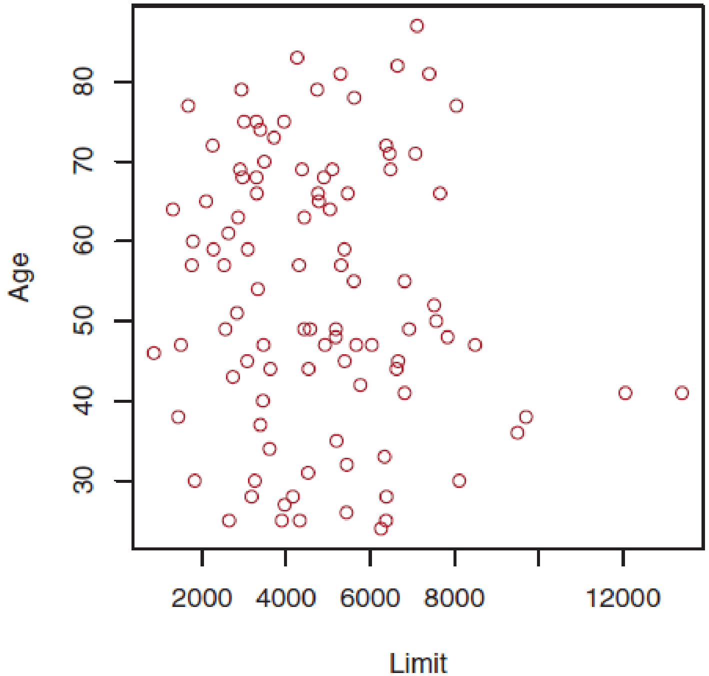
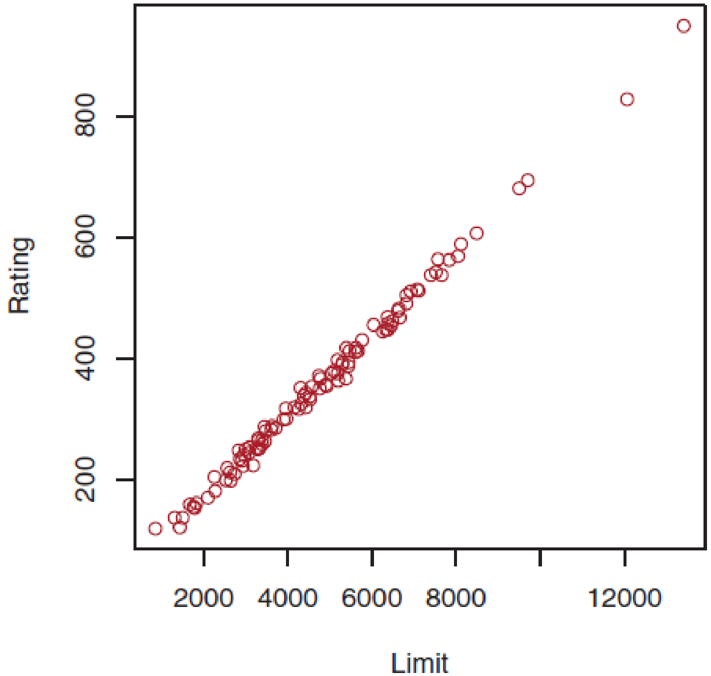
重回帰における注意点

- 共線性（collinearity）
  - 予測変数の間に強い関連がある場合

Credit データセット



予測変数 Limit と Age の関係

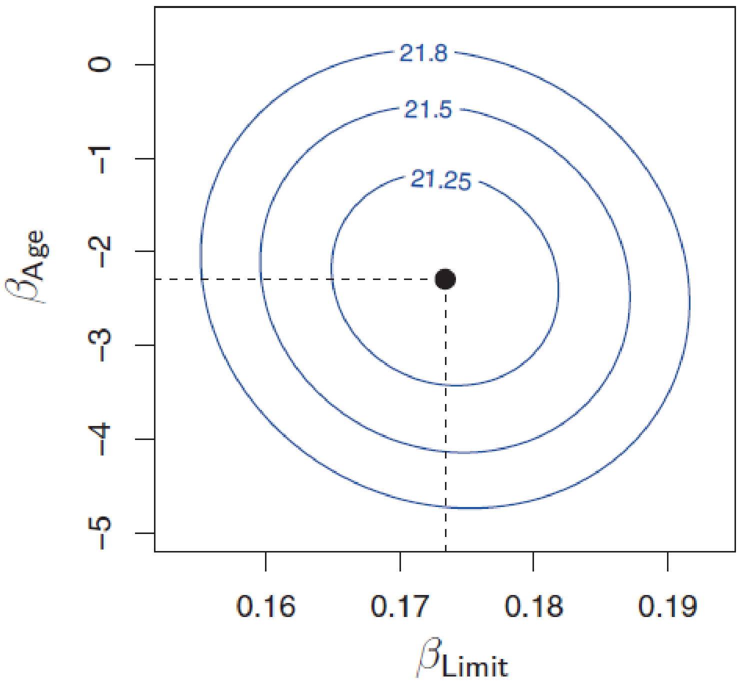


予測変数 Limit と Rating の関係

重回帰における注意点

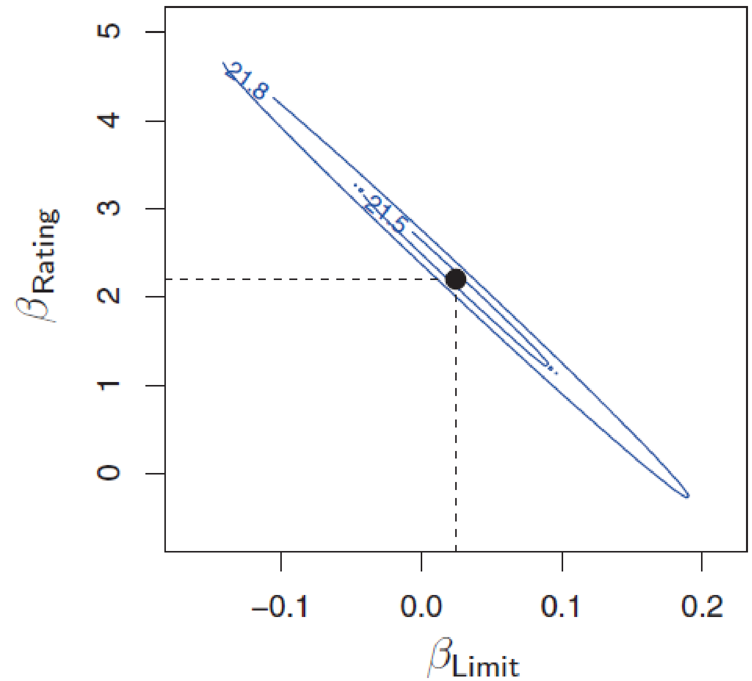
- 共線性（collinearity）
  - パラメータ推定の信頼性が落ちる

balance を limit と age で
予測するモデルのRSS



→ β_{Limit} は 0.15～0.20

balance を limit と rating で
予測するモデルのRSS



→ β_{Limit} は -0.2～0.2

重回帰における注意点

- 共線性（collinearity）
  - パラメータの信頼性

balance を limit と age で予測するモデル

	Coefficient	Std. error	t-static	p-value
Intercept	−173.411	43.828	−3.957	< 0.0001
age	−2.292	0.672	−3.407	0.0007
limit	0.173	0.005	34.496	< 0.0001

balance を limit と rating で予測するモデル

	Coefficient	Std. error	t-static	p-value
Intercept	−377.537	45.254	−8.343	< 0.0001
rating	2.202	0.952	2.312	0.0213
limit	0.025	0.064	0.384	0.7012

