演習

ロジスティック回帰分析

片山 翔太

ロジスティック回帰分析

- **目的**
  - 2値(0 or 1)の結果変数を説明変数から**予測**する
    - 製品を買う, 病気になる, 有害サイトの判定, 破産する, etc
  - Xが与えられた際のY=1になる確率を予測する

$$P(Y = 1|X) = 1 - P(Y = 0|X)$$

まずは単変量の場合

- **Defaultデータ**
  - カード借入金額(**balance**)と債務不履行(**default**)

```
> library(ISLR)
> head(Default)
```

	default	student	balance	income
1	No	No	729.5265	44361.625
2	No	Yes	817.1804	12106.135
3	No	No	1073.5492	31767.139
4	No	No	529.2506	35704.494
5	No	No	785.6559	38463.496
6	No	Yes	919.5885	7491.559

Y X

データの整理

パッケージ”dplyr”をインストール&読み込む

```
> library(dplyr)
> defa <- Default[,c("default","balance")]
> defa <- mutate(defa,default=as.numeric(default=="Yes"))
> head(defa)
  default   balance
1       0  729.5265
2       0  817.1804
3       0 1073.5492
4       0  529.2506
5       0  785.6559
```

結果変数をダミー変数へ

関数”mutate” :
データフレーム編集の際に便利

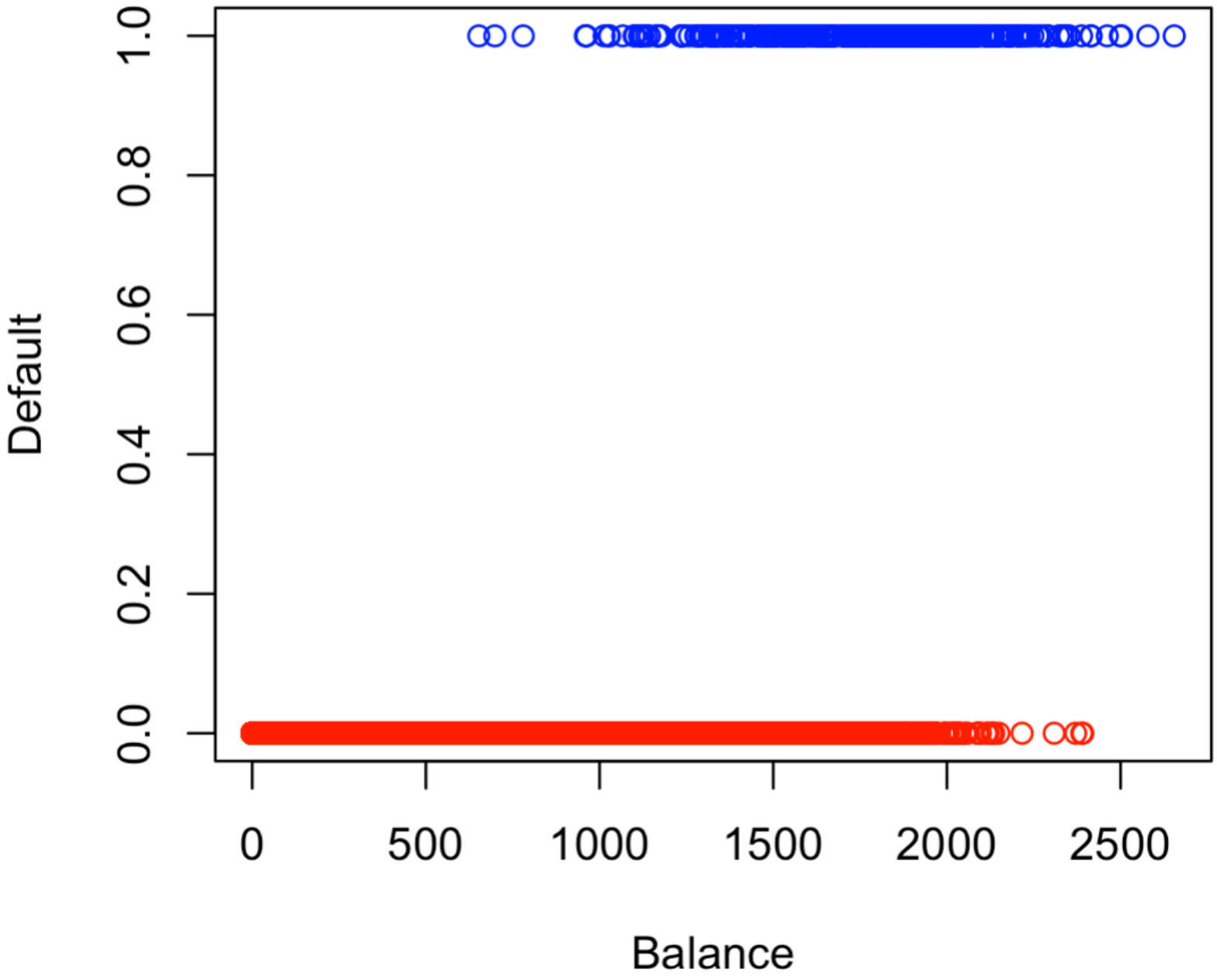
```
> defa.tr <- defa[1:9000,]
> defa.te <- defa[9001:10000,]
```

トレーニングデータ

テストデータ

・散布図

```
plot(x=NULL,y=NULL,xlim=c(0,max(defa[,2]+1)),ylim=c(0,1),xlab="Balance",ylab="Default")
points(defa[which(defa$default==1),2],defa[which(defa$default==1),1],col="blue")
points(defa[which(defa$default==0),2],defa[which(defa$default==0),1],col="red")
```



線形回帰モデルを当ててみる

$$E(Y|X) = P(Y = 1|X) = \beta_0 + \beta_1 X$$

```
> reg.defa <- lm(default~balance,data=defa.tr)
> pre <- predict(reg.defa,defa.te)
> table(defa.te[,1], as.numeric(pre > 0.5))
```

