

データマイニング

第8回 非線形判別分析

2023年春学期

宮津和弘

本日の講義・演習

日付	講義・演習内容
04/14/23	(1) イントロダクション
04/21/23	(2) ビジネスシミュレーション
04/28/23	(3) ID-POSデータ分析
05/12/23	(4) 対応分析
05/19/23	(5) クラスター分析
05/26/23	(6) 自己組織化マップ
06/02/23	(7) 線形判別分析
06/09/23	(8) 非線形判別分析
06/16/23	(9) ツリーモデル
06/23/23	(10) 集団学習
06/30/23	(11) サポートベクターマシン
07/04/23	(12) ネットワーク分析
07/14/23	(13) 共分散構造分析
07/21/23	(14) テキスト分析
07/28/23	(15) まとめ



本日の演習概要とポイント



- **非線形関数とロジット回帰モデル**

→ カラーテレビ普及率、企業の倒産確率モデル


- **非線形判別の演習**

→ ナイーブベイズ、ニューラルネットワーク





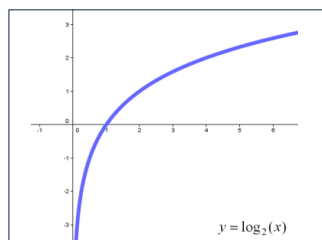
非線形回帰分析



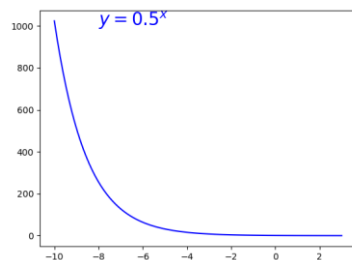
非線形回帰とは？

目的変数が説明変数と**非線形な関係**で表されるものを非線形回帰という。

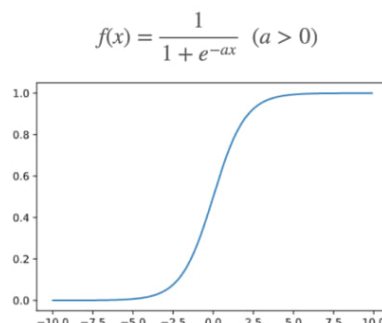
非線形関数の例



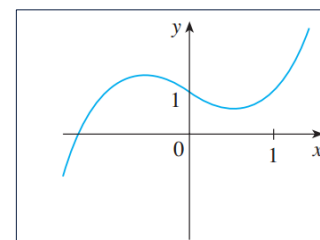
対数関数



指数関数

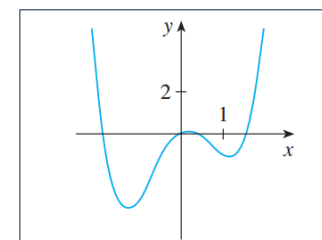


ロジスティック関数



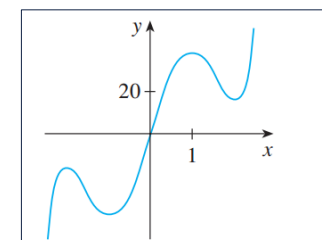
(a) $y = x^3 - x + 1$

3次関数



(b) $y = x^4 - 3x^2 + x$

4次関数



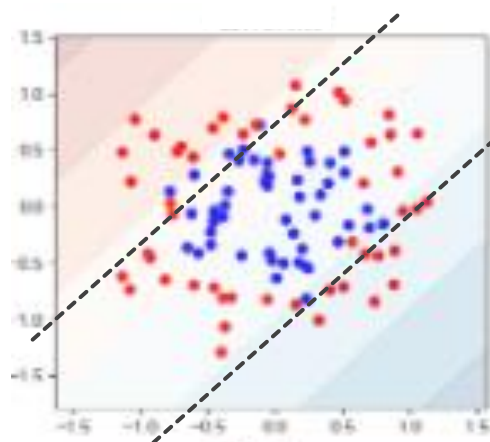