↑大きなp値

重回帰における注意点

- 共線性（collinearity）
  - 共線性の検出方法
    - ２つの予測変数の間の関係は相関行列で検出できる

- 多重共線性（multicollinearity）
  - ３つ以上の変数の間に共線性がある場合
  - 検出方法
    - VIF (variance inflation factor) が 5～10 を超える場合は注意

$$\mathrm{VIF}\left(\hat{\beta}_j\right)=\frac{1}{1-R^2_{X_j|X_{-j}}}$$

X_j を X_j 以外の変数で予測した場合の決定係数

```
> library(car)
> vif(lm.fit)
   crim     zn  indus     chas      nox       rm    age
   1.79   2.30   3.99     1.07     4.39     1.93   3.10
    dis    rad    tax  ptratio    black    lstat
   3.96   7.48   9.01     1.80     1.35     2.94
```

重回帰における注意点

- 共線性の問題の解決方法
  - 冗長な予測変数を除く

balance を age と limit と rating で予測するモデル（$R^2 = 0.754$）

	VIF
age	1.01
rating	160.67
limit	160.59

balance を age と limit で予測するモデル（$R^2 = 0.75$）

	VIF
age	～1
limit	～1

  - ２つの予測変数をひとつに統合
    - 標準化して平均する、など

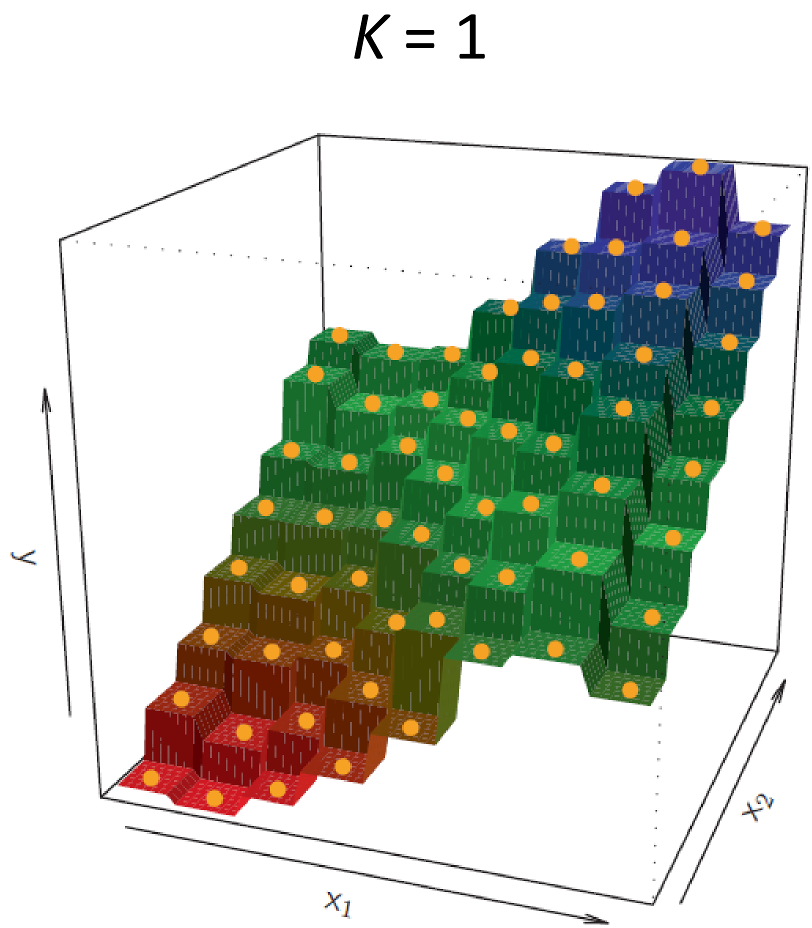
線形回帰とK最近傍回帰

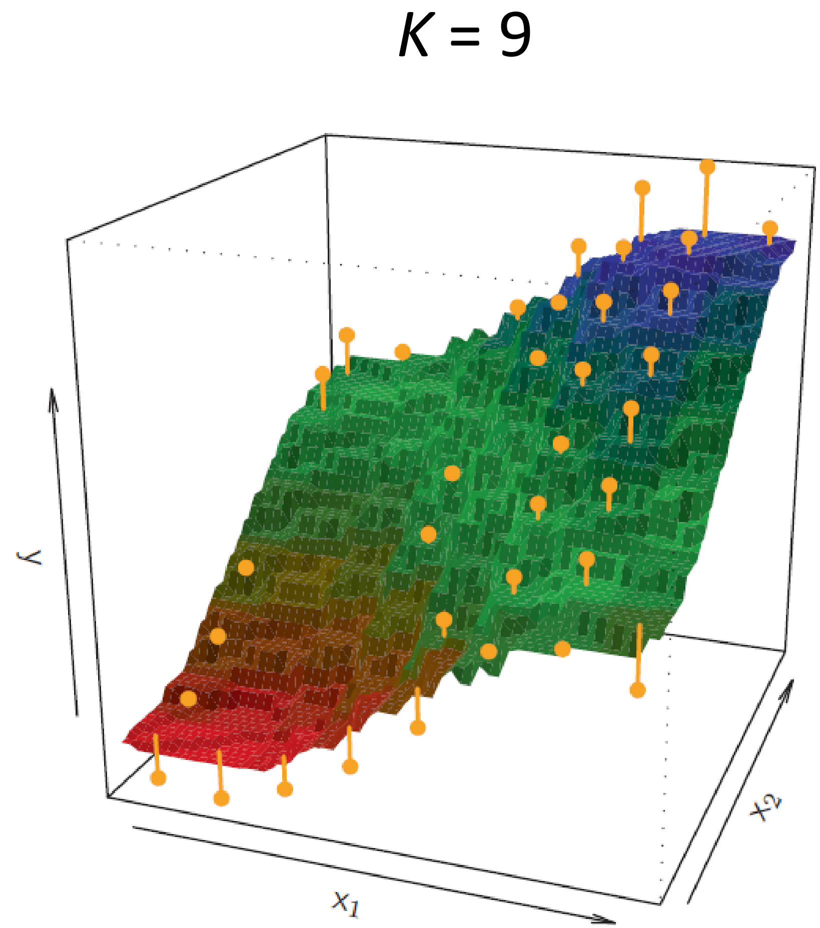
- K最近傍回帰（K-nearest neighbors regression, KNN regression）
  - 最も簡単なノンパラメトリック回帰手法のひとつ