テストデータでの予測値

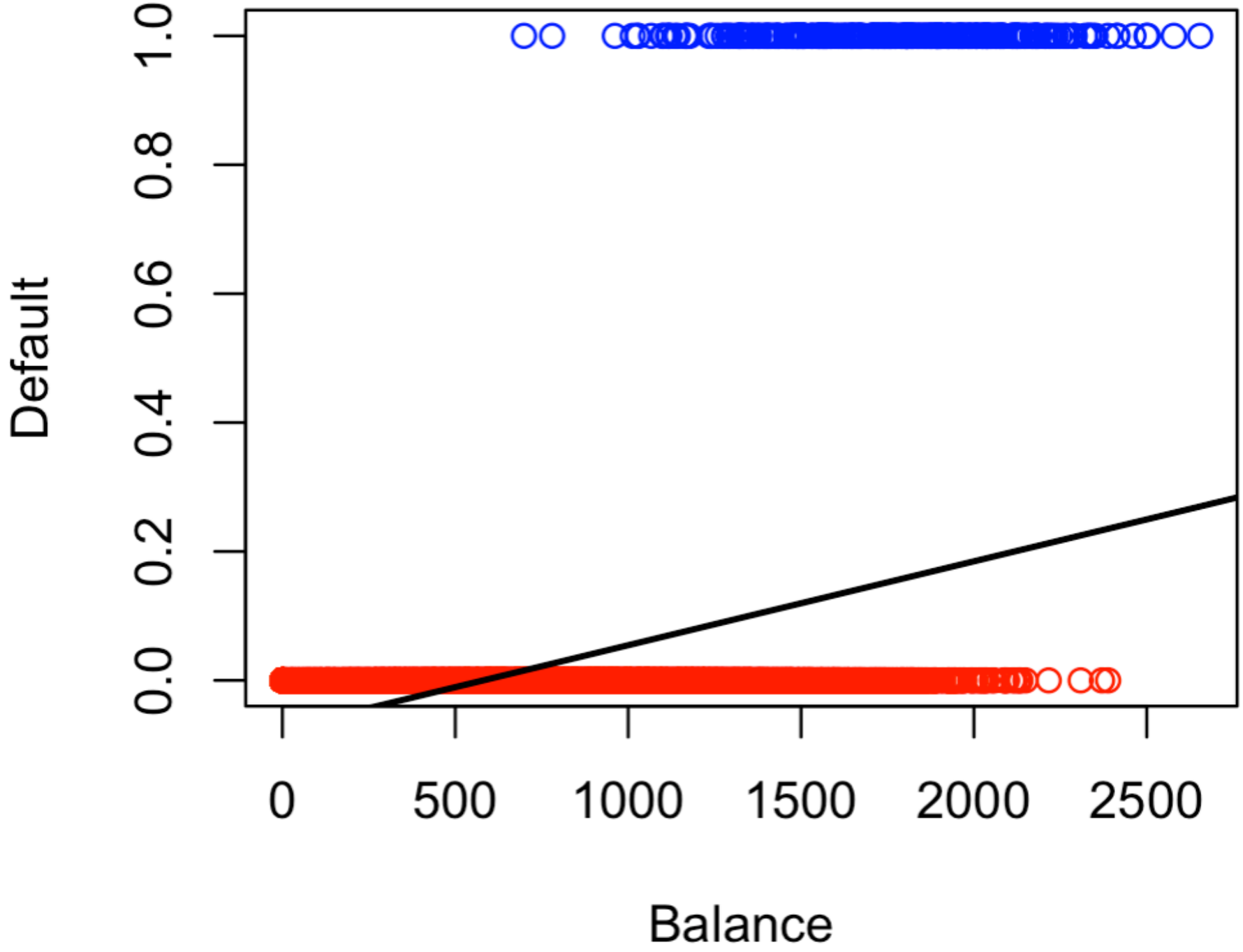
クロス集計表を表示

0.5より大きな予測値を1にする

```
   0
0 964
1  36
```

全くうまくいっていない...

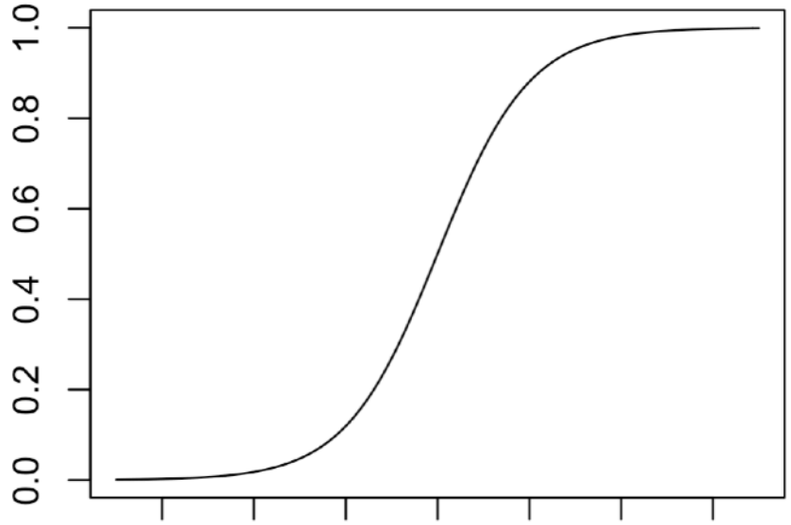
そもそも予測値が[0,1]を超える



ロジスティック回帰

- **ロジスティック関数**

$$f(\theta) = \frac{1}{1 + \exp(-\theta)}$$



この関数を使ってデータに当てはめる

- **ロジスティック回帰分析(単変量ver.)**
  - $\theta = \beta_0 + \beta_1 X$　　**上記関数の変数が線形モデル**

$$P(Y = 1|X) = \frac{1}{1 + \exp(-\beta_0 - \beta_1 X)}$$

とりあえずやってみる

```
> log.defa <- glm(default~balance,data=defa.tr,family="binomial")
> coef(log.defa)
  (Intercept)       balance
-10.817741102   0.005595708
```

$\hat{\beta}_0$ $\hat{\beta}_1$

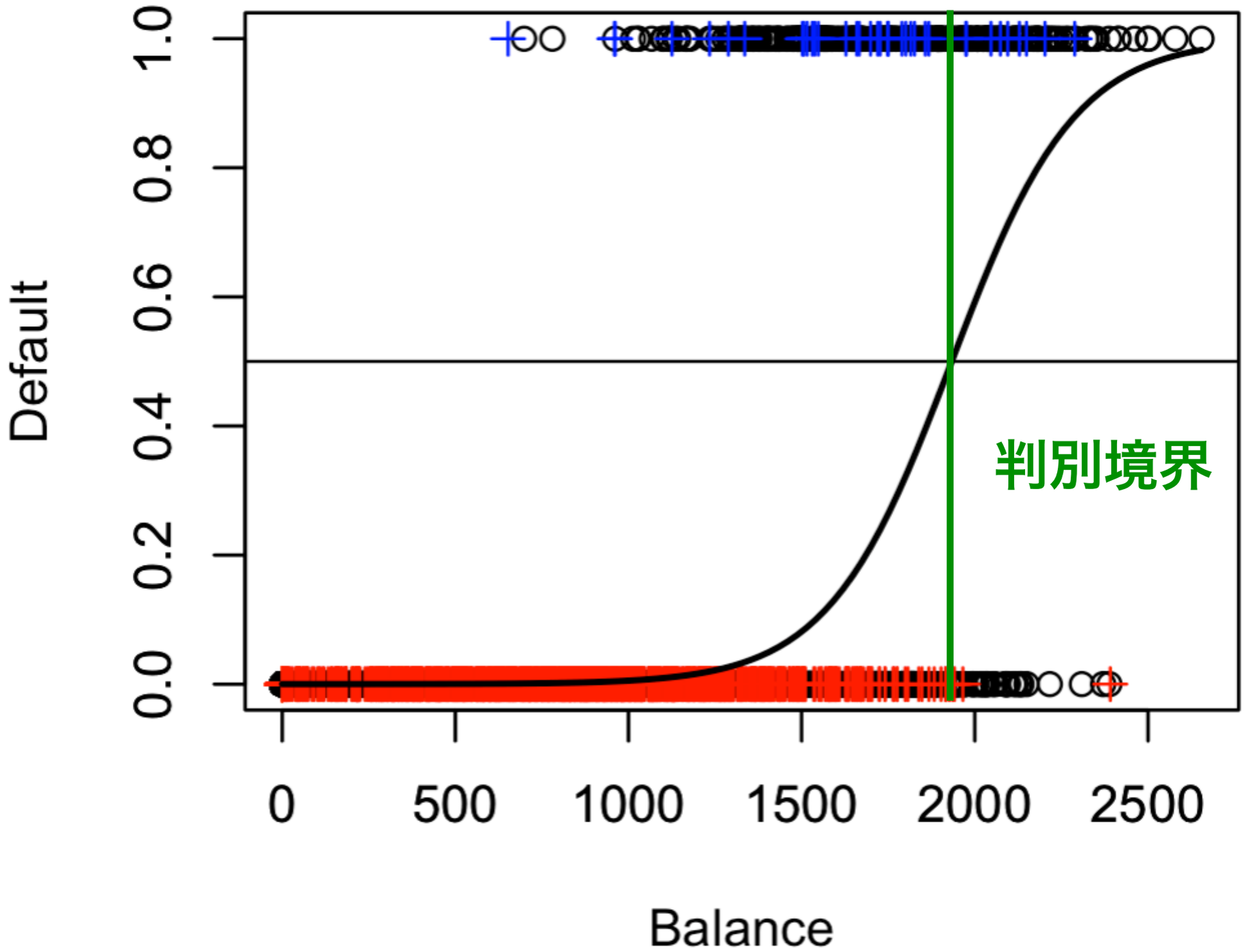
```
> pre2 <- predict(log.defa,defa.te,type="response")
> table(defa.te[,1],as.numeric(pre2 > 0.5))
```