(c) $y = 3x^5 - 25x^3 + 60x$

5次関数

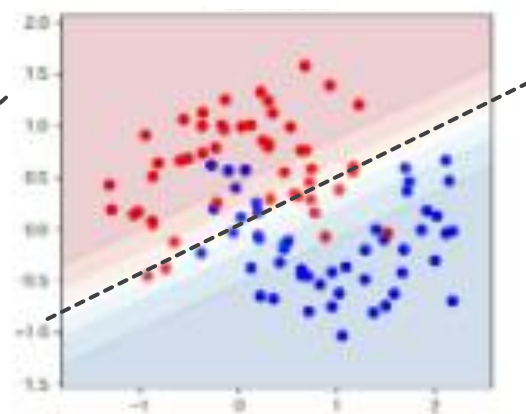
多項関数

データ判別の境界（再掲）

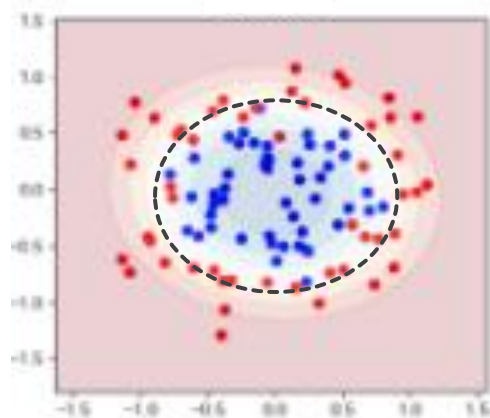
判別境界を**線形**で表現



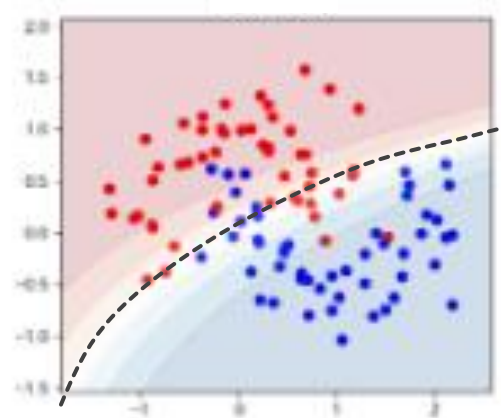
判別境界を**線形**で表現



判別境界を**二次形式**で表現

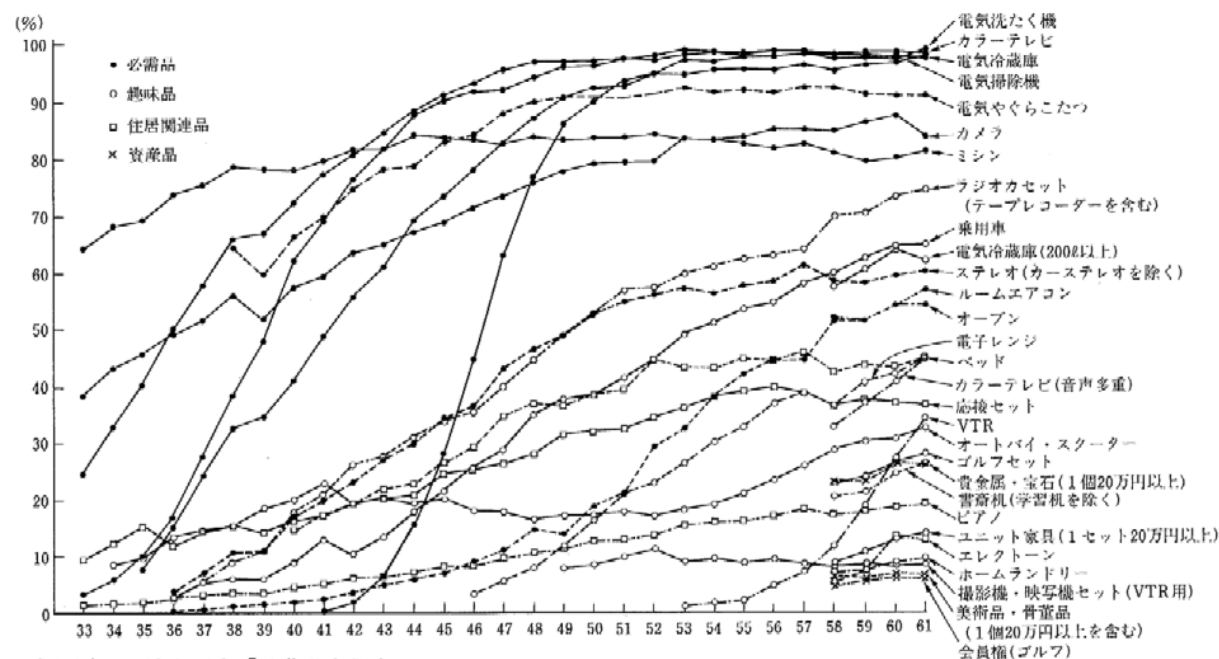


判別境界を**非線形**で表現



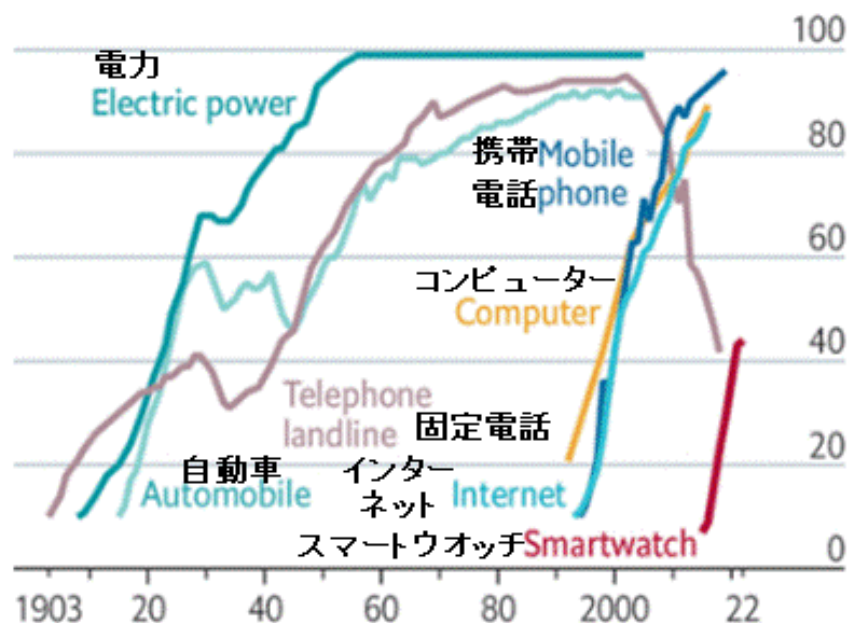
世の中は非線形な現象ばかり！

第2-33図 耐久消費財の普及率（全世帯）



(備考) 経済企画庁「消費動向調査」による。

米国における新技術の普及(世帯普及率%)



(注) Asymco; D. Comin and B. Hobijn, 2004; Deloitte
(資料) The Economist May 7th 2022

ちなみに、線形関数とは（復習）

関数 f が**線形**であるには、**加法性**と**斉次性**を満たす必要がある。

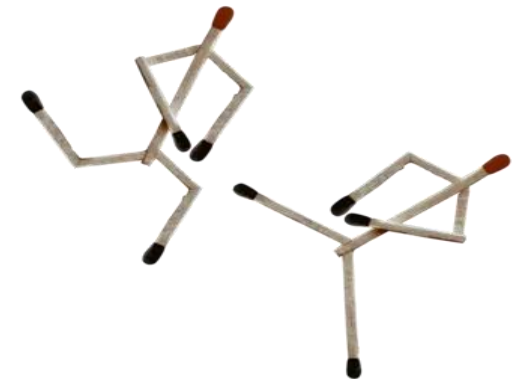
任意の x, y と任意のスカラー k に対して

■ **加法性** $f(x + y) = f(x) + f(y)$

■ **斉次性** $f(kx) = kf(x)$

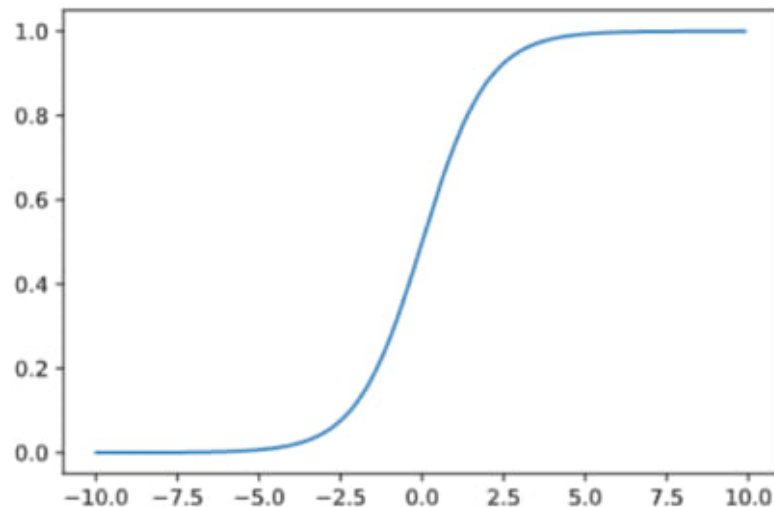
平たく言うと、**一次関数は線形関数**である。

$$y = a_0 + b_1x_1 + b_1x_1 + \cdots + b_Nx_N$$



ロジスティック回帰

$$f(x) = \frac{1}{1 + e^{-ax}} \quad (a > 0)$$



ここに数式を入力します。出力が $[0,1]$ となるようなデータを扱う**非線形モデル**

→ 普及率、出現確率、占有率など割合(%)

