知的情報処理
2. 似ているということ

櫻井彰人
慶應義塾大学理工学部

丸暗記とは

- 丸暗記といっても、暗記しただけでは、意味がない。
- 聞かれたら（暗記した内容を）、答えられないと意味がない。
  - 行動であれば、行動できればよいが、行動できなければ意味がない
- コンピュータであれば、どうする？

- これは愚問ですよね。コンピュータは、覚えれば忘れない。では、試してみましょう。

丸暗記学習するプログラム

- 「コンピュータ、ソフトなければ、ただの箱」ですから、学習するコンピュータというのは、学習するソフト（プログラム）が実行されるコンピュータである
- すなわち、「学習するコンピュータ」を作るとは、学習するプログラムを作るということ
  - 当たり前すぎて、ごめんなさい。
- そこで、丸暗記学習するプログラムを書いてみよう

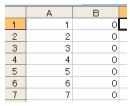
- プログラムはあと一回ぐらいしか表れません。
- Ruby を忘れた方は、見ていると思い出します。完全に忘れていてもOK。億劫がらずに、見ていてください。

消費税額を学習する

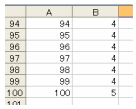
- 全ての可能な価格に対して、その消費税を覚えるわけにはいかない（何しろ無限個あるので）。そこで、有限個だけ覚えることにする。
- ちょっと非現実的だが、1円から100円までの価格について、その消費税を覚えることにしよう。
- コンピュータに覚えさせるために、まず、表を作ろう

学習データ・訓練データ

- 学習させることを、英語では、train ともいう。
- そして、そのために用いるデータを学習データまたは訓練データという
- 今回は、csv ファイルに作ろう



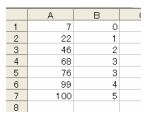
	A	B
1	1	0
2	2	0
3	3	0
4	4	0
5	5	0
6	6	0
7	7	0



	A	B
94	94	4
95	95	4
96	96	4
97	97	4
98	98	4
99	99	4
100	100	5

テストデータ

- （機械学習するプログラムが）ちゃんと学習できたかどうかを調べるデータ。
- 今回の、「消費税」学習では、価格と正解の組合せをいくつか。
- 例えば、



	A	B
1	7	0
2	22	1
3	46	2
4	68	3
5	76	3
6	99	4
7	100	5
8		

丸暗記プログラム in Ruby

- 次の方針でプログラムを書きます

要は、「53円の消費税はいくら？」と聞かれたら、
記憶の中から、53円を引き出し、対応する税金を思い出せばよい。

すなわち、kakaku の中から、ピッタリ、53になっている kakaku[i]をみつけ zei[i] を税金だと答えればよい

訓練データ ConsumptionTaxTra.csv を読み込んで、kakaku[i] に 第 i 番目の価格を記録し、zei[i] に第 i 番目の税額を記録する

↓

テストデータ ConsumptionTaxTes.csv を読み込んで、（第 i 番目の価格に対し）その価格に対応する税額を、kakaku と zei から求め、求めた税額が、テストデータ中の税額と合っていれば、OK、間違っていたら NG と印字する

簡単、簡単！

オブジェクト指向的ですが、まぁ、それは無視してください

```
# 丸暗記学習者 記憶
 def learn
  i=0
  CSV.open("./ConsumptionTaxTra.csv", 'r')
  { |row|
   @kakaku[i] = row[0]
   @zei[i]    = row[1]
   i = i+1
  }
 end

# 丸暗記学習者 想起
 def recall( test )
  (0..@kakaku.length-1).each
  { | i |
   if test==@kakaku[i] then
    return @zei[i]
   end
  }
  return -1
 end
end
```

丸暗記

鸚鵡返し

```
$ ruby ConsumptionTax.txt
test data is 7, 0
OK
test data is 22, 1
OK
test data is 46, 2
OK
test data is 68, 3
OK
test data is 76, 3
OK
test data is 99, 4
OK
test data is 100, 5
OK
```

```
require 'csv'
class RoteLearner
 def initialize
  @kakaku = Array.new(100)
  @zei    = Array.new(100)
 end

# テスト
ml = RoteLearner.new()
ml.learn()
CSV.open("./ConsumptionTaxTes.csv", 'r') { |row|
 testKakaku = row[0];   seikai    = row[1]
 puts "test data is " + testKakaku + ", " + seikai
 kekka = ml.recall( testKakaku )
 if kekka==-1 then
  puts "NG: I do not know."
 else
  if kekka==seikai then
   puts "OK"
  else
   puts "NG: " + kekka + " =!= " + seikai
  end
 end
}
```