	0	1
0	960	4
1	27	9

債務不履行**する**ことは当たらない

しないことは当たるが意味がない...

色付き = テストデータ


Default
1.0
0.8
0.6
0.4
0.2
0.0
0
500
1000
1500
2000
2500
Balance
判別境界

クロス集計表から計算される性能指標

	予測1	予測0
真に1	True Positive **TP**	False Negative **FN**
真に0	False Positie **FP**	True Negative **TN**

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FN} + \text{FP} + \text{TN}} \qquad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}$$

Defaultデータの場合

	予測1	予測0
真1	9	27
真0	4	960

$$\text{Accuracy} = \frac{966}{1000} = 0.966, \quad \text{Precision} = \frac{9}{9+4} = 0.692$$

$$\text{Recall} = \frac{9}{9+27} = 0.25, \quad \text{FPR} = \frac{4}{4+960} = 0.004$$

- AccuracyとFPRは**あまり信頼できない** (アンバランス)
- Recallが低い ➡ 取りこぼしが多い (効果が薄い)
- Precisionが低い ➡ 介入した際のリスクが高い

パラメータの推定

- **最尤法の復習**
  - サンプル(標本)は確率最大のものが出現したという仮定
- **ベルヌーイ分布の例**

$$X = \begin{cases} 1, & \text{with probability } p \\ 0, & \text{with probability } 1-p \end{cases}$$

独立に得られた10個のデータ

$$x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1, x_5 = 1$$
$$x_6 = 1, x_7 = 0, x_8 = 1, x_9 = 1, x_{10} = 1$$

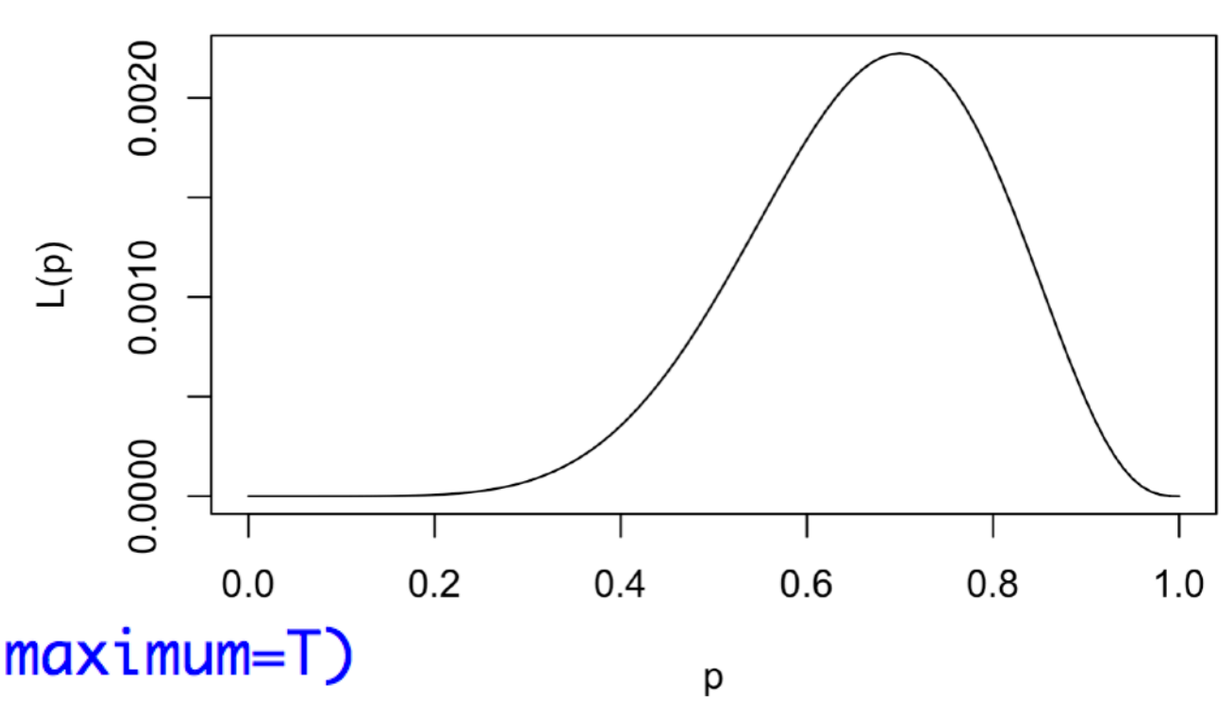
このデータが得られる確率を最大にするpを推定！

サンプルが得られる確率

$$L(p) := p^7(1-p)^3 = \prod_{i=1}^{10} p^{x_i}(1-p)^{1-x_i}$$

L(p)のプロットを描く

```
> lp <- function(p){(p^7)*(1-p)^3}
> curve(lp,xlim=c(0,1),xlab="p",ylab="L(p)")
```



最大値の計算

- for文を使って貪欲に行う
- optimize関数を使う

```
> optimize(lp,interval=c(0,1),maximum=T)
$maximum
[1] 0.6999843
```