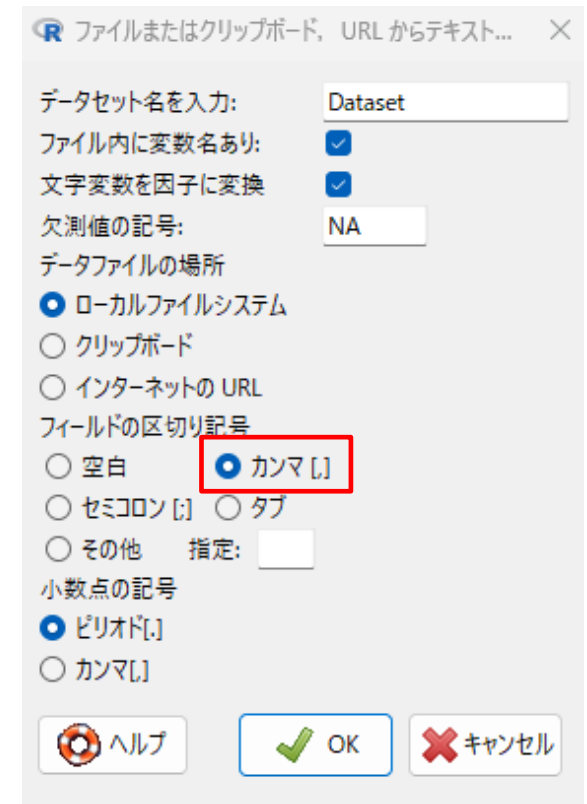
質的データ (1,0) を扱うような場合にも用いる

→ 企業倒産有無、商品選択有無など

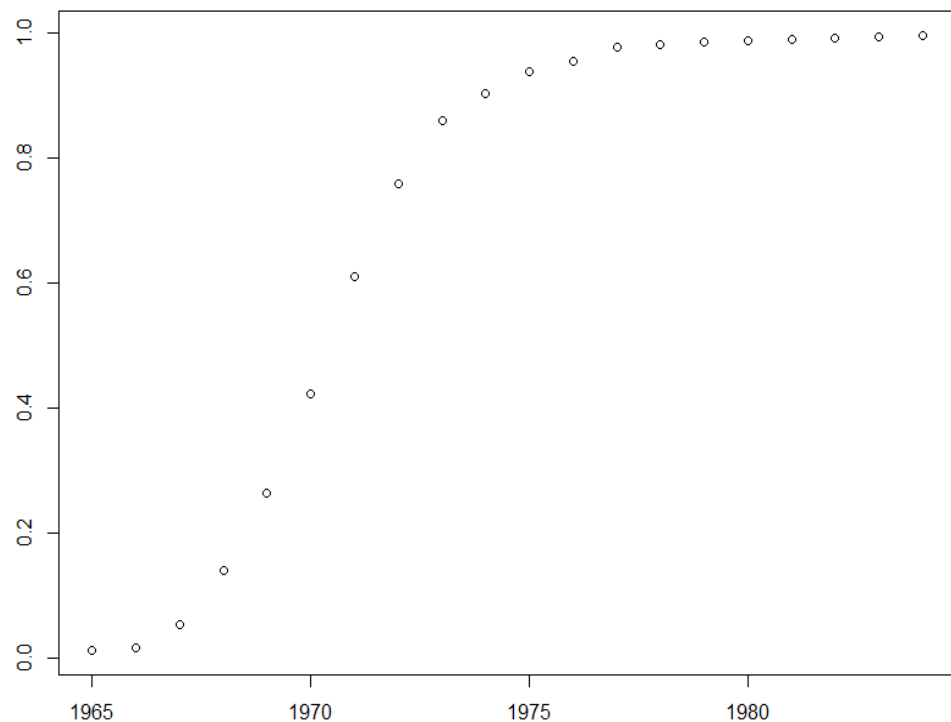
$$\frac{1}{1 + e^{-ax}} = \begin{cases} 1 & x \rightarrow +\infty \\ 0 & x \rightarrow -\infty \end{cases} \quad a > 0$$

演習データの読み込み

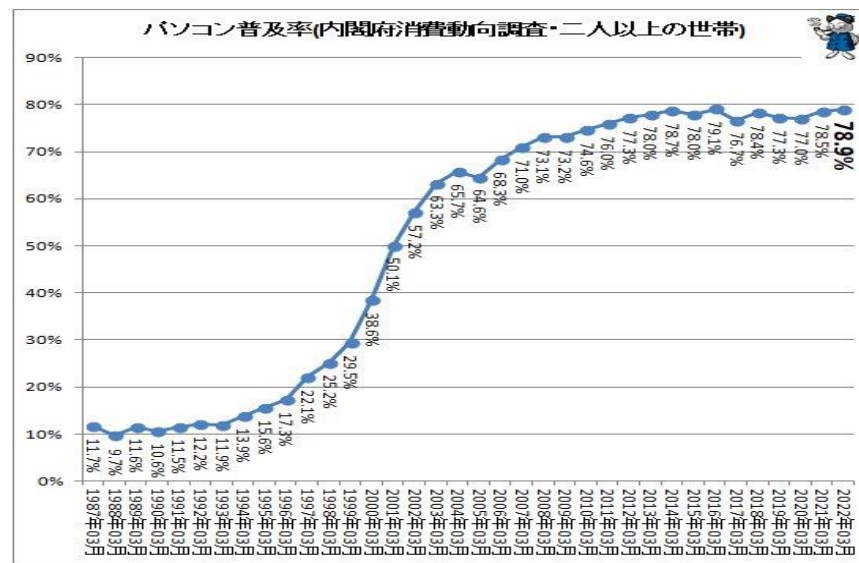
- ① Rstudio起動する
- ② `> library(Rcmdr)` ※コマンドラインから Rコマンダー を起動する
- ③ 演習ファイル “TV_ratio.csv” を読み込む
 - Rstudio `> Dataset<-read.csv(“TV_ratio.csv”)`
又は
 - Rコマンダー (データ) → (データインポート) → (テキストファイルまたはクリップボード...) →
✓ OKを選択して、TV_ratio.csv を指定する
- ④ 演習データが Dataset に読み込まれる
 - Datasetをtvにコピーして使う
`> tv<- Dataset`



世帯におけるカラーテレビの普及率データ



東京オリンピック(1964)から徐々に家庭への**カラーテレビ普及が高まり**現在に至る。同様なその他の傾向として、パソコン普及率などがある。



カラーテレビ普及率モデルの推定結果

```
> z.tv<-nls(TV_ratio~a/(1+b*exp(c*1:20)),data=tv,start=c(a=1,b=1,c=-1))
> summary(z.tv)
```

Formula: TV_ratio ~ a/(1 + b * exp(c * 1:20))

Parameters:

	Estimate	Std. Error	t value	Pr(> t)
a	0.983961	0.003637	270.572	< 2e-16 ***
b	118.889839	12.772753	9.308	4.38e-08 ***
c	-0.747479	0.016966	-44.058	< 2e-16 ***

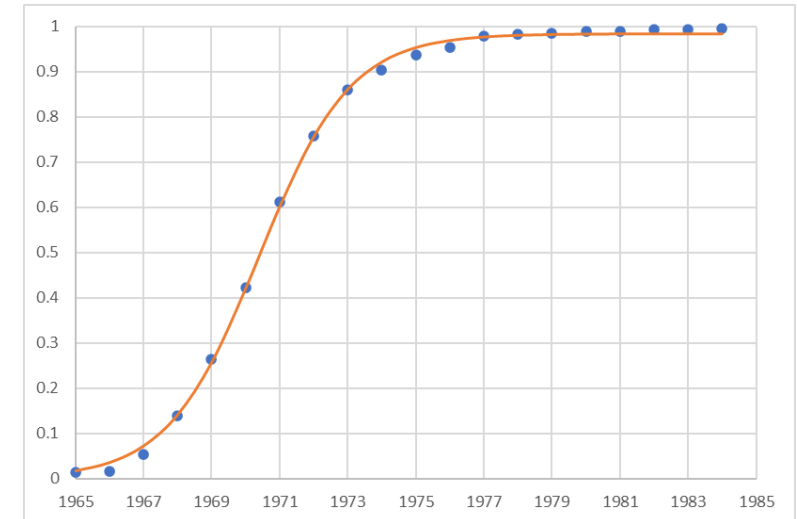
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.01103 on 17 degrees of freedom

