <dbl>	y <dbl>	label <fctr>
1	0.5477087	1.6073376	1
2	0.2236389	0.6460584	1
3	0.5484982	1.2307716	1
4	0.9882086	0.8199968	1
5	1.1657290	0.9876346	1

5 rows

Hide

```
# この散布図から、事前に適切なif-thenルールを想定してください。
plot(dat$x, dat$y, col = dat$label)
```



Hide

```
result <- rpart(formula = label ~ x + y,
                data = dat,
                method = "class",
                control = rpart.control(maxdepth = 2,
                                        minsplit = 20,
                                        minbucket = 5))
result
```

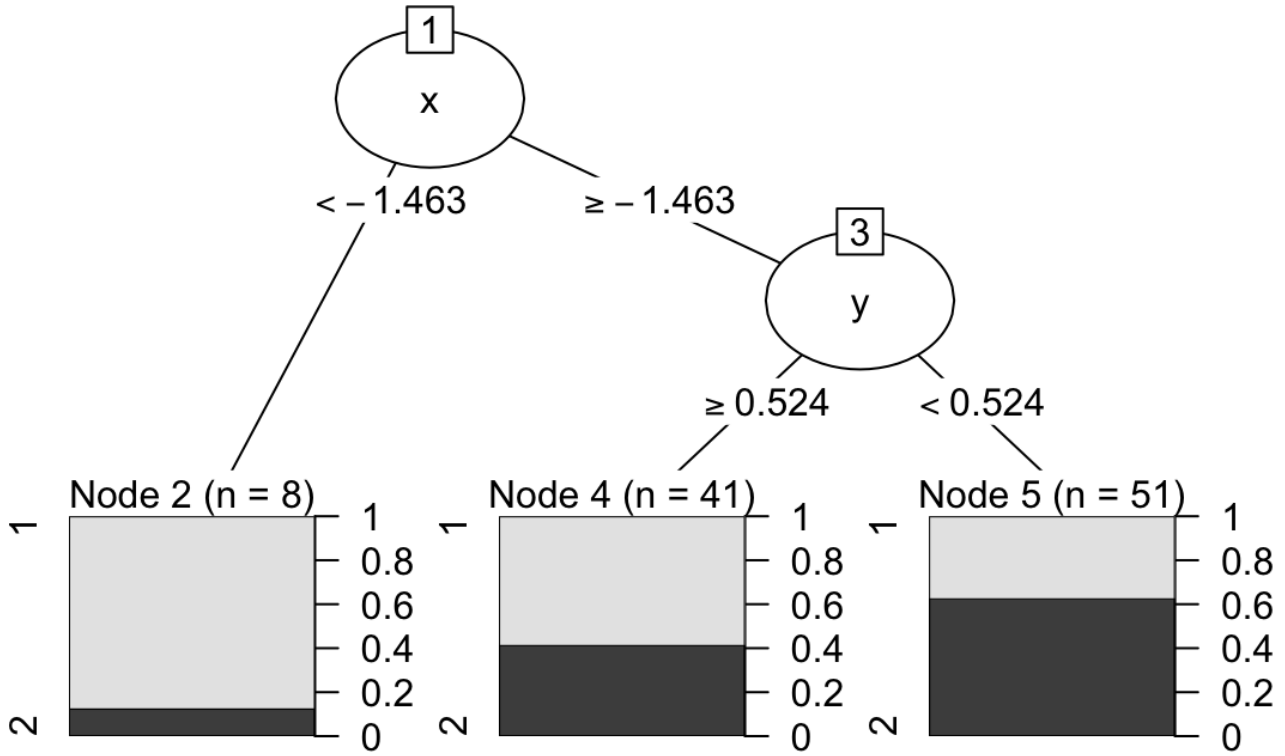
```
n= 100

node), split, n, loss, yval, (yprob)
      * denotes terminal node

1) root 100 50 1 (0.5000000 0.5000000)
  2) x< -1.462537 8  1 1 (0.8750000 0.1250000) *
  3) x>=-1.462537 92 43 2 (0.4673913 0.5326087)
    6) y>=0.5235568 41 17 1 (0.5853659 0.4146341) *
    7) y< 0.5235568 51 19 2 (0.3725490 0.6274510) *
```

Hide

```
# 得られた決定木と想定したif-thenルールを比較してみましょう。
plot(as.party(result))
```



3.4 変数重要度

あるノードにおいて、説明変数 x を用いてGiniエントロピーを改善した値は $I(x,t)$ をそのノードの標本サイズとの積、つまり $n(D)I(x,t)$ で表すことができます。これを**total improvement**といいます。説明変数 x の変数重要度は、x が使われたノードにおけるtotal improvementの合計のことです。つまり、変数重要度が大きい説明変数はそれだけ目的変数の予測に寄与していると解釈することができます。

Remark : 実は `rpart` ではもう少し修正を加えた変数重要度の計算を行っていますが、今回は最もシンプルな変数重要度の定義を紹介しました。