注意：optimizeは1変数最適化しかできない

ロジスティック回帰における推定

$$P(Y=1|X)=\frac{1}{1+\exp(-\beta_0-\beta_1 X)}\ (=p_X(\boldsymbol{\beta}))$$

$$\boldsymbol{\beta}=(\beta_0,\beta_1)$$

独立に得られたn個のデータ：$(y_1,x_1),(y_2,x_2),(y_3,x_3),\ldots,(y_n,x_n)$

上記データが得られる確率

$$L(\boldsymbol{\beta})=\prod_{i=1}^{n}\{p_{x_i}(\boldsymbol{\beta})\}^{y_i}\{1-p_{x_i}(\boldsymbol{\beta})\}^{1-y_i}$$

対数logを取る

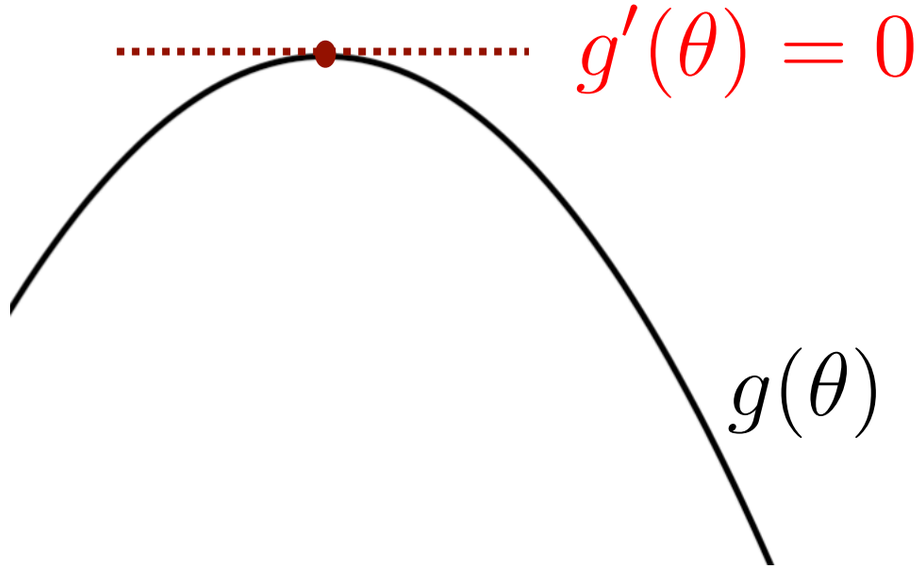
積記号を和記号へ

$$\log L(\boldsymbol{\beta})=\sum_{i=1}^{n}\left\{y_i\log p_{x_i}(\boldsymbol{\beta})+(1-y_i)\log p_{x_i}(\boldsymbol{\beta})\right\}$$

これを最大化！

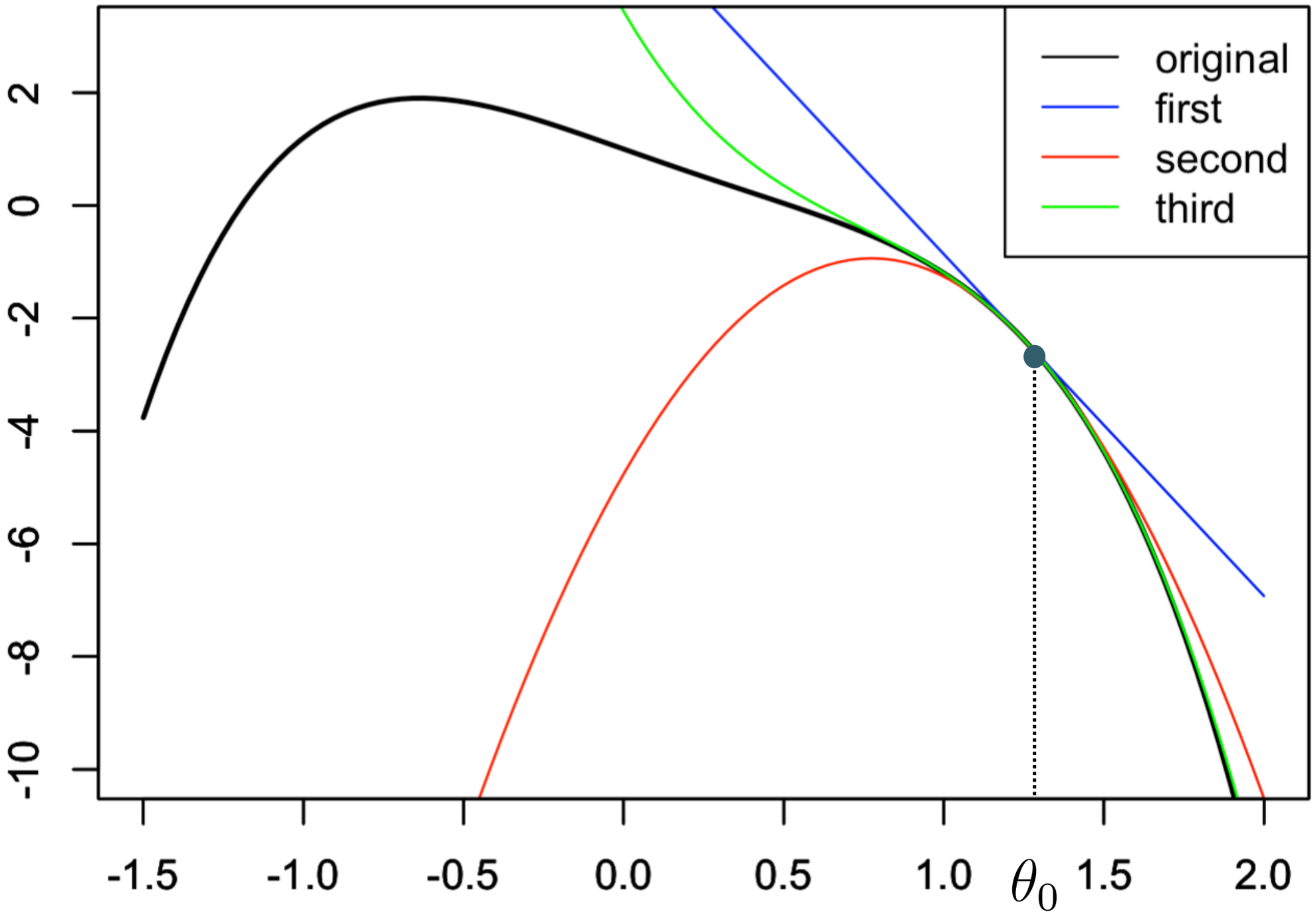
最適化入門(1変数)

- **ニュートン法 (統計分野だとスコア法)**
  - 極致 $g'(\theta) = 0$ を求める方法
  - 適当な初期値 θ_0 からスタート
  - 更新していく $\theta_1, \theta_2, \theta_3, \ldots$