Number of iterations to convergence: 15

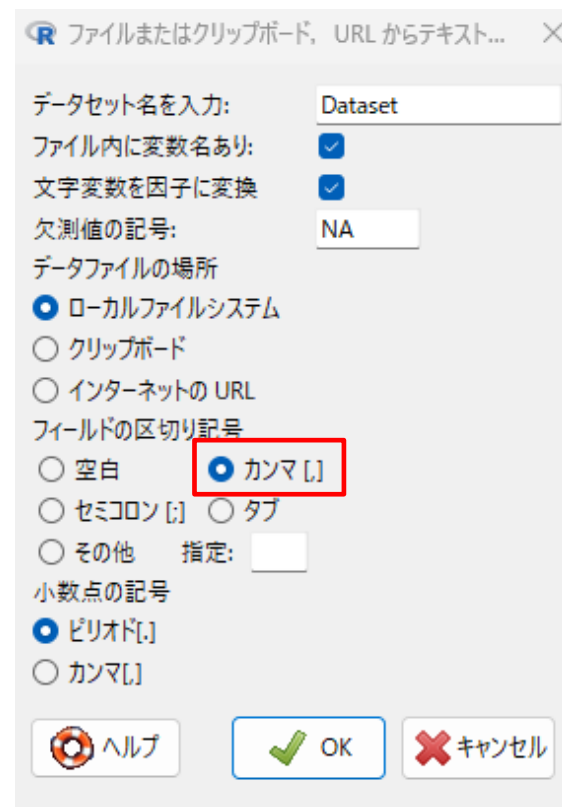
Achieved convergence tolerance: 2.722e-06

$$y = \frac{0.984}{1 + 118.9e^{-0.747}}$$



演習データの読み込み

- ① Rstudio起動する
- ② `> library(Rcmdr)` ※コマンドラインから Rコマンダー を起動する
- ③ 演習ファイル “credit_risk.csv” を読み込む
 - Rstudio `> Dataset<-read.csv(“credit_risk.csv”)`
又は
 - Rコマンダー (データ) → (データインポート) → (テキストファイルまたはクリップボード...) →
✓ OKを選択して、credit_risk.csv を指定する
- ④ 演習データが Dataset に読み込まれる



企業倒産確率をモデル化する

```
> Dataset<-read.csv("credit_risk.csv")
> Dataset
```

	x1	x2	z
1	15.50	-0.25	0
2	5.60	-0.31	0
3	28.30	-1.00	0
4	47.00	-4.66	0
5	-78.70	-0.52	0
6	-4.79	1.16	0
7	-0.63	1.81	0
8	14.16	-1.11	0
9	50.00	1.24	0
10	13.69	0.73	0
11	8.61	2.93	0
12	32.00	1.98	0
13	10.94	-0.16	0
14	8.68	0.95	0
15	-5.21	1.43	0
16	-1.86	1.98	0
17	11.40	2.30	1
18	13.70	1.93	1
19	30.25	2.43	1
20	34.06	5.50	1

z : 企業倒産フラグ (0 : 倒産、1 : 非倒産)

x1: 自己資本比率

→ 企業の総資本のうち純資産の占める割合であり、この割合が高いとは、負債(返済しなければならない資本)が少ないことを意味する

x2: インタレストカバレッジ比率

→ 営業活動と財務活動で得た利益がインタレスト(支払い利息など)をカバーできている割合のこと (最低条件は 1、適正なのは 2 以上、5 以上が望ましいと言われる)

ロジット回帰による倒産確率モデル

ロジットモデルで使用するロジット曲線は企業の評価スコア Z に対し、0から1の間の倒産（デフォルト）確率を返す役割を担います

$$PD = \frac{1}{1 + \exp(-Z)}$$

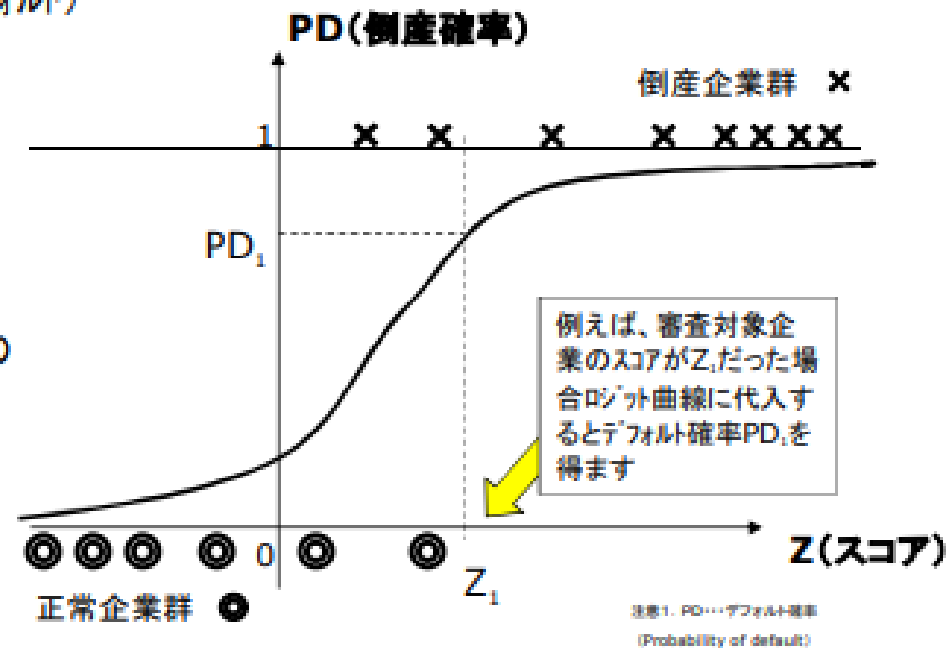
なお、評価スコア Z は説明変数 X 及びそのウェイト β で構成されます

$$Z = \beta_1 \times X_1 + \beta_2 \times X_2 + \dots$$

(評価スコア Z の例)

2次モデルでは説明変数 X の2次の項までスコアの式を精緻化しています

※ 演習では、 X_1 と X_2 に前頁の変数を用いる



倒産推定モデルと検証

```
> Dataset<-read.csv("credit_risk.csv")
> risk<-Dataset[2:49,]
> z.risk<-glm(z~x1+x2,family=binomial(link="logit"),data=risk)
> summary(z.risk)
```

```
call:
glm(formula = z ~ x1 + x2, family = binomial(link = "logit"),
    data = risk)
```

```
Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.91581  -0.40375   0.05725   0.17296   2.17051
```

```
Coefficients:
```

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	-2.74833	1.02183	-2.690	0.00715 **
x1	0.07575	0.03513	2.156	0.03107 *
x2	0.50157	0.21583	2.324	0.02013 *

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be 1)
```

```
Null deviance: 59.624  on 47  degrees of freedom
Residual deviance: 21.654  on 45  degrees of freedom
AIC: 27.654
```