$$\hat{f}(x_0) = \frac{1}{K} \sum_{x_i \in \mathrm{N}_0} y_i$$

$\mathrm{N}_0 : x_0$ に最も近いK個の学習データ

線形回帰とK最近傍回帰



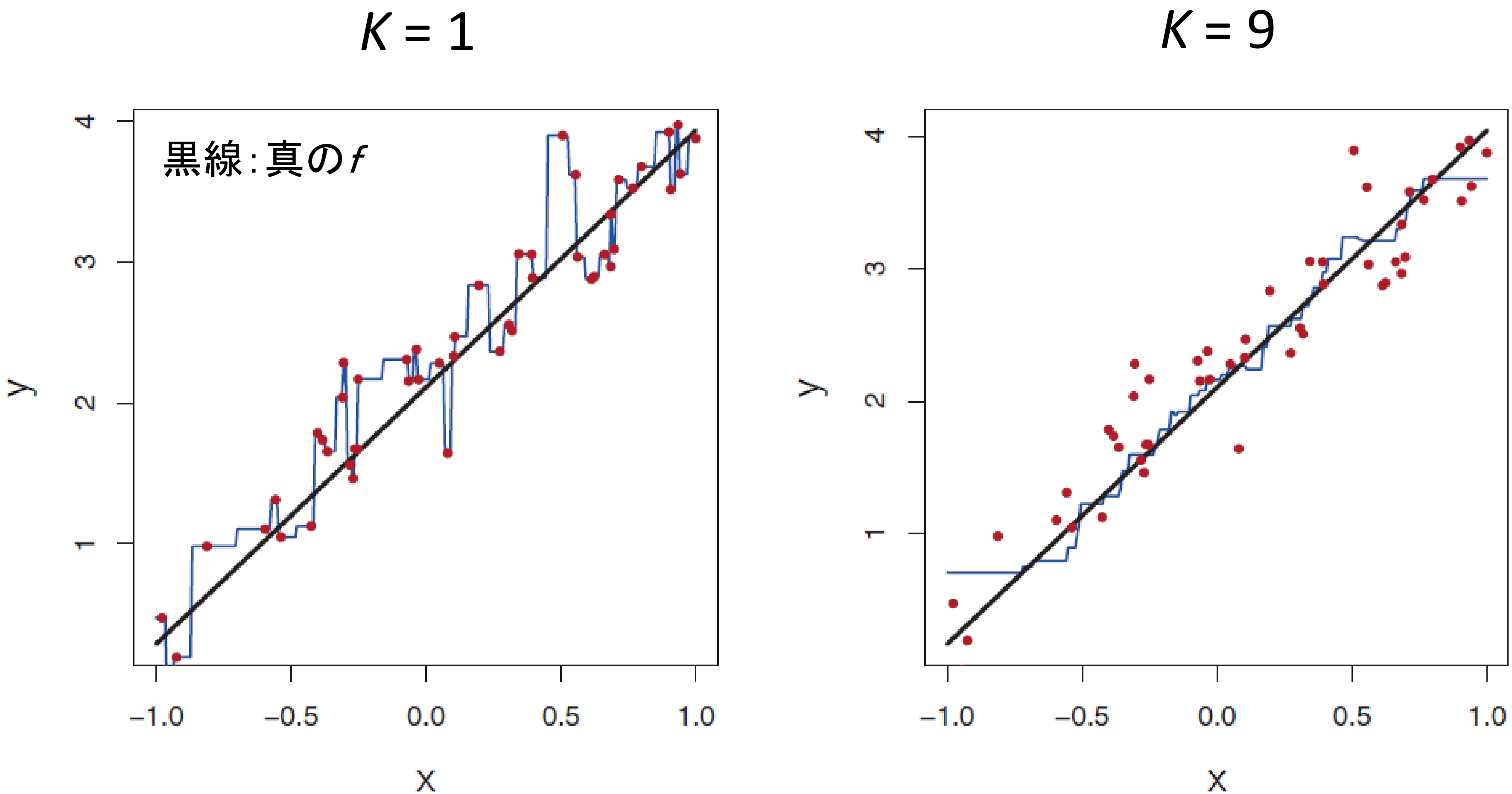


線形回帰とK最近傍回帰

- K最近傍回帰
  - K が小さい場合： バイアス小、バリアンス大
  - K が大きい場合： バイアス大、バリアンス小
  - 適切な K の値を選ぶ必要あり

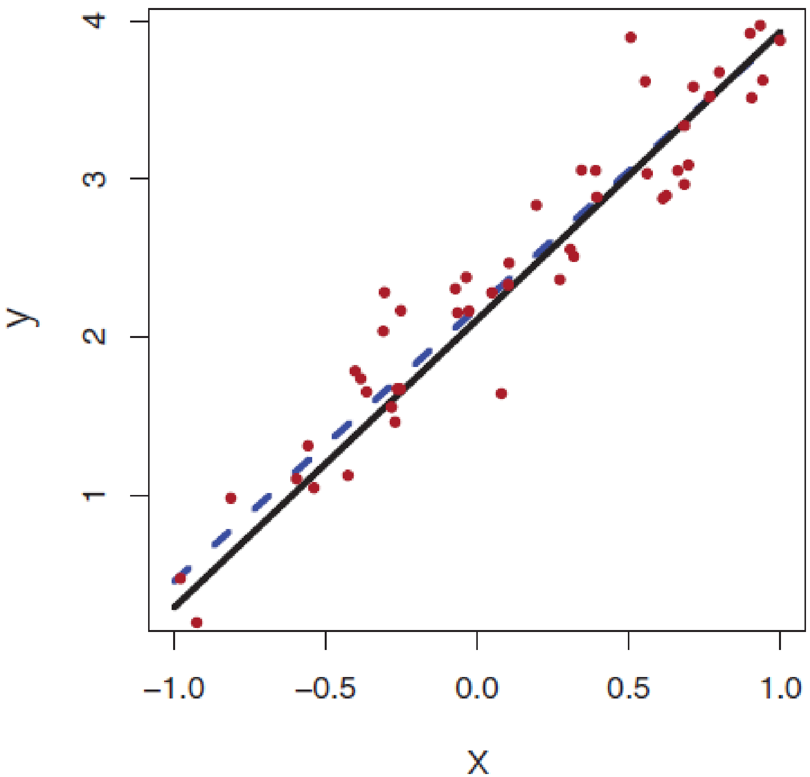
- パラメトリック vs ノンパラメトリック
  - パラメトリックモデルが真の f を適切に表現できるのであればパラメトリックな手法が有利