- **ニュートン法の仕組み**
  - 基本はテーラー展開
  - 既知のg(θ0)の値を使って適当なg(θ)を近似できる

$$g(\theta) = g(\theta_0) + g'(\theta_0)(\theta - \theta_0) + \frac{1}{2}g''(\theta_0)(\theta - \theta_0)^2 + \cdots$$


original
first
second
third
2
0
-2
-4
-6
-8
-10
-1.5
-1.0
-0.5
0.0
0.5
1.0
θ_0
1.5
2.0

$G(\theta) = g'(\theta)$ と思って1次項までテーラー展開

$$g'(\theta) \simeq g'(\theta_0) + g''(\theta_0)(\theta - \theta_0)$$

左辺が0になるようなθ (=θ_1)を求める

$$\theta_1 = \theta_0 - \frac{g'(\theta_0)}{g''(\theta_0)}$$

得られたθ_1を使って同様に

$$\theta_2 = \theta_1 - \frac{g'(\theta_1)}{g''(\theta_1)}$$

以上を収束(値が変わらなくなるまで)繰り返す！

- **2変数以上の場合**
  - 基本的に同様 (テーラー展開の式が多少変わる)
  - 詳細は省略 (詳しい話は微分積分学の教科書等を参照のこと)

$$\log L(\boldsymbol{\beta}) = \sum_{i=1}^{n} \left\{ y_i \log p_{x_i}(\boldsymbol{\beta}) + (1 - y_i) \log p_{x_i}(\boldsymbol{\beta}) \right\}$$

ロジスティック回帰における更新式

$$\boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 - (\boldsymbol{X}^T \boldsymbol{W}(\boldsymbol{\beta}_0) \boldsymbol{X})^{-1} \boldsymbol{X}^T (\boldsymbol{p}(\boldsymbol{\beta}_0) - \boldsymbol{y})$$

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad \boldsymbol{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \quad \boldsymbol{p}(\boldsymbol{\beta}) = \begin{pmatrix} p_1(\boldsymbol{\beta}) \\ p_2(\boldsymbol{\beta}) \\ \vdots \\ p_n(\boldsymbol{\beta}) \end{pmatrix} \quad p_i(\boldsymbol{\beta}) = p_{x_i}(\boldsymbol{\beta})$$

$$\boldsymbol{W}(\boldsymbol{\beta}) = \underline{\mathrm{diag}}\{p_1(\boldsymbol{\beta})(1 - p_1(\boldsymbol{\beta})), \cdots, p_n(\boldsymbol{\beta})(1 - p_n(\boldsymbol{\beta}))\}$$

対角行列

- **Exercise**
  - 必要箇所を埋めてロジスティック回帰におけるニュートン法を完成させよ

```
ff <- function(theta){1/(1+exp(-theta))}
x <- defa.tr[,2]; X <- cbind(1,x); y <- defa.tr[,1]
beta <- c(0,0); judge <- 1000

for(k in 1:100){      #高々何回繰り返すかを指定しておく
  if(judge >= 0.0001){
    old <- beta
    p <- ff(X%*%old)          #diag部分の行列サイズが大きい
    #H <- solve(t(X) %*% diag(c(p*(1-p))) %*% X)
    H <-                                  #行列サイズを小さく
    beta <- old - H %*% t(X) %*% (p-y)
    judge <- sum(abs(beta-old))
  } else break      #100回更新 or 収束すれば終了
}
beta
```

多変数の場合のロジスティック回帰

・ロジスティック回帰(多変数ver.)

- $\theta = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p$

$$P(Y = 1|X) = \frac{1}{1 + \exp(-\beta_0 - \beta_1 X_1 - \cdots - \beta_p X_p)}$$

・スコア法の更新式

・1変数の場合とほとんど同じ．説明変数行列が少し異なる

$$\boldsymbol{\beta}_1 = \boldsymbol{\beta}_0 - (\boldsymbol{X}^T \boldsymbol{W}(\boldsymbol{\beta}_0)\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{p}(\boldsymbol{\beta}_0) - \boldsymbol{y})$$

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \Rightarrow \boldsymbol{X} = \begin{pmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

Spotifyデータの分析

- **分析の目標**
  - トレーニングデータ(disneylist_train.csv)を使って曲を**推薦**
  - テストデータ(disneylist_test.csv)で推薦精度を検証

- **データの概要**
  - 曲の情報
    - danceability, energy, key, tempo, etc
  - 評価値 (5段階)
    - 1 : 嫌い, 2 : 興味なし, 3 : 普通, 4 : まあまあ, 5 : 好き
  - 評価値1~4を0, 評価値5を1としたロジスティック回帰を実行

Rで実装

- **データの整理**

#トレーニングデータの読み込み

```
> dtrain <- read.csv("disneylist_train.csv", header=T)
> title.train <- dtrain$track.name   #曲名を取っておく
> dtrain <- dtrain[,-c(1,17,18,19)] #不要な変数を削る
> dtrain <- mutate(dtrain,eval=as.numeric(eval>=5))
```

#evalが5以上を1, それ以外を0に変更

#テストデータに関しても同様

```
> dtest <- read.csv("disneylist_test.csv", header=T)
> title.test <- dtest$track.name
> dtest <- dtest[,-c(1,17,18,19)]
> dtest <- mutate(dtest,eval=as.numeric(eval>=5))
```

変数の詳細 (1)

danceability	踊りやすさ
energy	活発性や激しさ. デスメタルだと高い
key	全体的なKey. Pitch classに対応
loudness	デシベル(dB)の平均
mode	Major(1) or Minor(0)
speechiness	しゃべってるかどうか. オーティオブック等であれば1に近い
acousticness	アコースティックさ

変数の詳細 (2)

instrumentalness	ボーカルの少なさ
liveness	ライブさを表す指標
valence	曲の雰囲気. ポジティブ(高)とネガティブ(低)
tempo	曲のテンポ
duration_ms	曲の長さ(ms)
time_signature	ビートの数？
popularity	全体に占める人気具合

Rでロジスティック回帰

```
> result <- glm(eval~danceability+energy+key,data=dtrain,family="binomial")
> summary(result)
```

#glm関数を使う(familyをbinomialに指定)

$$\theta = \beta_0 + \beta_1\,\mathtt{danceability} + \beta_2\,\mathtt{energy} + \beta_3\,\mathtt{key}$$

Coefficients: $\hat{\beta}_i$

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-1.16022	0.52020	-2.230	0.0257	*
danceability	-1.58151	1.03729	-1.525	0.1273	
energy	1.57600	0.78673	2.003	0.0452	*
key	-0.03577	0.04054	-0.882	0.3776	

$$\frac{\mathtt{Estimate} - \mathtt{True}}{\mathtt{Std.Error}} \sim N(0,1)$$

#全変数を使いたい場合

```
> result <- glm(eval~.,data=dtrain,family="binomial")
```

・曲の推薦

```
> pre2 <- predict(result,dtest,type="response")
> title.test[which(pre2 > 0.5)]
[1] Pixar Playtime Pals - Tokyo DisneySea 2018
```

・クロス集計表を表示

```
> table(dtest[,1],as.numeric(pre2 > 0.5))

    0  1
0  23  1
1  12  0
```