```
Number of Fisher Scoring iterations: 7
```

本モデルによる評価スコア z :

$$z = -2.7483 + 0.07573x_1 + 0.50157x_2$$

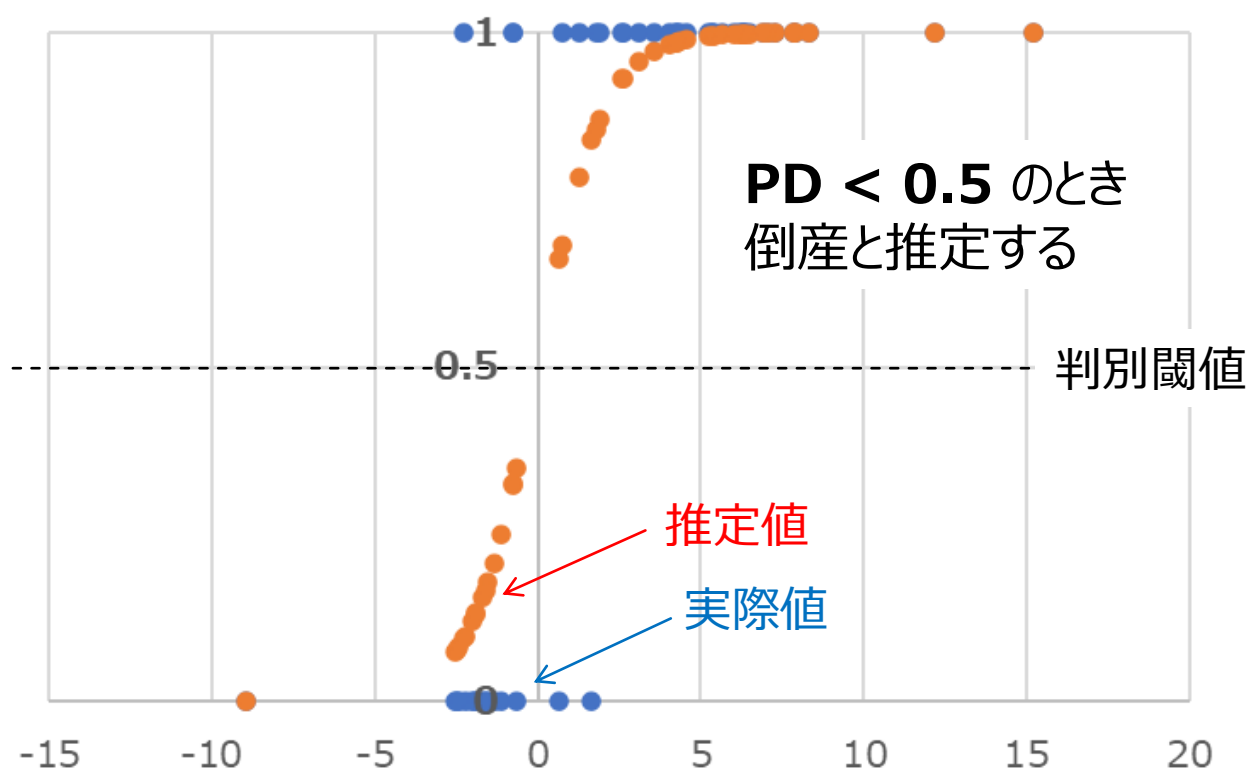
$$(x_1, x_2) = (15.5, -0.25) \rightarrow z = -1.700$$

\therefore **PD=0.154 (倒産)**

$$(x_1, x_2) = (41.9, 17.63) \rightarrow z = 4.252$$

\therefore **PD=0.986 (非倒産)**

倒産確率モデルの結果とデータ当てはまりの検証



	実際値	
	0	1
推定値	0	3
	13	2

モデルデータに対する**正答率**
→ 89.6%

さらに、2 件の検証データに対して
→ 2 件とも正しく推定された

ナイーブベイズとニューラルネットワーク

前回課題：SPAMデータを用いた線形判別

SPAM1で提供された7つの特徴量を用いて、ldaコマンドを用いて線形判別モデルを構築・評価しなさい。

SPAMとHAMの各サンプルから半分ずつを学習データとして線形判別モデルを構築し、残り半分を検証データを用いて判別能力を評価・考察しなさい。



迷惑メール

vs.



通常メール

	remove	free	email	you	charExclamation	capitalAve	capitalTotal	type
1	0.00	0.32	1.29	1.93	0.778	3.756	278	spam
2	0.21	0.14	0.28	3.47	0.372	5.114	1028	spam
3	0.19	0.06	1.03	1.36	0.276	9.821	2259	spam
4	0.31	0.31	0.00	3.18	0.137	3.537	191	spam
5	0.31	0.31	0.00	3.18	0.135	3.537	191	spam
6	0.00	0.00	0.00	0.00	0.000	3.000	54	spam
7	0.00	0.96	0.32	3.85	0.164	1.671	112	spam
8	0.00	0.00	0.00	0.00	0.000	2.450	49	spam
9	0.30	0.00	0.15	1.23	0.181	9.744	1257	spam
10	0.38	0.00	0.12	1.67	0.244	1.729	749	spam
11	0.96	0.00	0.96	3.84	0.462	1.312	21	spam
12	0.25	0.00	0.00	1.16	0.663	1.243	184	spam
13	0.00	0.34	1.39	2.09	0.786	3.728	261	spam
14	0.90	0.00	0.00	2.72	0.000	2.083	25	spam
15	0.00	5.35	0.00	3.21	0.357	1.971	205	spam
16	0.42	1.27	0.00	1.70	0.572	5.659	249	spam
17	0.00	0.00	0.00	1.88	0.428	4.652	107	spam
18	0.00	0.00	0.00	0.00	1.975	35.461	461	spam
19	0.18	0.00	0.37	3.15	0.455	1.320	70	spam
20	0.00	0.63	3.18	2.22	0.055	3.509	186	spam

前回課題解答：SPAMデータを用いた線形判別

```
> lab.spam<-c(rep("F",500),rep("T",1000))
> dataset<-data.frame(data[,1:7],lab.spam)

> even.n<-2*(1:750)-1
> data.train<-dataset[even.n,]
> data.test<-dataset[-even.n,]
```

```
> z.lda<-lda(lab.spam~.,data=data.train)
> table(data.train[,8],predict(z.lda)$class)
```

	F	T
F	147	103
T	32	468

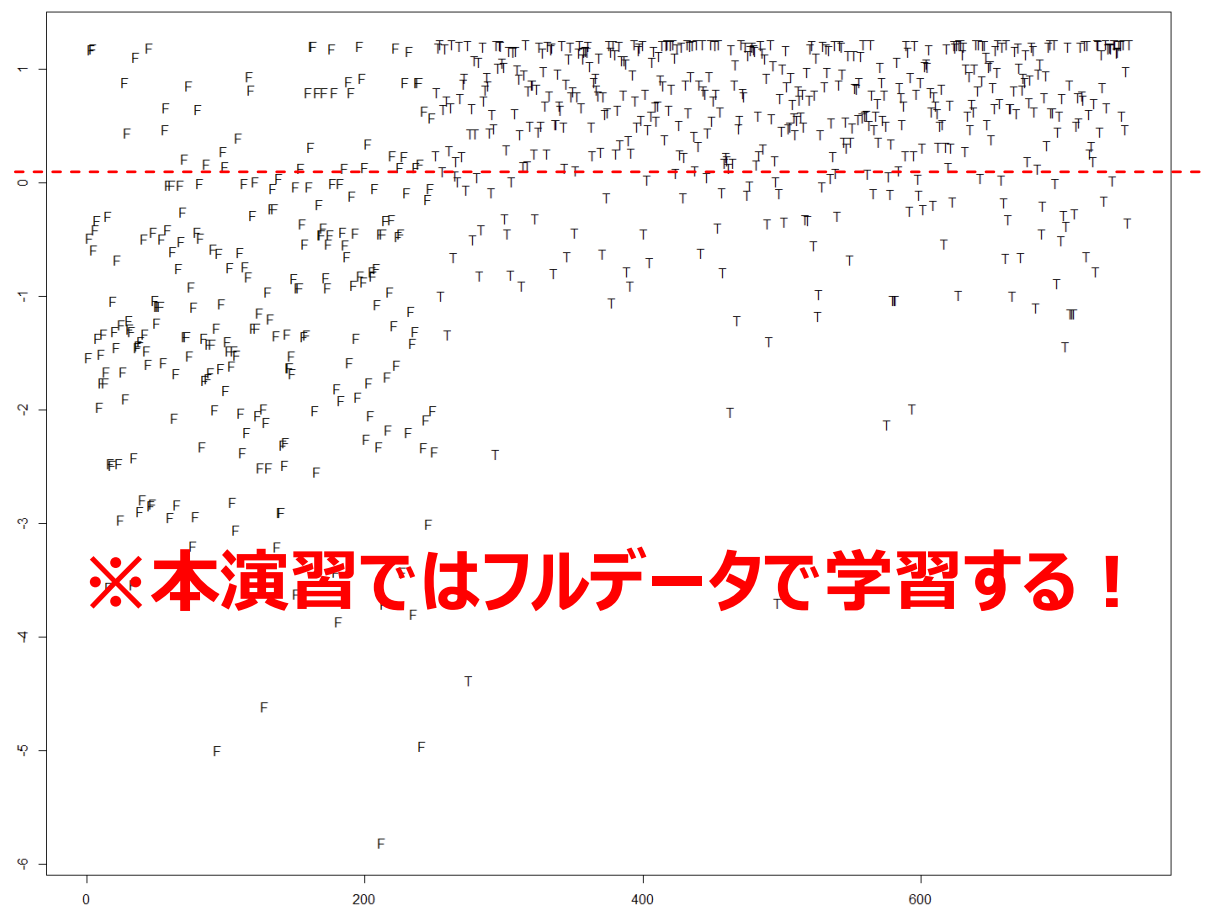
線形判別では精度が良く出ない！

