CS第2　テーマ2　演習ガイド

これからの予定

1. 課題2のレポートについて（復習）
2. 法則発見のためのツールの説明 → 宿題3

1. レポート課題2（復習）

データマイニングとは
法則を発見することない

毒キノコの法則を発見せよ

キノコの特徴（属性 attribute）から，与えられたキノコが毒キノコか否かを判定するルール（判別規則 binary decision rule）を見つけよう

1. レポート課題2（復習）

データマイニングとは
法則を発見することない

少し簡単にしました

毒・無毒

+1: poisonous
−1: edible

毒キノコの法則を発見せよ

19 個の属性

0, 1, 2 の
三種類

4000 種類のキノコ

−1	0	0	1	2	2	2	2	1	0	2	2	2	0	2	1	1	1	0	1
+1	0	2	0	1	1	2	1	1	0	2	2	0	0	2	1	2	0	1	0
+1	0	0	2	2	1	2	1	1	0	1	1	1	0	0	1	0	0	2	2
−1	0	0	1	1	2	2	1	1	0	2	2	2	0	2	1	2	0	1	2
+1	2	2	0	2	0	2	1	2	2	2	2	2	0	2	1	1	0	1	2
+1	1	0	0	2	1	2	1	1	0	1	1	1	0	0	1	0	0	2	2
−1	1	0	0	1	0	2	2	2	1	2	2	2	0	2	1	2	0	1	2
−1	2	2	1	2	0	2	1	2	2	2	1	2	1	1	1	1	0	1	0
+1	2	2	1	2	1	2	1	2	2	2	2	1	0	2	1	1	0	1	0
−1	0	1	0	1	0	2	1	1	0	1	2	2	0	2	1	2	0	0	1
+1	1	2	2	1	1	2	1	1	0	2	2	0	0	2	1	2	0	0	1

UC Irvine ML Repository, 1982

1. レポート課題2（復習）

データマイニングとは
法則を発見することない

何を作る？
what?

毒キノコの法則を発見せよ

毒・無毒
+1: poisonous
−1: edible

19 個の属性

0, 1, 2 の三種類

4000 種類のキノコ

−1	0	0	1	2	2	2	2	1	0	2	2	2	0	2	1	1	1	0	1
+1	0	2	0	1	1	2	1	1	0	2	2	0	0	2	1	2	0	1	0
+1	0	0	2	2	1	2	1	1	0	1	1	1	0	0	1	0	0	2	2
−1	0	0	1	1	2	2	1	1	0	2	2	2	0	2	1	2	0	1	2
+1	2	2	0	2	0	2	1	2	2	2	2	2	0	2	1	1	0	1	2
+1	1	0	0	2	1	2	1	1	0	1	1	1	0	0	1	0	0	2	2
−1	1	0	0	1	0	2	2	2	1	2	2	2	0	2	1	2	0	1	2
−1	2	2	1	2	0	2	1	2	2	2	1	2	1	1	1	1	0	1	0
+1	2	2	1	2	1	2	1	2	2	2	2	1	0	2	1	1	0	1	0
−1	0	1	0	1	0	2	1	1	0	1	2	2	0	2	1	2	0	0	1
+1	1	2	2	1	1	2	1	1	0	2	2	0	0	2	1	2	0	0	1

UC Irvine ML Repository, 1982

法則って何？

属性値を使って毒性の
有無±1を判定する条件式

属性 5 == 2

属性5 の値が 2　⇒ 毒（つまり，+1）
そうでない　⇒ 無毒（つまり，-1）

a_0 a_4 ↓ 一般の属性 a_j

毒性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
−1	0	0	1	2	2	2	2	1	0	2	2	2	0	2	1	1	1	0	1
+1	0	2	0	1	1	2	1	1	0	2	2	0	0	2	1	2	0	1	0
+1	0	0	2	2	1	2	1	1	0	1	1	1	0	0	1	0	0	2	2
−1	0	0	1	1	2	2	1	1	0	2	2	2	0	2	1	2	0	1	2

注）簡単のため毒性を 0 番目の属性 a_0 とする.