#好きな曲が全く当たっていない…

- **Exercise**
  - 推薦精度を少しでもあげてみよう
  - 今回は特例的にテストデータを当てにいく

推薦精度を上げるために(1)

- **ダウンサンプリング (downsampling)**
  - 今回のデータは正例と負例の比率がアンバランス
    - どちらかが多いと片方に引っ張られる
  - 比率が合うようにトレーニングデータを変更する

```
> sample(c(1,2,3,4,5,6,7,8,9,10),5)
[1]  1 10  3  4  7
```

#ベクトル1~10から5個ランダムに取り出す関数

#ランダムなので結果は人によって異なる

再現性を気にする場合は**乱数の種**を指定する

```
> set.seed(12); sample(c(1,2,3,4,5,6,7,8,9,10),5)
[1] 1 8 9 2 7
> set.seed(15); sample(c(1,2,3,4,5,6,7,8,9,10),5)
[1] 7 2 8 5 3
```

#seedの数を変えれば結果が異なる

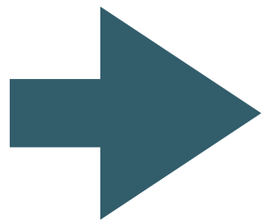
推定精度を上げるために (2)

- **変数の削除と追加**
  - 必要なさそうな変数を削除
    - p値が小さい or センス
  - 新しい変数を追加
    - 変数間の和や積を取る

```
> dtrain2 <- mutate(dtrain,newvar1=speechiness*liveness)
```