```
> y<-predict(z.lda,data.test[, -8])
> table(data.test[,8],y$class)
```

	F	T
F	149	101
T	33	467

線形判別では精度が良く出ない！

```
> plot(y$x,type="n")
> text(y$x,labels=data.test$lab.spam)
```



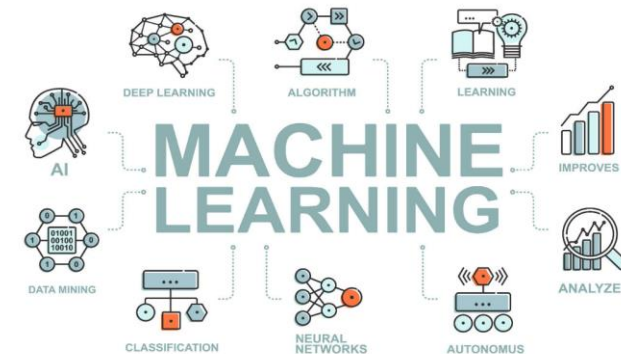
機械学習の手法

教師データあり

- (一般化)線形回帰
- ロジスティック回帰
- サポートベクターマシン(SVM)
- 決定木 (CART)
- 回帰木
- ランダムフォレスト
- 勾配ブースティング木
- ニューラルネットワーク(パーセプトロン)
- 畳み込みニューラルネットワーク(CNN)
- 再起型ニューラルネットワーク(RNN)
- 単純ベイズ (ナイーブベイズ)
- k近傍法(KNN)
- ブースティング
- バギング

教師データなし

- 階層型クラスタリング(ウォード法など)
- 非階層型クラスタリング (k-meansなど)
- トピックモデル (LDAなど)
- 自己組織化マップ (SOM)
- アソシエーション分析
- 協調フィルタリング (ユーザベースなど)



SPAMデータの特徴量

実際のSMSメッセージを57項目の特徴量で定量化したデータを入力とする

> head(spam)

	make	address	all	num3d	our	over	remove	internet	order	mail	receive	will	people	report	addresses	free	business	email	you	credit	your	font	num000
1	0.00	0.64	0.64	0	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.64	0.00	0.00	0.00	0.32	0.00	1.29	1.93	0.00	0.96	0	0.00
2	0.21	0.28	0.50	0	0.14	0.28	0.21	0.07	0.00	0.94	0.21	0.79	0.65	0.21	0.14	0.14	0.07	0.28	3.47	0.00	1.59	0	0.43
3	0.06	0.00	0.71	0	1.23	0.19	0.19	0.12	0.64	0.25	0.38	0.45	0.12	0.00	1.75	0.06	0.06	1.03	1.36	0.32	0.51	0	1.16
4	0.00	0.00	0.00	0	0.63	0.00	0.31	0.63	0.31	0.63	0.31	0.31	0.31	0.00	0.00	0.31	0.00	0.00	3.18	0.00	0.31	0	0.00
5	0.00	0.00	0.00	0	0.63	0.00	0.31	0.63	0.31	0.63	0.31	0.31	0.31	0.00	0.00	0.31	0.00	0.00	3.18	0.00	0.31	0	0.00
6	0.00	0.00	0.00	0	1.85	0.00	0.00	1.85	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0	0.00

	money	hp	hpl	george	num650	lab	labs	telnet	num857	data	num415	num85	technology	num1999	parts	pm	direct	cs	meeting	original	project	re	edu	table
1	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0.00	0	0	0.00	0	0	0.00	0	0.00	0.00	0
2	0.43	0	0	0	0	0	0	0	0	0	0	0	0	0.07	0	0	0.00	0	0	0.00	0	0.00	0.00	0
3	0.06	0	0	0	0	0	0	0	0	0	0	0	0	0.00	0	0	0.06	0	0	0.12	0	0.06	0.06	0
4	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0.00	0	0	0.00	0	0	0.00	0	0.00	0.00	0
5	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0.00	0	0	0.00	0	0	0.00	0	0.00	0.00	0
6	0.00	0	0	0	0	0	0	0	0	0	0	0	0	0.00	0	0	0.00	0	0	0.00	0	0.00	0.00	0

	conference	charsemicolon	charRoundbracket	charsquarebracket	charExclamation	charDollar	charHash	capitalAve	capitalLong	capitalTotal	type
1	0	0.00	0.000	0	0.778	0.000	0.000	3.756	61	278	spam
2	0	0.00	0.132	0	0.372	0.180	0.048	5.114	101	1028	spam
3	0	0.01	0.143	0	0.276	0.184	0.010	9.821	485	2259	spam
4	0	0.00	0.137	0	0.137	0.000	0.000	3.537	40	191	spam
5	0	0.00	0.135	0	0.135	0.000	0.000	3.537	40	191	spam
6	0	0.00	0.223	0	0.000	0.000	0.000	3.000	15	54	spam

“正常メール”



VS.



“スパムメール”のラベル



H2Oパッケージのディープラーニングを実行する

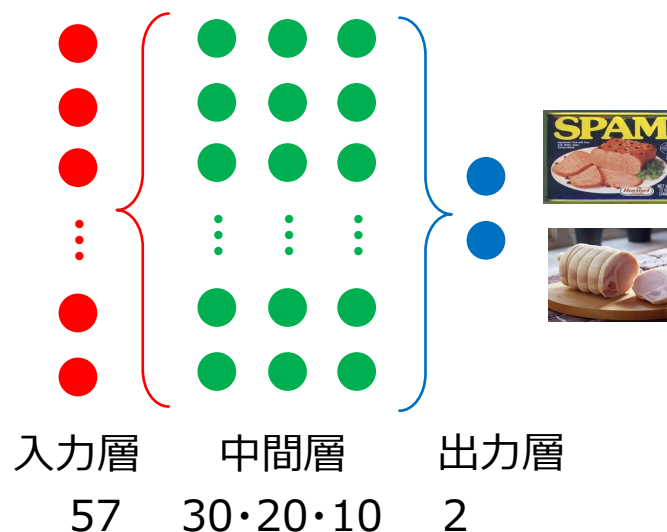
```
> library(kernlab)
> data(spam)
> library(h2o)
> h2o.init()
> even.n <- 2*(1:2300)-1
> spam.train <- spam[even.n,]
> spam.test <- spam[-even.n,]
> spam.train.hex <- as.h2o(spam.train)
> spam.test.hex <- as.h2o(spam.test)
> tr1 <- h2o.deeplearning(x=1:57,y=58,training_frame=spam.train.hex, hidden=c(30,20,10), epochs=100, nfolds=3)
```

※ 訓練データは 1 ~ 5 7 番目、教師データは 5 8 番目を用いる

※ 3つの中間層のノード数が30,20,10であること

※ エポック数 (epoch) とは、学習用の訓練データを何回繰り返し学習するか
→ epoch=100 (回)

※ 訓練データの全てを用いるのではなく、ランダムに抽出して複数回学習した結果の平均値を算出
→ nfolds=3 (回)



H2Oニューラルネットワーク分析の結果