丸暗記学習のポイント

- プログラムの中味はさておき、
- ポイントは、
  - 丸暗記と
  - オウム返し（丸思い出し(?)）

  である
- つまり何の工夫もない
- とはいえ、教えたことは着実にこなす

これって、学習？

- 学習の基本であることは確か。人間だって、
  - 文字を覚える
  - 数を覚える
  - 九九を覚える、
- しかし、人間（動物だって）の本質は応用力
  - 言語(母国語）は、実は、丸暗記されていない
    - 聞いたことがないことを発言する。
    - 時には、すばらしい創造力を発揮する
- この（学習における）応用力とは何か？

学習における応用力

- 応用力とは何だろう？
- 考え出すと限がないないので、基本中の基本は？

- 私の答え：「似た」状況に適用する。従って、
  - ぴったり正解であったり、
  - いつも正解であったりする

  わけではない。単に、
  - ほぼ正しい、
  - たいていの場合、正しい、

  あわせて、
  - たいていの場合、ほぼ、正しい

  ことを期待する

ところで、「似た」状況とは？

- 「似ている」かどうかというのは価値判断が入るので、「近い」という言葉にしよう（似たようなものだが、、、）
- 例えば、先ほどの消費税の例では、
  - 価格に小数があるときへの対応は？
    - 世の中には、「銭」単位の価格があるのです。
  - また、偶数のときだけ、覚えていたとしたら？
- 具体的には？

とにかく、決めてみよう

- 「近さ」は、（対象を表現する何らかの）値の差が小さいこと、であろう。
- 「対象を表現する値」とは、例えば、ある人を表す、身長、体重、生年月日、氏名、、、、
- こういったものを「属性」という
  - attribute とか feature とかいう
- 今の場合は、対象商品の価格。

- そこで、価格の差が小さければ、近いとしよう。
- 例えば、差が 0.5 円以下なら近いとしよう

では、プログラム

ちょっと修正

```
# 丸暗記学習者 記憶
 def learn
  i=0
  CSV.open("./ConsumptionTaxTra.csv", 'r')
  { |row|
   @kakaku[i] = row[0]
   @zei[i]   = row[1]
   i = i+1
  }
 end

# ちょっと工夫した想起
 def recall( test )
  (0..@kakaku.length-1).each
  { | i |
   if (test - @kakaku[i]).abs<0.5 then
    return @zei[i]
   end
  }
  return -1
 end
end
```

丸暗記

近いのを見つけたら、それにしよう

```
require 'csv'
class RoteLearner
 def initialize
  @kakaku = Array.new(100)
  @zei   = Array.new(100)
 end

# テスト
ml = RoteLearner.new()
ml.learn()
CSV.open("./ConsumptionTaxTes.csv", 'r') { |row|
 testKakaku = row[0];  seikai    = row[1]
 puts "test data is " + testKakaku + ", " + seikai
 kekka = ml.recall( testKakaku )
 if kekka==-1 then
  puts "NG: I do not know."
 else
  if kekka==seikai then
   puts "OK"
  else
   puts "NG: " + kekka + " =!= " + seikai
  end
 end
}
```

少々欠点が

- これは、「聞かれた値（test）と覚えている値（@kakaku[i]）との差が0.5より小さければ、それ（i番目に記憶した値）を用いる」というものである。しかし、
- 0.5は妥当か？ i.e. 0.5 では大きすぎる？小さすぎる？
- 基準を満たせばそれでいいのか？
  - もっとよいものはないのか？
- 基準が満たされなかったらどうなるのか？
  - 0.5 より近いものがない場合は？

解決案

- 「最も近いもの」にしよう。そうすれば、

- 「どのくらい近ければよい」という基準を考えなくてすむ。
- （適当に決めた基準で）見つからなかった場合にも対処できる

```
(0..@kakaku.length-1).each
 { | i |
  if (test - @kakaku[i]).abs<0.5
  then
   return @zei[i]
  end
 }
return -1
```