- **変数のダミー化**
  - カテゴリ変数を分離して複数の列へ

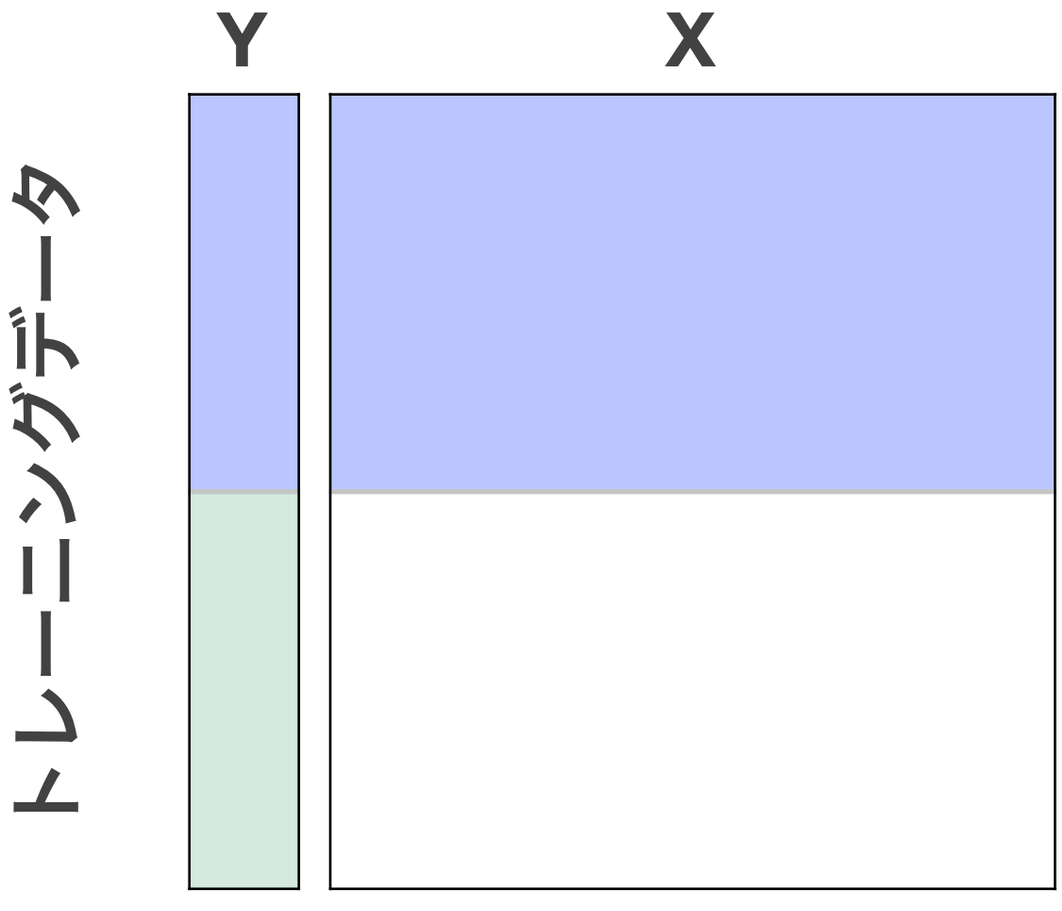
サンプル	得意科目
Aさん	国語
Bさん	算数



サンプル	国語	算数
Aさん	1	0
Bさん	0	1

- **スタッキング (Stacking)**
  - 予測でメタ特徴量を作成する

モデルを用意 ex.) $\theta = \beta_0 + \beta_1 \, \texttt{tempo}$



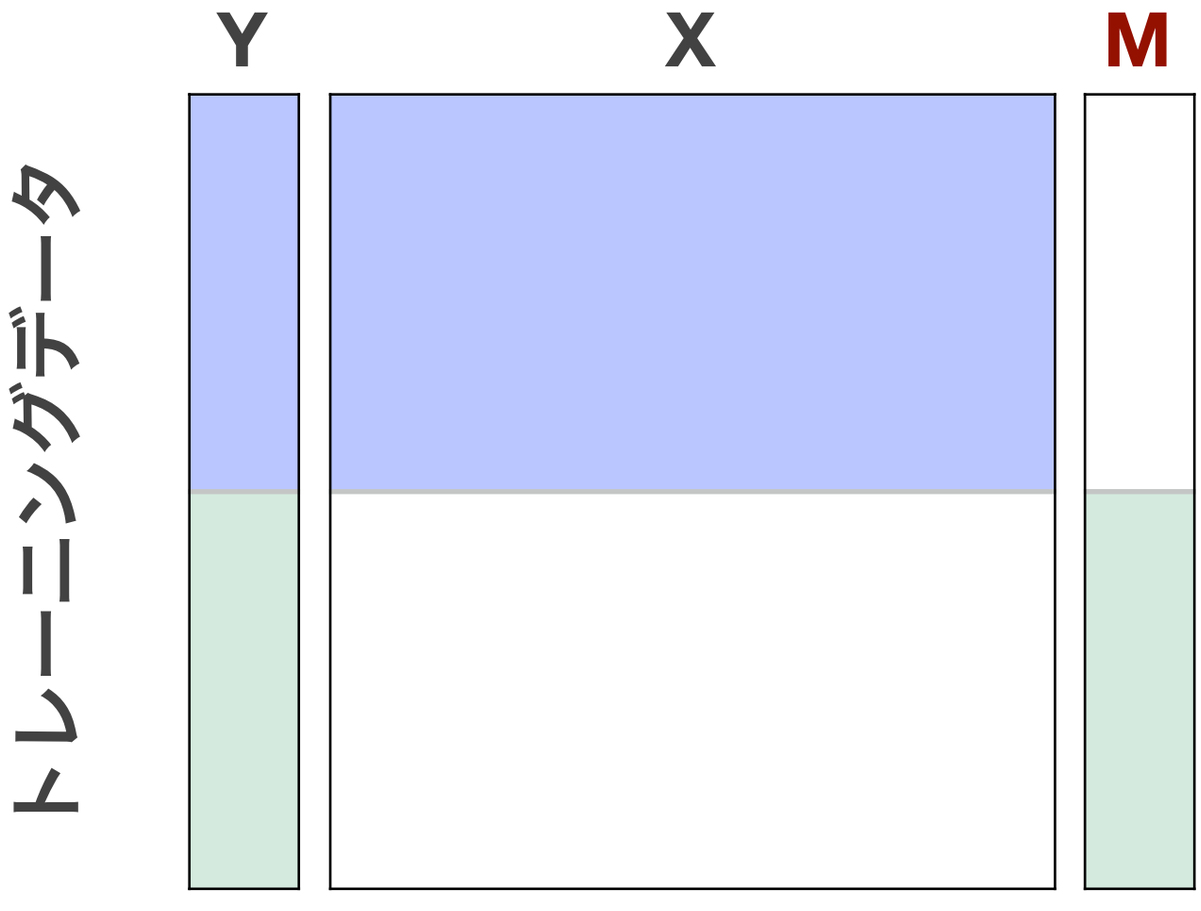
Step1

青い部分でモデルを学習

緑の部分を予測 (確率値でも可)

- **スタッキング (Stacking)**
  - 予測でメタ特徴量を作成する

モデルを用意 ex.) $\theta = \beta_0 + \beta_1 \, \mathtt{tempo}$



Step1

青い部分でモデルを学習

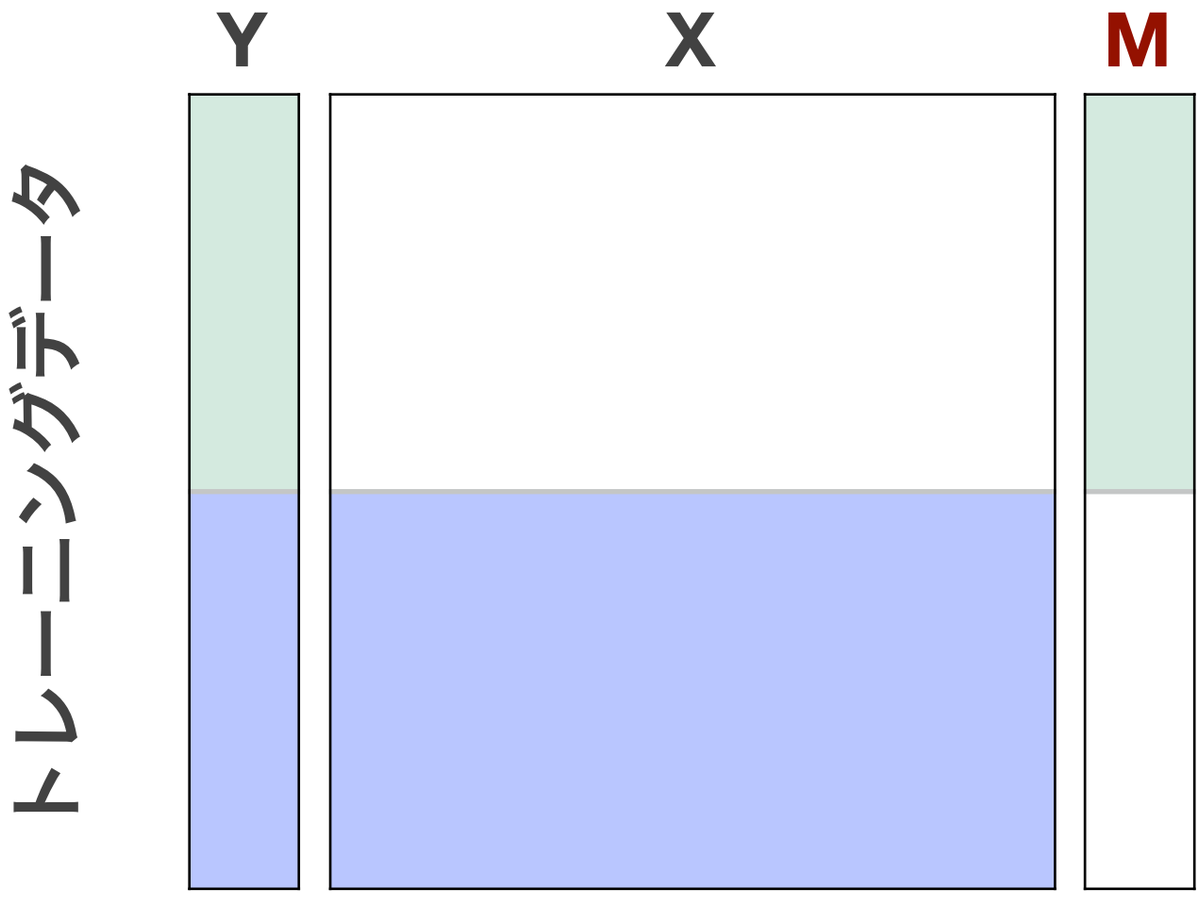
緑の部分を予測 (確率値でも可)

Step2

緑の部分の予測値を特徴量Mへ

- **スタッキング (Stacking)**
  - 予測でメタ特徴量を作成する

モデルを用意 ex.) $\theta = \beta_0 + \beta_1 \, \texttt{tempo}$



Step1

青い部分でモデルを学習

緑の部分を予測 (確率値でも可)