```
> tr1
Model Details:
=====

H2OBinomialModel: deeplearning
Model ID: DeepLearning_model_R_1686130560088_1
Status of Neuron Layers: predicting type, 2-class classification, bernoulli distribution, CrossEntropy loss, 2,592 weights/biases
size 1
```

	layer	units	type	dropout		l1	l2	mean_rate	rate_rms	momentum	mean_weight	weight_rms	mean_bias	bias_rms
1	1	57	Input	0.00 %		NA	NA	NA	NA	NA	NA	NA	NA	NA
2	2	30	Rectifier	0.00 %	0.000000	0.000000		0.029427	0.032589	0.000000	-0.001651	0.209574	0.502781	0.099228
3	3	20	Rectifier	0.00 %	0.000000	0.000000		0.010699	0.020044	0.000000	-0.003087	0.230354	1.027288	0.097248
4	4	10	Rectifier	0.00 %	0.000000	0.000000		0.023632	0.078200	0.000000	0.025377	0.348013	0.997972	0.058614
5	5	2	Softmax		NA	0.000000	0.000000	0.087357	0.198167	0.000000	0.041937	1.789724	-0.002168	0.037151

```
H2OBinomialMetrics: deeplearning
** Reported on training data. **
** Metrics reported on full training frame **
```

```
MSE: 0.03031053
RMSE: 0.1740992
LogLoss: 0.1156961
Mean Per-Class Error: 0.01403774
AUC: 0.9986398
AUCPR: 0.9980325
Gini: 0.9972797
```

混同行列

Confusion Matrix (vertical: actual; across:

	nonspam	spam	Error	Rate
nonspam	1380	13	0.009332	=13/1393
spam	17	890	0.018743	=17/907
Totals	1397	903	0.013043	=30/2300

↑ 全体の誤り率は1%程度

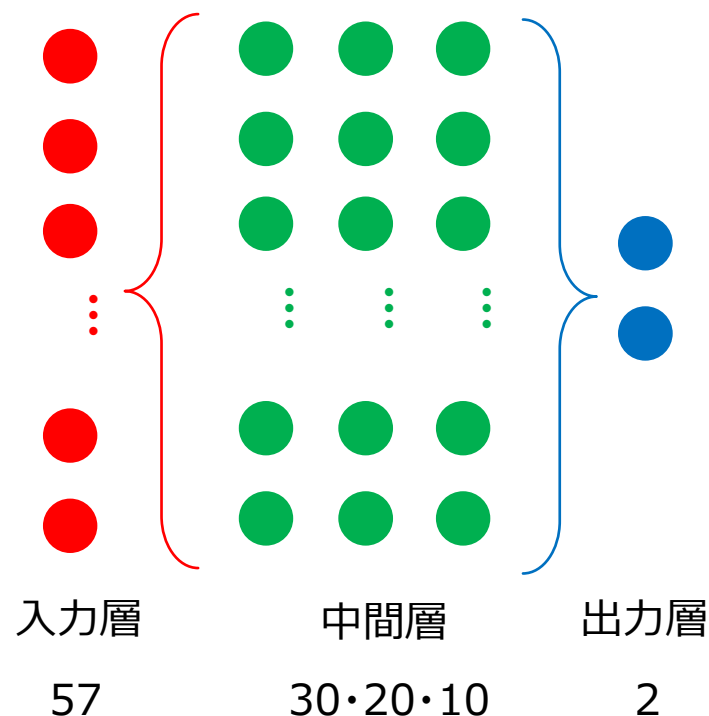
H2Oニューラルネットワークモデルによる検証データの推定結果

```
> p1<- h2o.predict(tr1,spam.test.hex)
> p10<- as.data.frame(p1)
> table(p11[,1],spam.test[,58])
```

検証データの適用

	nonspam	spam
nonspam	1315	77
spam	80	829

全体の誤り率は6.8%



```
> p10[1:30,]
```

	predict	nonspam	spam
1	spam	2.110969e-09	1.00000000
2	spam	4.748186e-06	0.99999525
3	spam	4.809083e-06	0.99999519
4	spam	5.882295e-06	0.99999412
5	spam	4.508349e-07	0.99999955
6	spam	3.522702e-05	0.99996477
7	spam	2.304439e-06	0.99999770
8	spam	7.782047e-09	0.99999999
9	spam	3.730714e-08	0.99999996
10	spam	4.625032e-12	1.00000000
11	spam	1.492746e-07	0.99999985
12	spam	3.607054e-10	1.00000000
13	spam	1.490560e-07	0.99999985
14	spam	3.081479e-03	0.99691852
15	spam	7.991176e-15	1.00000000
16	spam	3.280018e-09	1.00000000
17	spam	2.326337e-03	0.99767366
18	nonspam	9.646040e-01	0.03539603
19	spam	1.876393e-08	0.99999998
20	spam	4.321522e-07	0.99999957
21	spam	6.026348e-10	1.00000000
22	spam	4.247413e-07	0.99999958
23	spam	5.502700e-10	1.00000000
24	spam	5.726396e-12	1.00000000
25	spam	2.910199e-08	0.99999997
26	spam	1.811079e-16	1.00000000
27	spam	1.303436e-07	0.99999987
28	spam	8.979515e-05	0.99991020
29	spam	4.736505e-03	0.99526350
30	spam	3.021247e-07	0.99999970



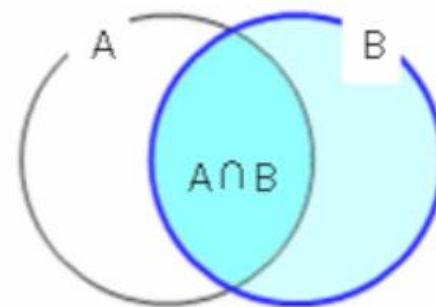
ベイズの定理

確率変数 X, Y に対する**同時確率**を**条件付き確率**に分解する

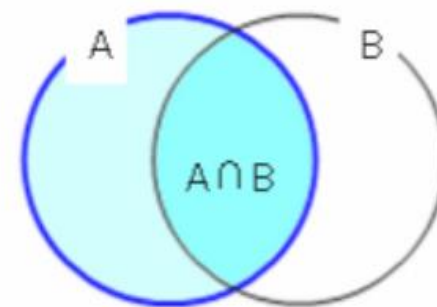
$$p(X, Y) = p(Y|X)p(X) = p(X|Y)p(Y)$$

$$\therefore p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

同時確率を 2 つの条件付き確率で表す



$$p(A|B)p(B)$$



$$p(B|A)p(A)$$

ナীবベイズとは？

ベイズの定理 を用いた判別法

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

X : 説明変数 (特徴量)

Y : 目的変数 (ラベル)

$Y = (y_1, y_2, \dots, y_n)$ で判別されたグループに対して $p(Y|X)$ が最大になるようにグループを決める！

$p(X), p(Y)$ に確率分布を仮定する必要があるが
ここでは正規分布を仮定して、条件付き確率を最大化するように判別する

$X = (x_1, x_2, \dots, x_n)$ は判別条件には無関係なので
以下の条件付き確率を最大化する

$$p(Y|X) \propto p(X|Y)p(Y)$$

H2Oパッケージのナイーブベイズを実行する