一般に条件式とは

a_j == k もしくは a_j != k を
AND, OR, NOT でつなげた論理式

⇒ 宿題

結局欲しいのは
このように
使える条件式 ⟶

if 条件式（が成立）
　　a_0 は +1 と判定
else
　　a_0 は -1 と判定

1. レポート課題2（復習）

頻度を調べる

$a_4 == 2$ && $a_0 == +1$
$a_4 == 2$ && $a_0 == -1$

前にもやったぞ！

毒性	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	1
−1	0	0	1	2	2	2	2	1	0	2	2	2	0	2	1	1	1	(
+1	0	2	0	1	1	2	1	1	0	2	2	0	0	2	1	2	0	1	0
+1	0	0	2	2	1	2										0	0	2	2
−1	0	0	1	1	2	2										2	0	1	2
+1	2	2	0	2	0	2										1	0	1	2
+1	1	0	0	2	1	2										0	0	2	2
−1	1	0	0	1	0	2										2	0	1	2
−1	2	2	1	2	0	2										1	0	1	0
+1	2	2	1	2	1	2	1	2	2	2	2	1	0	2	1	1	0	1	0
−1	0	1	0	1	0	2	1	1	0	1	2	2	0	2	1	2	0	0	1
+1	1	2	2	1	1	2	1	1	0	2	2	0	0	2	1	2	0	0	1
−1	0	2	1	2	2	1	1	1	1	1	2	2	0	0	1	2	2	0	0
−1	0	1	1	1	2	2	1	1	0	2	2	2	0	0	1	2	0	2	2
+1	0	1	0	2	1	2	1	1	0	1	1	1	0	1	1	0	0	2	0
+1	0	0	0	2	1	2	1	1	0	1	1	1	0	1	1	0	0	2	1
+1	0	1	1	2	0	2	1	2	2	2	1	1	1	1	1	1	0	1	0

$a_4 = 2$ の時，ほとんどが+1（毒性有り）ならば，それをルールに取り入れるのがよいのでは！？

⋮

1. レポート課題2（復習）

やるべきこと

1. 毒キノコを判定する条件式を求める

   - ランダムにサンプルした**サンプルデータ**を解析
     ⇒ 条件式を導き出す
   - その条件式の精度を**テストデータ**で確認する

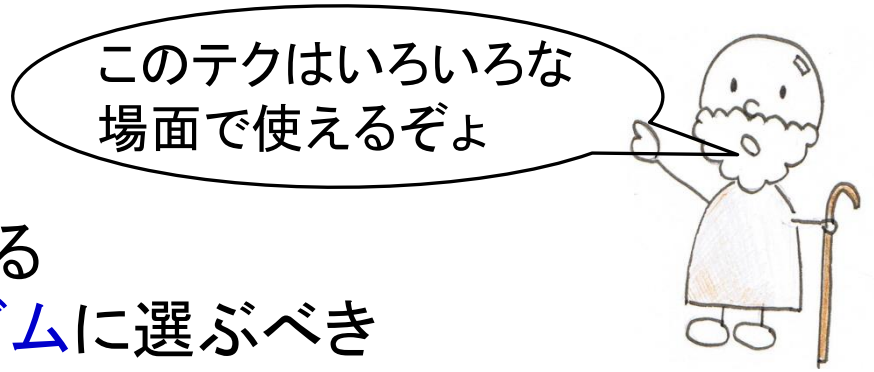
提出物と採点基準（満点 20）

1. 条件式　※グループ共通でよい (5)
2. その条件式の精度　※これもグループ共通でよい (5)
3. その条件式を導いた方法についての説明 (10)
   ※もちろんグループの構成員が同じ方法を使って
   求めるのは当然．でも，**説明は自分の言葉で書くこと**

以下はオプショナル（加点 ≦ 5）

4. 自分なりの解析 ⇒ 自分なりの条件式

2. データ解析のためのツール



(1) ランダムサンプルの作り方

- 元データは偏っている場合もある
- 条件式作成用のデータはランダムに選ぶべき
- 残りはテスト用に取っておこう

1つの有効な方法（1つのデータが1行になっている場合）

1. データの各行の先頭に (0,1) 区間の乱数を付ける
   ruby addrnd.rb < data4000org.txt > tmp.txt
2. tmp.txt をエクセルで見て先頭の行でソート
   - エクセルでセミコロン ; 区切りで開く
   - 最初の行（A 欄）でソートし，最初の行を削除
   - テキスト（タブ区切り）で保存（data4000.txt などの名前で）
3. 最初の n 行をランダムサンプルとして取り出す
   head -500 < data4000.txt > data500.txt ← 行数は適宜
4. 最後の行に 0 を付けておく ← 今回は行数がわからないので
   echo 0 >> data500.txt 最後の行の印が必要

2. データ解析のためのツール

・要するに数勘定の道具
・このくらいは，エクセル上でも可能
・もちろん，本職はもっと高級な道具を使います．でも，この程度でも結構できる

1. 条件式の精度テスト用プログラム

プログラム名：test.rb
使い方：・hantei = の右辺に条件式を書く
・ruby test.rb < data500.txt

↑ファイルの最後に 0 のみの行が必要
※データの行数が不定なので

2. 属性 $a_j == k$ と毒性との関係を調べるプログラム

プログラム名：count.rb
使い方：・調べたい属性と属性値でプログラムを修正
・ruby count.rb < data500.txt

test.rb　条件式の精度を検査するプログラム

```
a = Array.new(20)
stop = false
num = 0;  numpos = 0;  err = 0;  epos = 0;  eneg = 0    ← 各変数を 0 にセット
while !stop do
  a = gets().split.map(&:to_i)
  if a[0] == 0 then
    stop = true
  else

    num = num + 1
    if a[0] == 1 then  numpos = numpos + 1  end

    hantei = true          ← ここに条件式を書く（現在はすべて true）

    if  hantei && a[0] != 1  then
      epos = epos + 1      ← 偽陽性の数を増やす
      err = err + 1        ← 誤りの数を増やす
    end
    if  !hantei && a[0] != -1  then
      eneg = eneg + 1      ← 偽陰性の数を増やす
      err = err + 1        ←  誤りの数を増やす
    end
  end
end
```