線形回帰とK最近傍回帰

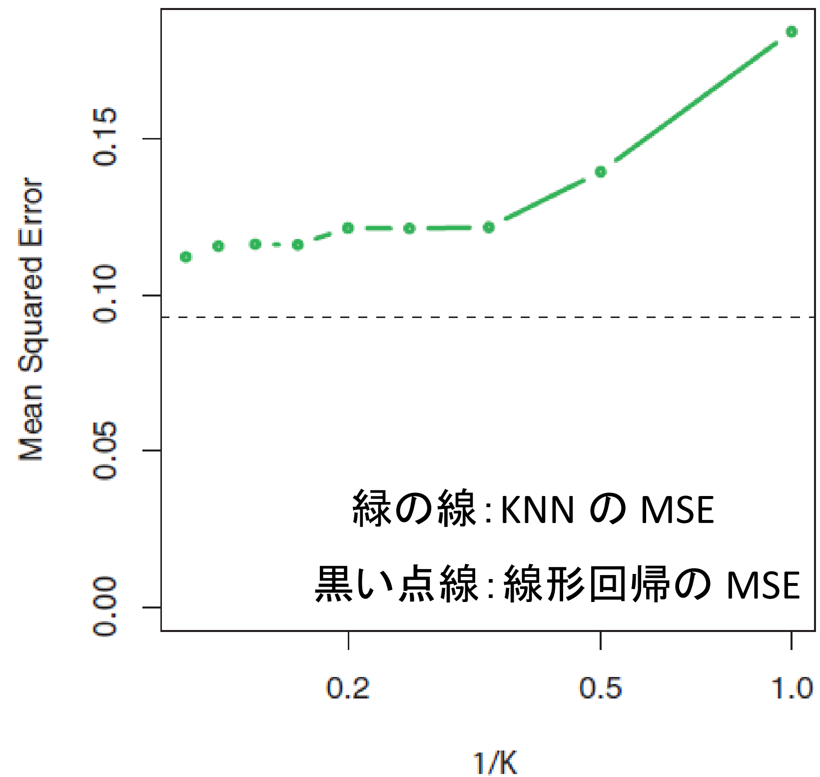


線形回帰とK最近傍回帰

線形回帰



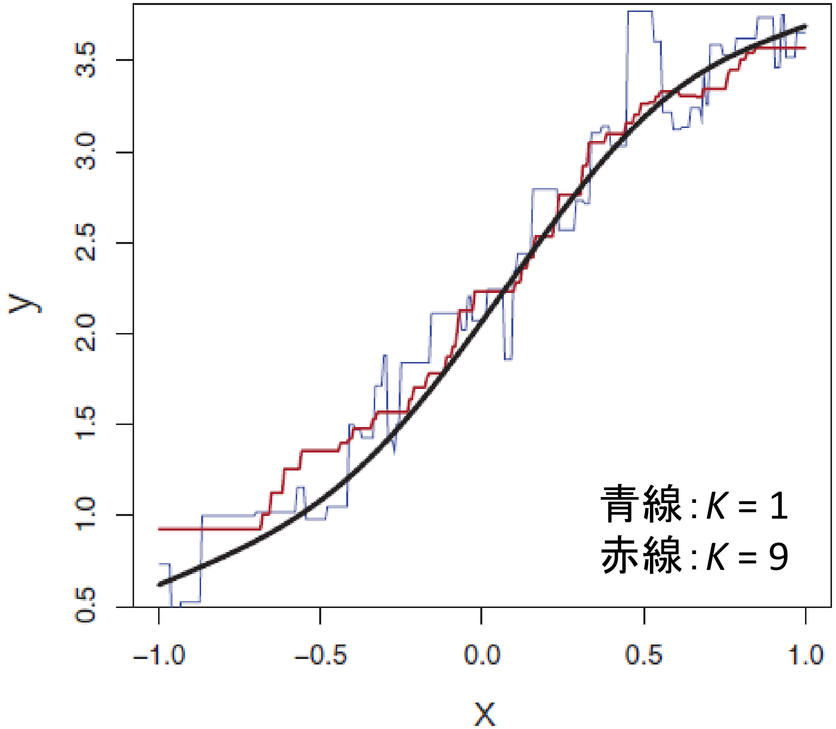
テストMSE



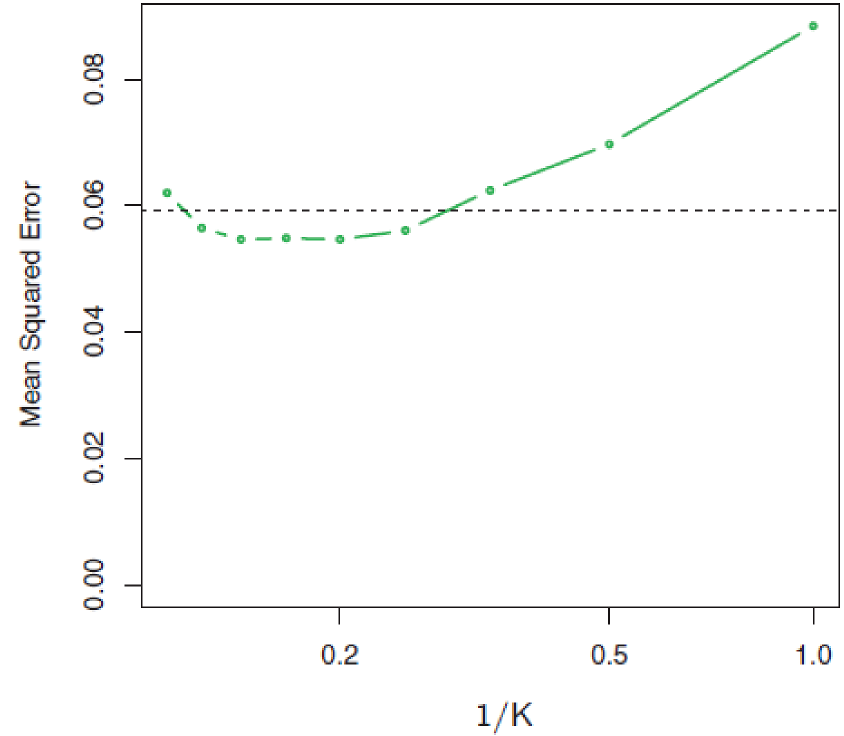
最適な K を選んでも
線形回帰の方が良い

線形回帰とK最近傍回帰

KNN



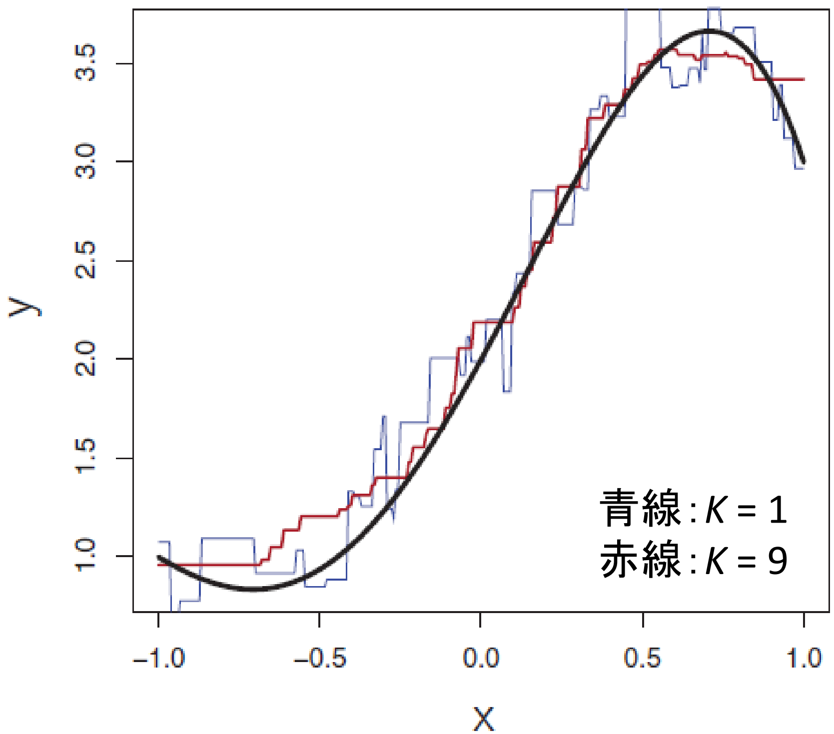
テストMSE



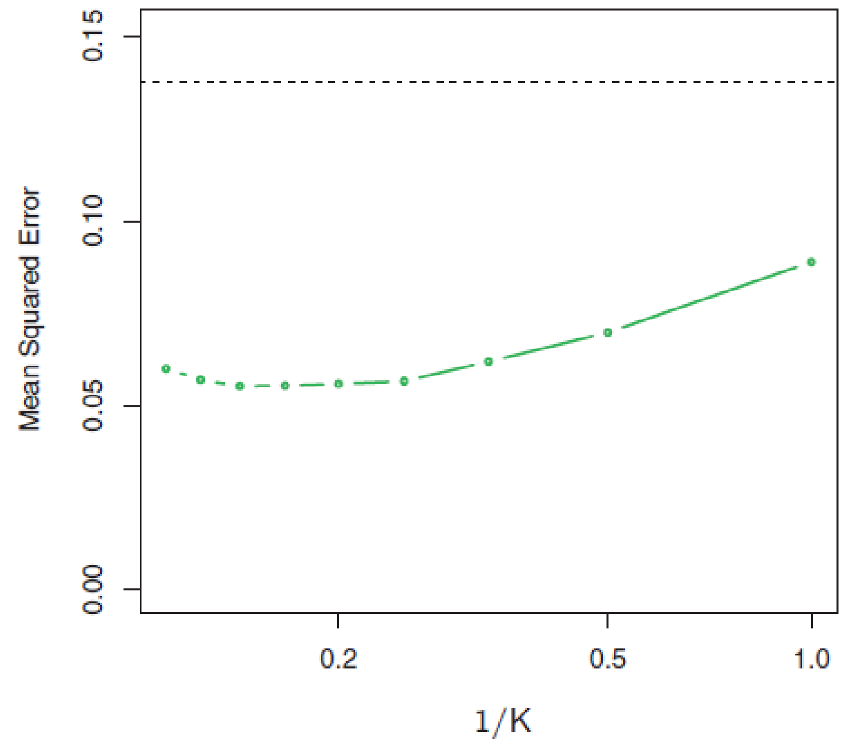
真の f に非線形がある場合は
K の値によっては KNN の方が良い

線形回帰とK最近傍回帰

KNN