⇒

```
nearest=1
(0..@kakaku.length-1).each
 { | i |
  if (test - @kakaku[i]).abs < (test - @kakaku[nearest]).abs
   then
    nearest = i
   end
 }
return @zei[nearest]
```

近いものが複数あると？

- 「一つ」に決めてしまったが、ちょっと不安。
- 例えば、
  - 価格50.3円に対する消費税は？
    - 50.3円に近いのは、50円。従って、2円が税。
    - 51円に近い。従って、2円が税。
  - どっち？といっても税額は同じだからどちらでもいいか。
  - 59.6円のときは？
    - 59円に近いので、税は2円
    - 60円に近いので、税は3円
  - どっち？

いずれにせよ、ほぼ正しい

- ほぼ正しいのだから、どちらでもよいとは言える。
- しかし、少しでも正確な方がよい
  - そうしないと、「モラル」崩壊がありうる。
  - 強面のお兄さんが「59.1円ってほぼ60円だろ、消費税分3円、ほら、さっさと払え」というかもしれないし。
- どうするか？

解決方法の例

- 妥当性はあまり(「全然」ではない)ない、が、
- 比例配分する
  - 59.1円なら、
    2* (59.1–59)/(60–59) + 3* (60 – 59.1)/(60–59)
    とする
- 確率で、按分する。サイコロを振って、
  - 59.1円なら、確率 (59.1–59)/(60–59) で2円、確率 (60 – 59.1)/(60–59) で3円とする
    とする

うまくいったか？

- まぁね。
  - 税金の場合は、こんなことはないけど、例えば、身長から体重を推測するとか、円からドルに変換するとか、といった場合にはいいね。
- しかし、不安がある。
- 「近さ」の概念
  - 一般化されていない。例えば、属性(消費税の例では、価格)が一個だけであったが、複数個になったら、どうする？
- 結果の近似の仕方。
  - 一般にはどうする？そもそも近似していいの？
- 誤差
  - ほぼ正しいというが、誤差はどのくらいか？
- 誤る頻度
  - たいてい正しいというが、どのくらいの割合で、正しいのか？

まず、近さとは

- 近いとは「距離」が小さいこと。つまり、近さとは距離のこと
- では、距離とは？
- 例として、身長と体重で、AさんとBさんの近さをみることにしよう
  - Aさんは (h_A, w_A), Bさんは (h_B, w_B) としよう
- 距離は？
  - $\sqrt{((h_A–h_B)^2+(w_A–w_B)^2)}$? or $|h_A–h_B|+|w_A–w_B|$? or …
- そもそも単位は？
  - 身長は、cm? mm? μm? m? km?
  - 体重は、kg? g? mg? pg? t?

数学から

- 「近さ」とは「距離」である。
  - 同語反復。でもない。
  - 「距離」というと、数学では、距離の公理を満たす概念。
  - 距離の公理とは？
    - 非負性: $d(x,y)\geq 0$
    - 同一性: $x=y \Leftrightarrow d(x,y)=0$
    - 対称性: $d(x,y)=d(y,x)$
    - 三角不等式: $d(x,y)+d(y,z) \geq d(x,z)$
  - 代表的なものは:
    - ユークリッド距離: $\sqrt{((x_1-y_1)^2+(x_2-y_2)^2+(x_3-y_3)^2)}$
    - マンハッタン距離: $|x_1-y_1|+|x_2-y_2|+|x_3-y_3|$

解決？

- そうでもない。
- 例えば、「単位」に問題がある。
- 身長と体重を2個の属性とし、次の三人の距離をみよう
  - A: (170cm, 65kg)
  - B: (180cm, 60kg)
  - C: (175cm, 70kg)
- マンハッタン距離を考えよう
  - d(A,B)=10+5=15
  - d(B,C)=5+10=15
  - d(C,A)=5+5=10
- cm と kg を用いるのは不公平。単位の名称からいって、m と g だろう。となると、
  - d(A,B)=0.01+5000=5000.01
  - d(B,C)=0.005+10000=10000.005
  - d(C,A)=0.005+5000=5000.005
- 他にも、いくらでも、考えられる。さて、どうしたものか。