```
> tr2 <- h2o.naiveBayes(x=1:57,y=58,training_frame=spam.train.hex)
> tr2
```

Model Details:

=====

```
H2OBinomialModel: naivebayes
Model ID: NaiveBayes_model_R_1686148863918_3
Model Summary:
  number_of_response_levels min_apriori_probability max_apriori_probability
1                          2                0.39435                0.60565
```

実際のデータではhamが60.6%
spamが39.4%サンプルに含まれている

```
H2OBinomialMetrics: naivebayes
** Reported on training data. **
```

判別ラベルが2種類である

```
MSE: 0.3006072
RMSE: 0.5482766
LogLoss: 8.594954
Mean Per-Class Error: 0.1481063
AUC: 0.8733877
AUCPR: 0.7677326
Gini: 0.7467753
```

混同行列

Confusion Matrix (vertical: actual; across: predicted) for F1-optimal threshold:

	nonspam	spam	Error	Rate
nonspam	1200	193	0.138550	=193/1393
spam	143	764	0.157663	=143/907
Totals	1343	957	0.146087	=336/2300

全体の誤り率は14.6%

モデルのあてはまりでは、spamが58.4%
hamが41.6%サンプルに含まれると推定

H2Oナイーブベイズによる検証データの推定結果

```
> p2<- h2o.predict(tr2,spam.test.hex)
> p20<- as.data.frame(p2)
> table(p20[,1],spam.test[,58])
```

検証データの適用

	nonspam	spam
nonspam	1172	112
spam	223	794

全体の誤り率は14.6%



VS.



```
> p20[1:30,]
```

	predict	nonspam	spam
1	spam	7.200405e-32	1
2	spam	1.528628e-21	1
3	spam	4.572052e-28	1
4	spam	1.611076e-28	1
5	spam	1.530615e-22	1
6	spam	2.660869e-18	1
7	spam	4.698982e-42	1
8	spam	3.530072e-25	1
9	spam	4.481220e-48	1
10	spam	4.954148e-51	1
11	spam	9.132395e-28	1
12	spam	1.230302e-18	1
13	spam	9.299988e-28	1
14	spam	9.521486e-90	1
15	spam	5.871974e-34	1
16	spam	7.084098e-22	1
17	spam	1.745787e-35	1
18	spam	1.878367e-19	1
19	spam	2.615732e-37	1
20	spam	1.447472e-19	1
21	nonspam	1.943822e-15	1
22	spam	9.573678e-22	1
23	spam	2.862027e-36	1
24	spam	6.980342e-46	1
25	spam	1.153922e-27	1
26	spam	0.000000e+00	1
27	spam	6.959075e-30	1
28	spam	9.570308e-19	1
29	spam	5.635211e-18	1
30	spam	1.179703e-19	1

```
> p20[2001:2031,]
```

	predict	nonspam	spam
2001	nonspam	2.321767e-15	1.000000e+00
2002	nonspam	2.405972e-15	1.000000e+00
2003	nonspam	2.738940e-16	1.000000e+00
2004	nonspam	1.000000e+00	0.000000e+00
2005	nonspam	1.000000e+00	0.000000e+00
2006	nonspam	1.000000e+00	1.584315e-185
2007	nonspam	1.000000e+00	0.000000e+00
2008	nonspam	1.000000e+00	8.525712e-127
2009	nonspam	1.000000e+00	3.423193e-166
2010	nonspam	1.000000e+00	0.000000e+00
2011	nonspam	1.000000e+00	0.000000e+00
2012	nonspam	1.000000e+00	0.000000e+00
2013	nonspam	1.000000e+00	4.014977e-116
2014	nonspam	1.000000e+00	5.521790e-43
2015	nonspam	1.000000e+00	4.307223e-136
2016	nonspam	1.000000e+00	0.000000e+00
2017	spam	2.066472e-116	1.000000e+00
2018	nonspam	1.000000e+00	1.093180e-31
2019	nonspam	2.173595e-13	1.000000e+00
2020	nonspam	1.000000e+00	0.000000e+00
2021	nonspam	1.000000e+00	4.064152e-52
2022	nonspam	3.623950e-13	1.000000e+00
2023	spam	1.145214e-34	1.000000e+00
2024	nonspam	1.000000e+00	1.021287e-19
2025	spam	2.664849e-21	1.000000e+00
2026	nonspam	1.000000e+00	1.813191e-47
2027	nonspam	1.000000e+00	7.059446e-98
2028	nonspam	1.000000e+00	1.912750e-37
2029	nonspam	1.001353e-09	1.000000e+00
2030	nonspam	9.299400e-13	1.000000e+00
2031	spam	1.128400e-19	1.000000e+00

課題：IRISデータの非線形分析

IRISデータのサンプルから半分を学習データとしてニューラルネットワークとナイーブベイズのモデルを構築し、残り半分を検証データを用いて判別能力を評価・考察しなさい。

(参考) 線形判別の混同行列

	C	S	V
C	24	0	1
S	0	25	0
V	1	0	24

学習データ

	C	S	V
C	24	0	1
S	0	25	0
V	2	0	23

検証データ

Sepal.Length	Sepal.width	Petal.Length	Petal.width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
7.0	3.2	4.7	1.4	versicolor
6.4	3.2	4.5	1.5	versicolor
6.9	3.1	4.9	1.5	versicolor
5.5	2.3	4.0	1.3	versicolor
6.5	2.8	4.6	1.5	versicolor
5.7	2.8	4.5	1.3	versicolor
6.3	3.3	4.7	1.6	versicolor
4.9	2.4	3.3	1.0	versicolor
6.6	2.9	4.6	1.3	versicolor
5.2	2.7	3.9	1.4	versicolor
6.3	3.3	6.0	2.5	virginica
5.8	2.7	5.1	1.9	virginica
7.1	3.0	5.9	2.1	virginica
6.3	2.9	5.6	1.8	virginica
6.5	3.0	5.8	2.2	virginica
7.6	3.0	6.6	2.1	virginica
4.9	2.5	4.5	1.7	virginica
7.3	2.9	6.3	1.8	virginica
6.7	2.5	5.8	1.8	virginica
7.2	3.6	6.1	2.5	virginica





データマイニングを楽しもう！