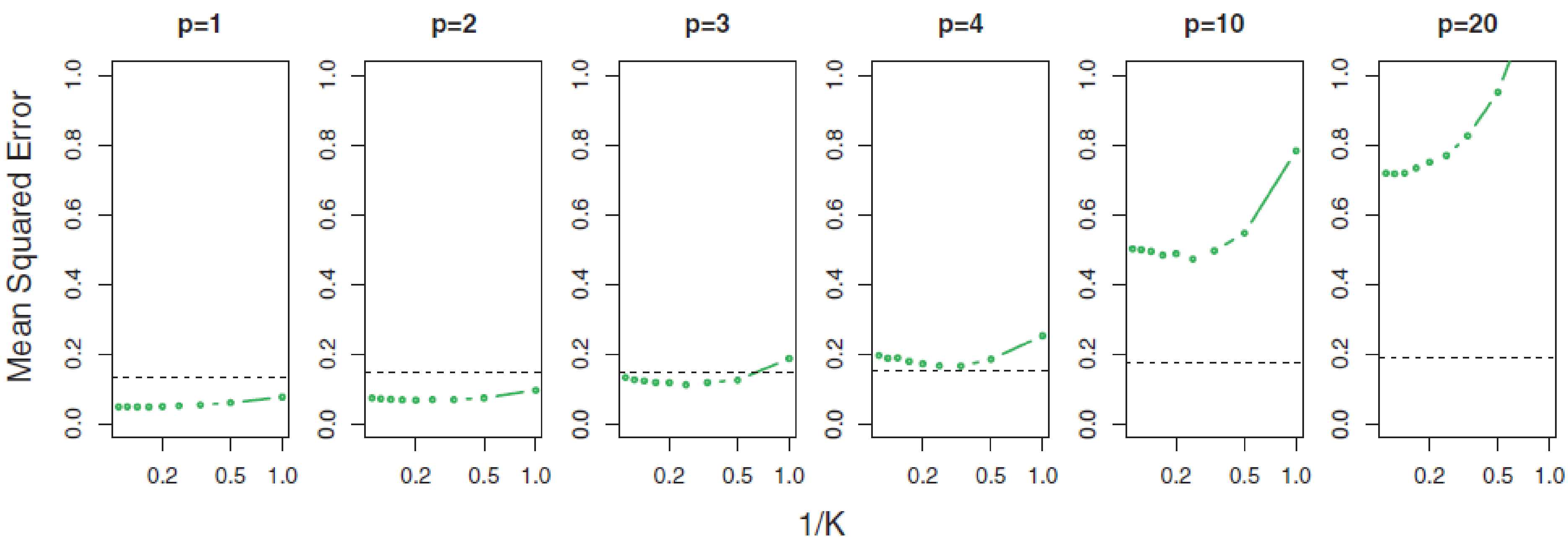
テストMSE



真の f の非線形が強い

線形回帰とK最近傍回帰

- 次元の呪い（curse of dimensionality）
  - 例）２次元目以降はすべてノイズ



KNN の性能悪化が線形回帰よりも大きい

分類（CLASSIFICATION）

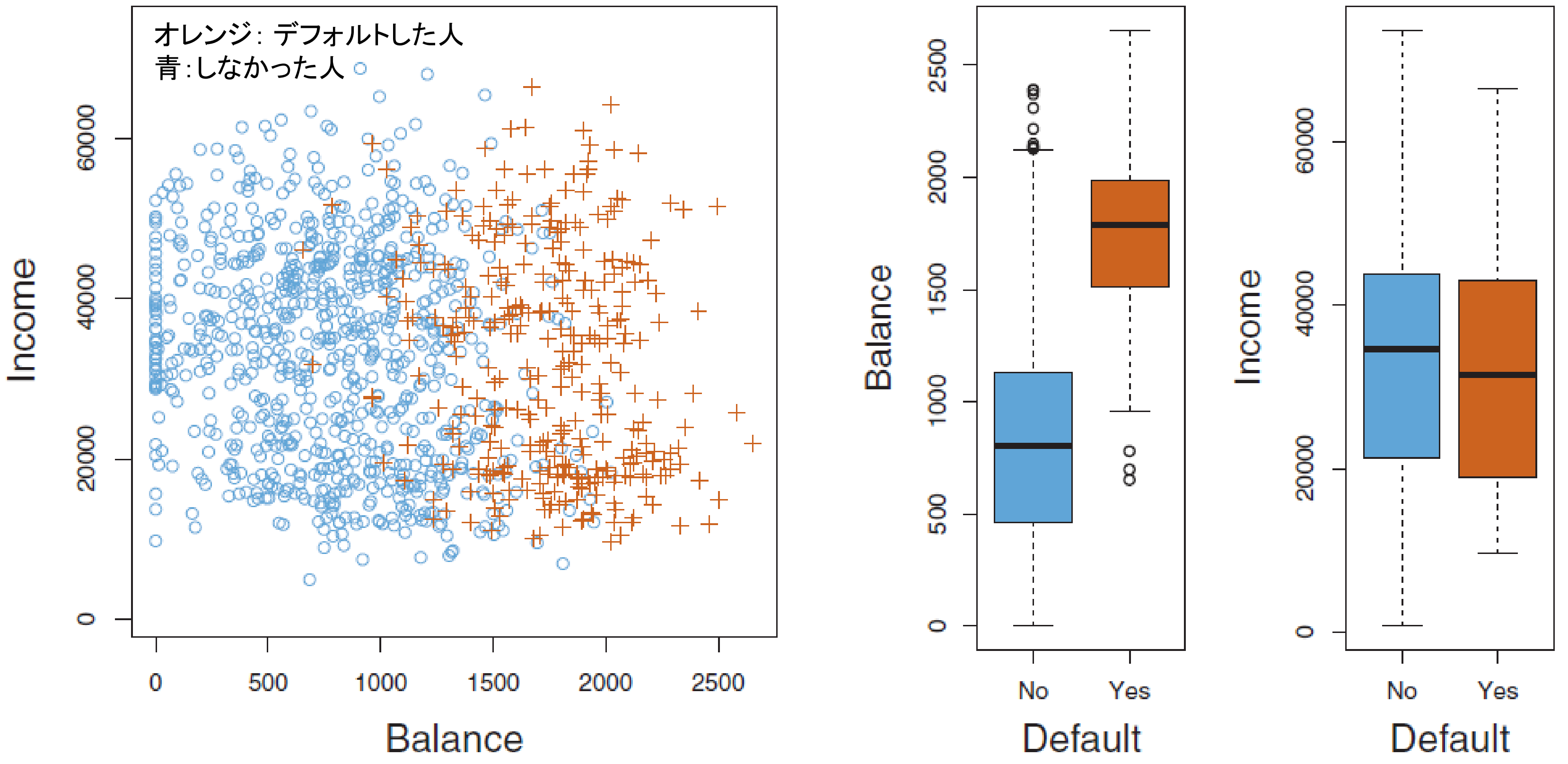
分類問題

- 分類問題
  - 応答変数が質的変数である予測問題
  - クラス(class)を予測する問題とも呼ばれる

- 例
  - 患者の症状から疾患を予測
  - オンラインバンキングのトランザクションが不正なものかどうかを判定
  - メールがスパムメールかどうかを判定
  - 画像から数字を認識
  - etc

分類問題

- Default データセット（人工データ）
  - – クレジットカードのデフォルト（債務不履行）を予測



なぜ線形回帰ではダメなのか

- 例）患者の症状から疾患を予測
  - 応答変数

$$Y = \begin{cases} 1 & \text{if } \texttt{stroke}; \\ 2 & \text{if } \texttt{drug overdose}; \\ 3 & \text{if } \texttt{epileptic seizure}. \end{cases}$$

として線形回帰を行うと

- ３つの疾患の間に順序関係（大小関係）が存在
- stroke と drug overdose の差と drug overdose とepileptic seizure の差が同じ