1つのキノコのデータを入力
先頭行が0 か？（終了か？）を判断する

↓結果を画面に出力する部分へ続く

test.rb　条件式の精度を検査するプログラム（結果出力部分）

出力方法は見よう見まねでOKです

得られた結果
num = データ（キノコの）総数
numpos = 毒キノコの数，　numneg = num - numpos 無毒キノコの数
err ＝ 判定間違いをしたキノコの数
epos ＝ 偽陽性（間違って「毒」+1 と判定した）キノコの数
eneg ＝ 偽陰性（間違って「無毒」-1 と判定した）キノコの数

```
printf("*** statisitcs ***¥n")
printf("basic stat: num. = %d, pos.num. = %d (%3.2f)¥n",   ← 出力書式
        num, numpos, numpos.to_f / num.to_f)  ← 実際に出力する値
```

小数扱いにする

```
printf("    error:     %d (%3.2f)¥n", err, err.to_f / num.to_f)
printf("    false positive: %d (%3.2f)¥n", epos, epos.to_f / numneg.to_f)
printf("    false negative: %d (%3.2f)¥n", eneg, eneg.to_f / numpos.to_f)
```

補足：　画面に出力する簡便な方法として puts を使ってきたが，見易さを考えると，コメントや**書式**（出し方）を指定できる printf の方が便利．なお，最後の ¥n は改行のコード．¥ 記号は Mac では Option + ¥ キーで出せる．

count.rb　属性値と毒・無毒の関係を調べるための数勘定のプログラム

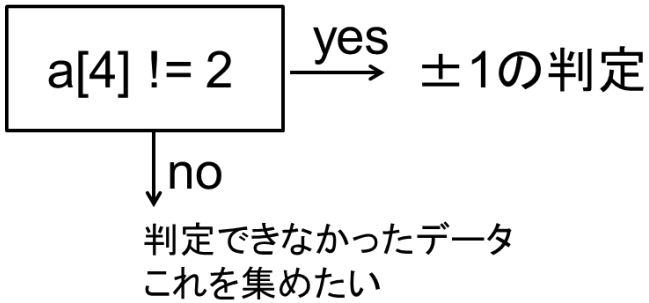
```
a = Array.new(20)
num = 0;  numpos = 0;  n42p = 0;  n42n = 0    ←  初期値 0 にセット
stop = false
while !stop do
  1 つのキノコデータを入力. 終わりの行か？のチェック（test.rb と同じ）
  else
    num と numpos の勘定をする（test.rb と同じ）num = num + 1

    # a4 == 2 と a0 = +1/-1 の関係を数えるための部分
    if a[4] == 2 then
      if a[0] == 1 then
        n42p = n42p + 1    ← a[4] == 2 かつ a[0] == 1 となるキノコの数を増やす
      else
        n42n = n42n + 1    ← a[4] == 2 かつ a[0] == -1 となるキノコの数を増やす
      end
    end
  end
end
# 集計と出力
n42 = n42p + n42n              ← a[4] == 2 となるキノコの総数
r42p = n42p.to_f / n42.to_f    ← n42p の n42 に対する割合
r42n = n42n.to_f / n42.to_f    ← n42n の n42 に対する割合
結果を出力する
```

3. 宿題：データ解析のためのツール

自分たちの考えた条件式に合わなかったデータを抽出するプログラムを作成し使い方を示せ

使い方の説明　＋ プログラム
〆切：1月19日（次回の演習の始め）

- たとえば条件式が「$a_4 \neq 2$」だったとすると
- 「$a_4 \neq 2$」でないデータだけを抜き出すプログラム
- test.rb（条件式の精度を検査するプログラム）を参考に



まとめ 条件式(論理式)の Ruby での書き方

【基本関係】

関係	使用例	意味
==	x == y	x は y と等しい
!=	x != y	x は y は等しくない

【論理演算子】

論理記号	使用例	意味
&&	x && y	x と y の論理積 (両方が真のとき真)
||	x || y	x と y の論理和 (少なくとも一方が真のとき真)
!	!x	x の論理否定 (x が真のとき偽, x が偽のとき真)

まとめ Terminal 上のコマンド

命令	使用例	意味
mkdir	mkdir kadai2	kadai2 というフォルダ（部屋）を作る
cd	cd kadai2	kadai2 というお部屋に入る
	cd ..	上の（大きな）部屋に戻る
ls	ls	その部屋にあるファイルを表示する
cat	cat xxx.txt	ファイル xxx.txt の中身を見る（全部）
less	less xxx.txt	ファイル xxx.txt の中身を見る（部分的） 注）スペースキーで先，やめるのは q
head	head $-n$ xxx.txt	最初の n 行だけ表示
tail	tail $-n$ xxx.txt	最後の n 行だけ表示
rm	rm foo.rb	foo.rb を消す（戻らないので注意）
リダイレクト <	ruby xx.rb < aa	xx.rb を実行．入力は aa から取り込む
リダイレクト >	ruby xx.rb > bb	xx.rb を実行．結果は bb へ出す
echo + >>	echo 文字列 >> aa	文字列を aa の最後の行に付ける