Step2

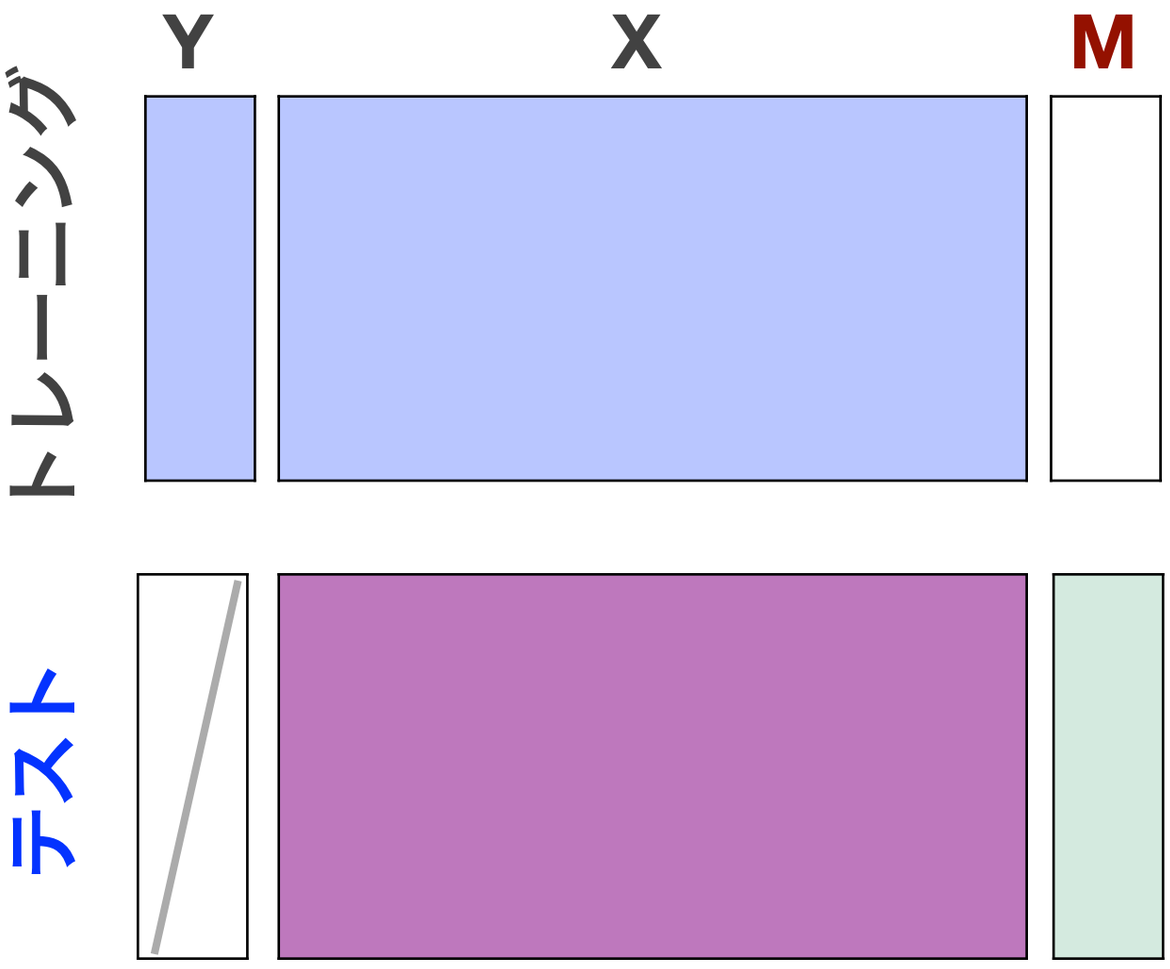
緑の部分の予測値を特徴量Mへ

Step3

役割を逆転させる

- **スタッキング (Stacking)**
  - 予測でメタ特徴量を作成する

モデルを用意 ex.) $\theta = \beta_0 + \beta_1 \, \texttt{tempo}$



Step4

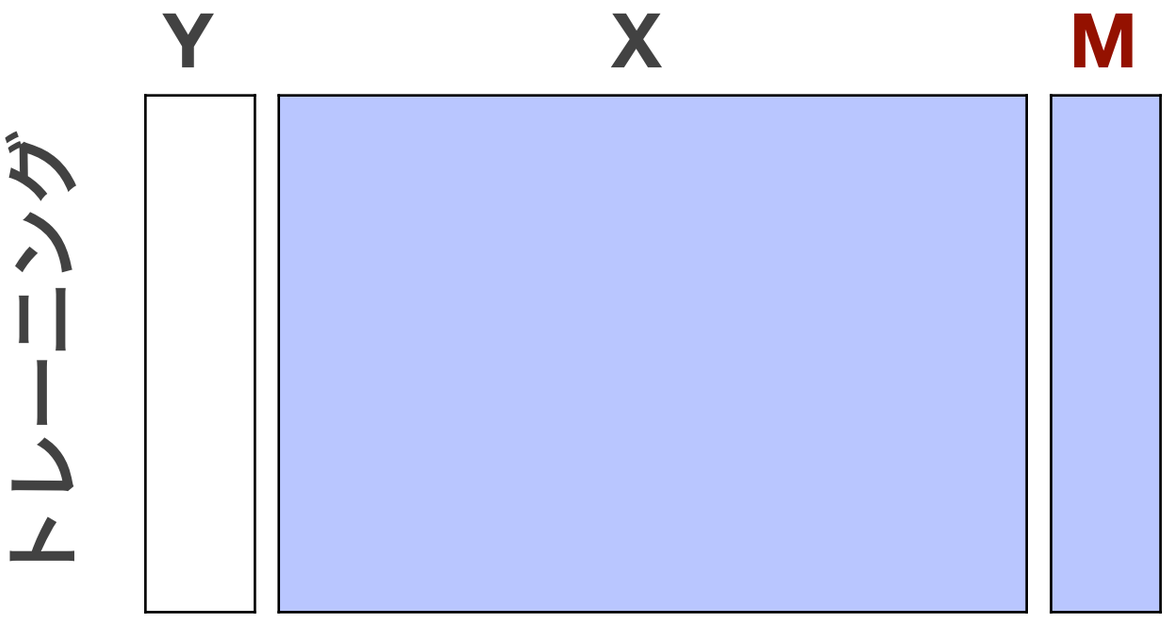
トレーニングデータを
分割せずにモデルを学習

Step5

学習したモデルをテストの
説明変数に適用し，テスト用
のメタ特徴量を作成

- **スタッキング (Stacking)**
  - 予測でメタ特徴量を作成する

モデルを用意 ex.) $\theta = \beta_0 + \beta_1 \, \mathtt{tempo}$



Step6

説明変数を(X,M)として
新しいモデルを考察・学習

注意

分割数は2でなくても良いし，メタ特徴量は複数作成しても良い

メタ特徴量を作成する際のモデルは単純なものでOK

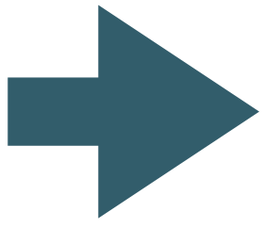
推定精度を上げるために (3)

- **アンサンブル学習**
  - いくつかのモデルを合わせてひとつのモデルへ合併

```
> m1 <- glm(eval~danceability,data=dtrain,family="binomial")
> m2 <- glm(eval~energy,data=dtrain,family="binomial")
>
> p1 <- predict(m1,dtest,type="response")
> p2 <- predict(m1,dtest,type="response")
>
> final.pre <- (p1 + p2)/2
```

$$\theta = \beta_0 + \beta_1 \texttt{danceability}$$

$$\theta = \beta_0 + \beta_1 \texttt{energy}$$



最終判断

#合併の方法は様々

#0,1で予測して多数決でもOK

- **候補モデルの作り方**
  - ダウンサンプリングを複数回実行する
    - 同じモデルでも係数の値がそれぞれ異なる
  - 説明変数を選択する
    - フルモデルで推定した後, p値の小さい説明変数を選択
    - ランダムサンプリングしてもOK
- ロジスティック回帰以外の方法で予測する(**future**)