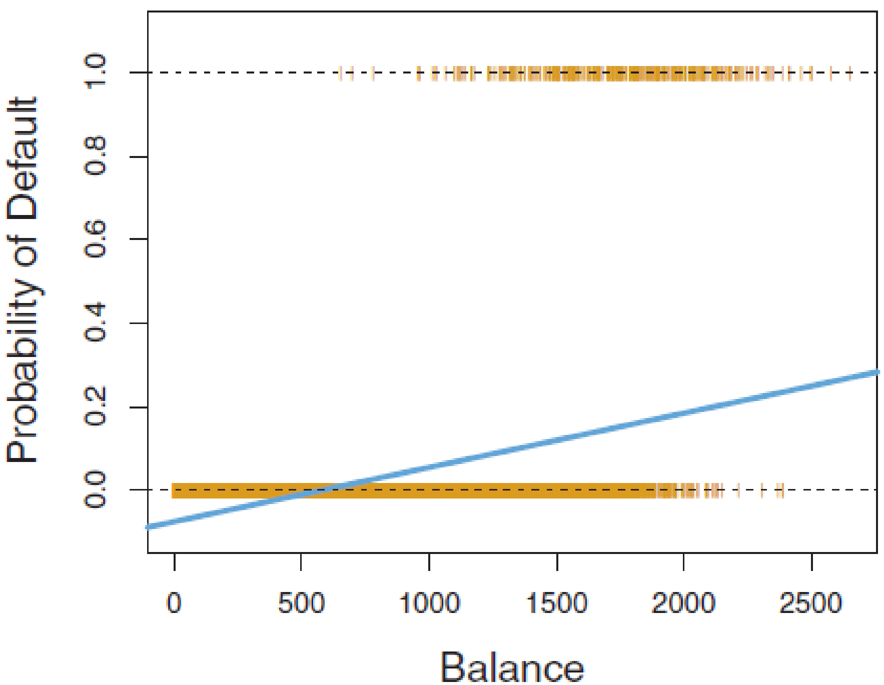
といった変な仮定をすることになる

ロジスティック回帰

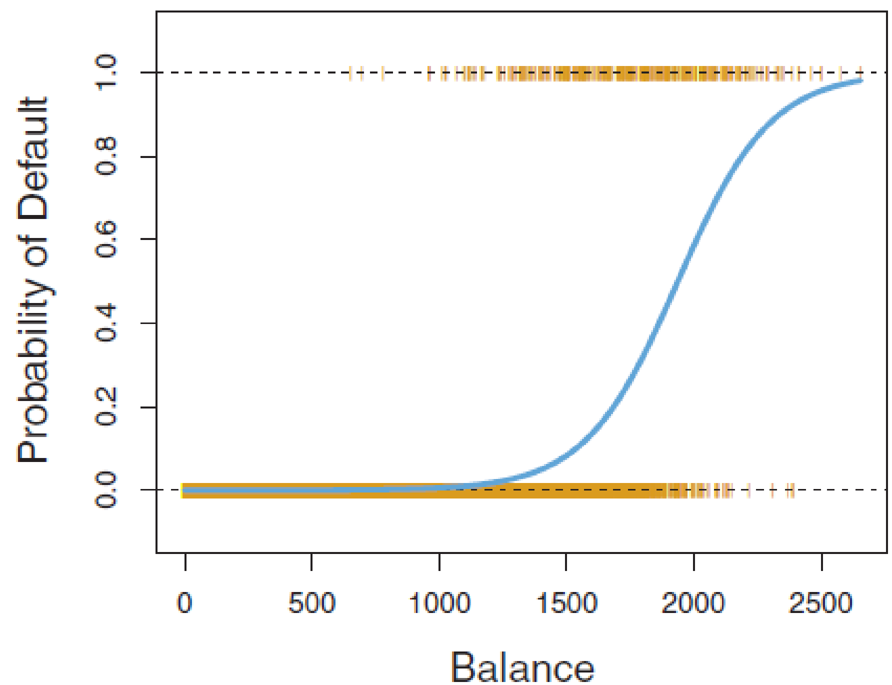
- Default データセット
  – 応答変数がとる値：yes または no

$\Pr(\texttt{default} = \texttt{Yes} | \texttt{balance})$ を予測するモデルを考える

線形回帰



ロジスティック回帰



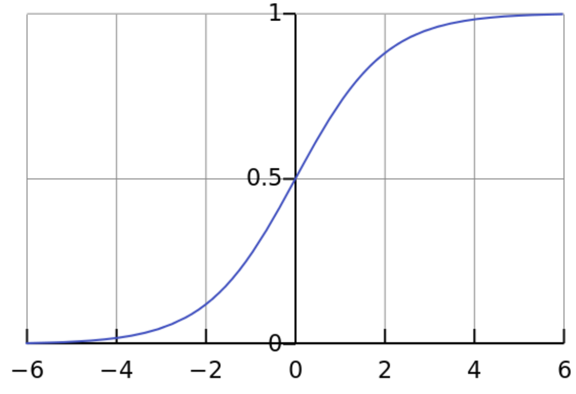
ロジスティック回帰

- ロジスティックモデル（logistic model）
  - ロジスティック関数を用いて確率値を出力

$$p(X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}}$$

  - 参考）標準シグモイド関数（ロジスティック関数の一例）

$$f(x) = \frac{1}{1 + e^{-x}}$$



ロジスティック回帰

- ロジスティックモデル
  - オッズ（odds）

$$\frac{p(X)}{1-p(X)} = e^{\beta_0+\beta_1 X}$$

  - ロジット（logit, log-odds）

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \beta_1 X$$

オッズ

$$\frac{0.5}{1-0.5}=1$$

$$\frac{0.9}{1-0.9}=9$$

$$\frac{0.99}{1-0.99}=99$$

ロジットが X の線形関数になっている

ロジスティック回帰

- 最尤推定（maximum likelihood estimation）
  - 代表的なパラメータ推定手法
  - 尤度関数（likelihood function）
    - パラメータの尤度（学習データの生成確率）

$$\ell(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i':y_{i'}=0} \left(1 - p(x_{i'})\right)$$

  - 尤度関数の値が最大になるようにパラメータの値を決定する

ロジスティック回帰

- パラメータ推定の例
  - balance から default する確率を予測するモデル

	Coefficient	Std. error	Z-static	p-value
Intercept	−10.6513	0.3612	−29.5	< 0.0001
balance	0.0055	0.0002	24.9	< 0.0001

↑ balance が 1 増えるとロジットが 0.0055 増える

ロジスティック回帰

- 予測
  - balance が \$1,000 の人が default する確率は？

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} = \frac{e^{-10.6513 + 0.0055 \times 1000}}{1 + e^{-10.6513 + 0.0055 \times 1000}} = 0.00576$$

  - balance が \$2,000 の人が default する確率は？

$$\hat{p}(X) = \frac{e^{\hat{\beta}_0 + \hat{\beta}_1 X}}{1 + e^{\hat{\beta}_0 + \hat{\beta}_1 X}} = \frac{e^{-10.6513 + 0.0055 \times 2000}}{1 + e^{-10.6513 + 0.0055 \times 2000}} = 0.586$$

ロジスティック回帰

- 質的変数の利用
  - student から default する確率を予測するモデル

	Coefficient	Std. error	Z-static	p-value
Intercept	−3.5041	0.0707	−49.55	< 0.0001
student [Yes]	0.4049	0.1150	3.52	0.0004

$$\widehat{\Pr}(\texttt{default}=\texttt{Yes}|\texttt{student}=\texttt{Yes}) = \frac{e^{-3.5041+0.4049\times 1}}{1+e^{-3.5041+0.4049\times 1}} = 0.0431$$

$$\widehat{\Pr}(\texttt{default}=\texttt{Yes}|\texttt{student}=\texttt{No}) = \frac{e^{-3.5041+0.4049\times 0}}{1+e^{-3.5041+0.4049\times 0}} = 0.0292$$

→ 学生のほうが default する確率が大きい

ロジスティック回帰

- 多重ロジスティック回帰（multiple logistic regression）
  - 複数の予測変数を利用

$$\log\left(\frac{p(X)}{1-p(X)}\right)=\beta_0+\beta_1X_1+\cdots+\beta_pX_p$$

$$p(X)=\frac{e^{\beta_0+\beta_1X_1+\cdots+\beta_pX_p}}{1+e^{\beta_0+\beta_1X_1+\cdots+\beta_pX_p}}$$