単位だけの問題ではない

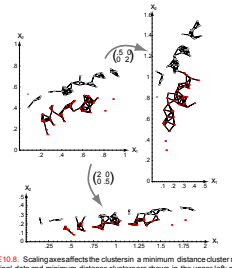
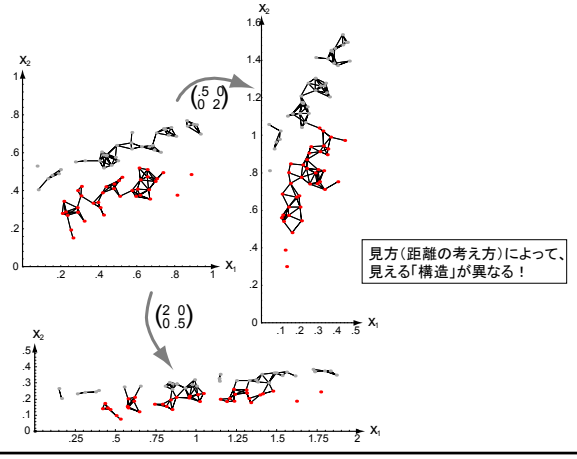


FIGURE 10.8. Scaling axes affects the clusters in a minimum distance cluster method. The original data and minimum-distance clusters are shown in the upper left; points in one cluster are shown in red, while the others are shown in gray. When the vertical axis is expanded by a factor of 2.0 and the horizontal axis shrunk by a factor of 0.5, the clustering is altered (as shown at the right). Alternatively, if the vertical axis is shrunk by a factor of 0.5 and the horizontal axis is expanded by a factor of 2.0, smaller more numerous clusters result (shown at the bottom). In both these scaled cases, the assignment of points to clusters differ from that in the original space. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification* . Copyright 2001 by John Wiley & Sons, Inc.



次元と単位

- 問題は2つある: 次元と単位
- 次元: 異なる物理量を区別する概念。次元が異なる物理量は比較さえできない。
  - 1kgと1mとは、どちらがより偉いか？
  - この例に表れる次元は3個。重さ、長さ、偉さ。
    - 「偉さ」が物理量かというと、まぁ、そうではないが。
- 単位: 物理量を数値で表すとき、数値の1に対応する物理量
  - 長さの単位の例: m, mile, 尺, Å
  - 勿論、1cm を単位にしても、3.14km を単位にしてもよい

次元と単位と距離

- 次元解析: 物理量を、質量、長さ、時間、電荷、温度などの次元の有理数冪で表現し、方程式や仮説の妥当性を調べること
- 目下の話題は、「次元」
  - 異なる次元の量は、比較・加算ができない。
    - 当然、(170cm, 65kg) と (180cm, 60kg) の距離をユークリッド距離的に、また、マンハッタン距離的には定義できない。
  - では、どうする？
- もう一つの話題は、「単位」
  - 異なる単位の量は、比較・加算ができない。
    - 身長と指の長さが属性のとき、(1.7m, 10cm) と (1.8m, 12cm) の距離をユークリッド距離的に、また、マンハッタン距離的には定義できない。
  - では、どうする？

もしデータが大量にあれば

- そして、各属性値が正規分布していれば（正規分布していなくとも）、「値/標準偏差」という量を使えばよい
  - この量は、無次元量。なぜなら、「値」もその「標準偏差」も同じ次元。
  - 「無次元量だから比較・加算できる」という訳ではないが、
  - 無次元の上に、各属性値の標準偏差が1に規格化されたのだから、よかろう

結果の近似の仕方

- 俗に言う「足して2で割る」は、線形補間して、中央を取ったもの
  - 国会の会期延長60日vs.30日。間をとって45日。
  - 線形補間に意味があるかどうかが問題。
  - 線形近似の精度が良いときには使える。
- 俗に言う「貸し借り」は、頻度で、中央を取ったもの。
  - 今日は勝たせるから、次回は勝たせろよ
  - 中間の値がなく、「足して2で割」れないときに有効。

2で割れず、一回勝負のとき

- 2で割れず、頻度も使えないときは？
- 多数決が次善の策
- 7人の賢者に意見を聞こう。意見が割れたら、多数に従おう。
- もし、賢者の意見が確率的に正しければ、これは、確率最大の意見に従おう！ということ
  - 「最尤推定法」という極めてまともな推定方法
  - Bayesを説明するときに説明します
- tie-break が課題だが、確率的には、どの意見に従うも可

http://bbs.enjoykorea.jp/tbbs/read.php?board_id=tanimation&nid=367033&

では、何人の意見を聞くか？

- 「距離」0.5 以内、などというのがよいように思えるが、この「0.5」などの値の決め方が難しい
- そこで、近い方から3人（とか5人とか7人とか、、、）の意見を聞くことにすればよい。
- 近くの k 人から聴くことにした、この方法を k-nearest neighbor 法という

プログラムはちょっと複雑

```
# 丸暗記学習者 起起
 def mostFrequent( nearest )
  uniq = nearest.uniq
  work = nearest.dup
  f = 0
  fitem=uniq[0]
  uniq.length.times{ |i|
   ftmp = work.length; work.delete(uniq[i])
   ftmp = ftmp - work.length
   if ftmp>f then
    f = ftmp
    fitem = uniq[i]
   end
  }
  return fitem
 end
```

```
def recall( test, k )
   nearest = Array.new( k, 0 )
   (0..@kakaku.length-1).each { | i |
    if (test.to_f - @kakaku[i].to_f).abs <
       (test.to_f - @kakaku[nearest[k-1]].to_f).abs  then
     j=0
     (0..nearest.length-1).each{ | v |
      if (test.to_f - @kakaku[i].to_f).abs <
         (test.to_f - @kakaku[nearest[v]].to_f).abs  then
       j = v
       break
      end
     }
     nearest.insert( j, i ).delete_at(-1)
    end
   }
   return mostFrequent( nearest.map!{ | i | @tax[i] } )
  end
```

まとめにかえて

用語について

- 「データ」は多義なので、機械学習では
  - 事例、サンプル、インスタンス
  - 学習事例、学習サンプル、学習インスタンス
  - テスト事例、テストサンプル、テストインスタンス

  といった言葉が用いられる。
- 機械学習の識別問題では、2つのクラスに分けることが基礎となる（例えば、病気と健康）。片方がある目的のクラスで他方がそれ以外ということがよくある。そこで、
  - 目的のクラスに属する事例: 正事例 positive instance
  - 目的外のクラスに属する事例: 負事例 negative instance

  というように分けて呼ぶことが多い。

データの例

一つ一つがサンプル

一つ一つが属性

特別な属性 クラス 分類先 予測する種別

Make	Model	Year	Head inj. c.	Chest decel.	L. Leg	R. Leg	D/
Acura	Integra	87	599	35	791	262	Driver
Acura	Integra RS	90	585		1545	1301	Driver
Acura	Legend LS	88	435	50	926	708	Driver
Audi	80	89	600	49	168	1871	Driver
Audi	100	89					
BMW	325i	90					
Buick	Century	91					
Buick	Elect. Park	88					
Buick	LeSabre	90					

Date	Open	High	Low	Close	Volume	Adj. Cl	YearWeek
27/11/2000	53.6875	54.5156	51.0312	51.25	40198100	51.250	200049
28/11/2000	51.9375	53.1875	50.625	51	52037000	51.000	200049
29/11/2000	51.3125	53	50.3125	51.6875	55316000	51.688	200049
30/11/2000	50.1875	50.9375	45.1875	47.875	10840500	47.875	200049
01/12/2000	49.1875	51.625	47.25	48.5	70468000	48.500	200049
04/12/2000	49.0625	49.5625	45	45.8125	9501200	45.813	200050
05/12/2000	47.75	52.125	47.3125	52.125	90848900	52.125	200050
06/12/2000	52	53.5625	51.2656	51.4375	71419200	51.438	200050
07/12/2000	50.3125	51	49	49.9375	46448400	49.938	200050
08/12/2000	51.9375	53.25	51	52.375	55400200	52.375	200050
11/12/2000	52.875	55.75	52.625	54.8125	78621500	54.813	200051
12/12/2000	54.75	55.125	53.3125	54.375	39485300	54.375	200051
13/12/2000	55.1875	55.25	50.8125	51.125	54330600	51.125	200051
14/12/2000	51.0625	52.5625	50.875	50.9375	46244400	50.938	200051
15/12/2000	50.0625	50.1875	47.125	48.1719	100237900	48.172	200051
18/12/2000	49	50.125	42.3125	42.9375	126032400	42.938	200052
19/12/2000	43	46	41.5	41.75	99018800	41.750	200052

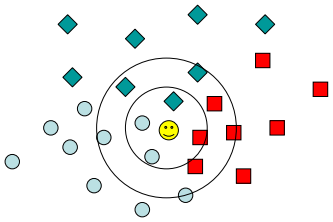
萼片長	萼片幅	花弁長	花弁幅	種別
5.1	3.5	1.4	0.2	Iris-s
4.9	3	1.4	0.2	Iris-s
4.7	3.2	1.3	0.2	Iris-s
4.6	3.1	1.5	0.2	Iris-s
5	3.6	1.4	0.2	Iris-s
5.4	3.9	1.7	0.4	Iris-s
4.6	3.4	1.4	0.3	Iris-s
5	3.4	1.5	0.2	Iris-s
4.4	2.9	1.4	0.2	Iris-s
4.9	3.1	1.5	0.1	Iris-s
5.4	3.7	1.5	0.2	Iris-s
4.8	3.4	1.6	0.2	Iris-s

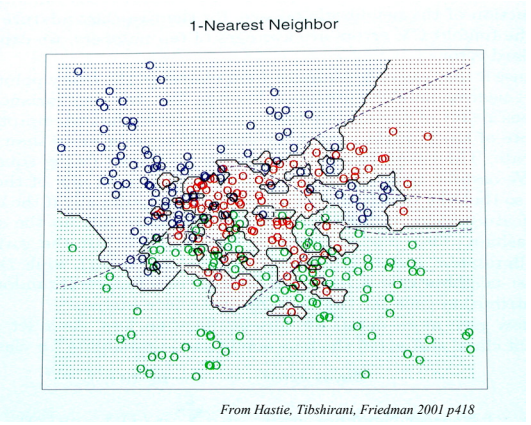
通算	曜日	室温	前夕のアルコール量	血圧(mmHg) (LOW)	(HIGH)
0	火	18	なし	107	153
1	水	20	少々	78	132
2	木	20	少々	92	133
3	金	20	少々	87	130
5	日	20	少々	86	134
6	月	20	適度	90	134
7	火	18	少々	87	134
8	水	18	少々	104	149
9	木	20	少々	83	130
10	金	20	適度	94	141

事例ベース

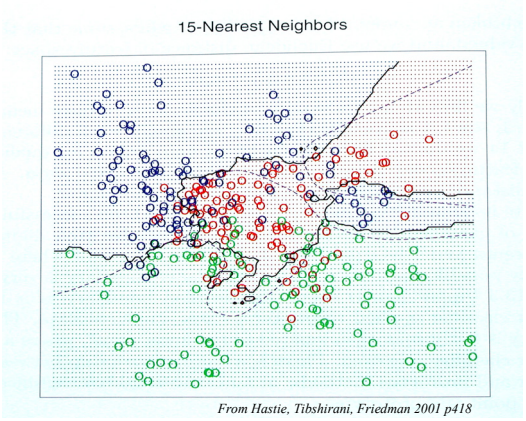
- このように、学習データをそのまま記憶し、推論しなければならないときに、必要な計算を行う推論方式を、事例ベース推論という
  - case/memory/instance-based reasoning/inference. 学習に重点があれば、instance/case/memory-based learning
- k-nearest neighbor は、事例ベース学習の一つであり、類似事例の選択方法が「k-nearest neighbor」である方法である。
- 簡単なようで奥が深い。「距離」の定義が難しい場合にも適用が試みられている
  - 「事例ベース翻訳」もある

k-nearest neighbor





From Hastie, Tibshirani, Friedman 2001 p418



From Hastie, Tibshirani, Friedman 2001 p418

From Hastie, Tibshirani, Friedman 2001 p419