ロジスティック回帰

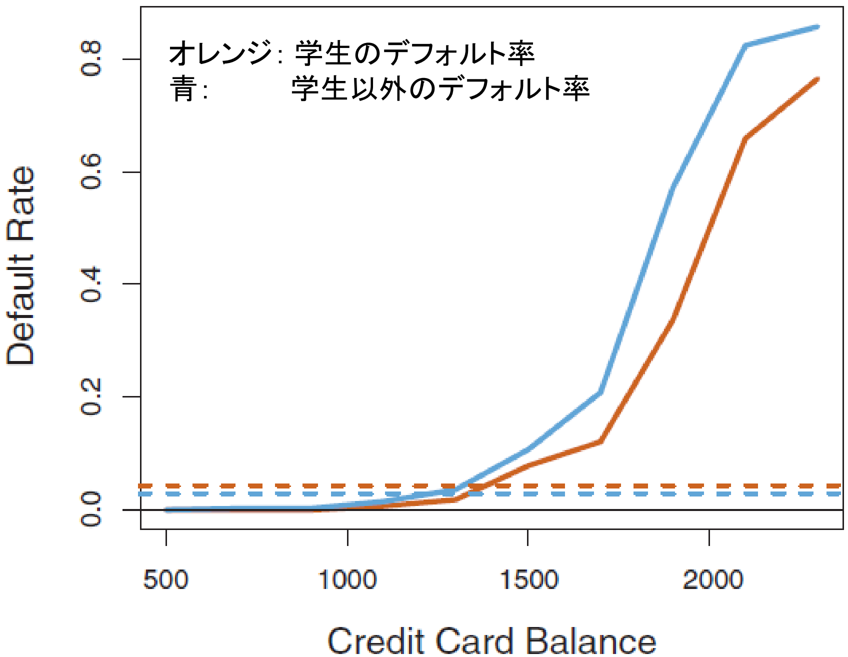
- 多重ロジスティック回帰の例
  - balance, income, student を使って default を予測

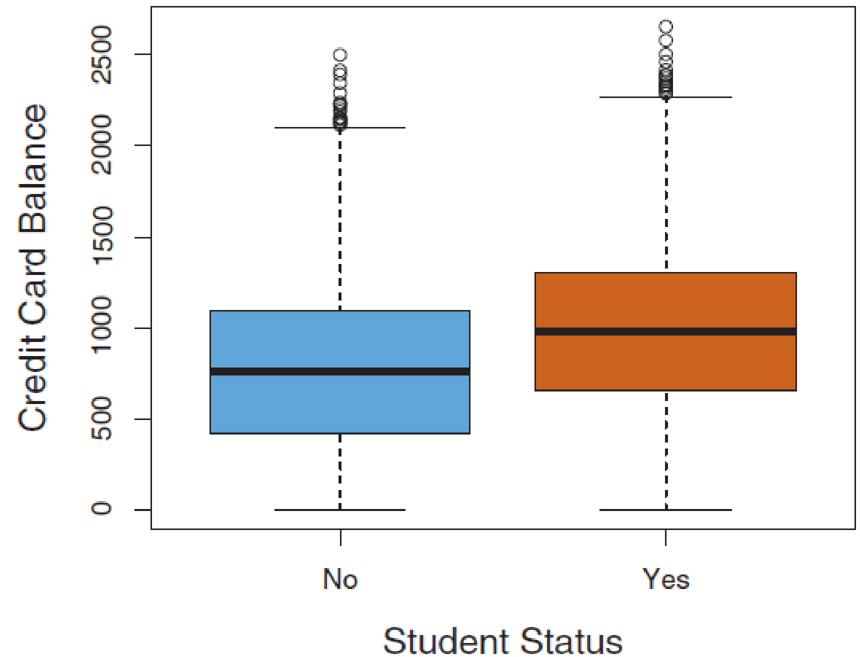
	Coefficient	Std. error	Z-static	p-value
Intercept	−10.8690	0.4923	−22.08	< 0.0001
balance	0.0057	0.0002	24.74	< 0.0001
income	0.0030	0.0082	0.37	0.7115
student [Yes]	−0.6468	0.2362	−2.74	0.0062

  - student の係数が負になっている
    - 学生のほうが default しやすかったはずでは？？

ロジスティック回帰

- 交絡（confounding）
  – 入力変数と出力変数の両方に相関する隠れた変数がある





全体的なデフォルト率は学生の方が高いが balance が同じであれば学生の方が低い

学生の方が balance が大きい

ロジスティック回帰

- 多重ロジスティック回帰による予測
  - balance が \$1,500、income が \$40,000 の学生が default する確率は？

$$\hat{p}(X) = \frac{e^{-10.869+0.00574\times 1,500+0.003\times 40-0.6468\times 1}}{1+e^{-10.869+0.00574\times 1,500+0.003\times 40-0.6468\times 1}} = 0.058$$

  - 同じ balance, income で学生でない場合は？

$$\hat{p}(X) = \frac{e^{-10.869+0.00574\times 1,500+0.003\times 40-0.6468\times 0}}{1+e^{-10.869+0.00574\times 1,500+0.003\times 40-0.6468\times 0}} = 0.105$$

ロジスティック回帰

- 応答変数が多値をとる場合
  - 例）患者の症状から疾患を予測
    - 応答変数の値：stroke, drug overdose, epileptic seizure

$$\Pr(Y = \text{stroke} \mid X)$$
$$\Pr(Y = \text{drug overdose} \mid X)$$
$$\Pr(Y = \text{epileptic seizure} \mid X) = 1 - \Pr(Y = \text{stroke} \mid X) - \Pr(Y = \text{drug overdose} \mid X)$$

  - ソフトマックス回帰（softmax regression）

$$\Pr(Y = k \mid X) \propto \exp\left(\sum_p \beta_{kp} X_p \right)$$

Python実習

- Stock Market データセット
  - Smarket
    - S&P 500 stock index の値動き(%)データ
    - 2001年～2005年(1250日分)

Year	Lag1	Lag2	Lag3	Lag4	Lag5	Volume	Today	Direction
2001	0.381	-0.192	-2.624	-1.055	5.01	1.1913	0.959	Up
2001	0.959	0.381	-0.192	-2.624	-1.055	1.2965	1.032	Up
2001	1.032	0.959	0.381	-0.192	-2.624	1.4112	-0.623	Down
2001	-0.623	1.032	0.959	0.381	-0.192	1.276	0.614	Up
2001	0.614	-0.623	1.032	0.959	0.381	1.2057	0.213	Up
2001	0.213	0.614	-0.623	1.032	0.959	1.3491	1.392	Up
2001	1.392	0.213	0.614	-0.623	1.032	1.445	-0.403	Down
:	:	:	:	:	:	:	:	:

Python実習

- データの読み込み
  - ダウンロード
    - http://www.logos.t.u-tokyo.ac.jp/~tsuruoka/lecture/sml/Smarket.csv
    - 上記ファイルをホームディレクトリに保存（Windowsの場合）

```
>>> import os
>>> os.chdir(os.path.expanduser("~"))
```

  - データフレームとして読み込み

```
>>> import pandas as pd
>>> Smarket = pd.read_csv('Smarket.csv')
>>> Smarket.head(10)
...
>>> list(Smarket)
...
>>> Smarket.shape
...
```

Python実習

- 相関
  - corr()

```
>>> Smarket.corr()
             Year      Lag1      Lag2  ...      Lag5    Volume     Today
Year     1.000000  0.029700  0.030596  ...  0.029788  0.539006  0.030095
Lag1     0.029700  1.000000 -0.026294  ... -0.005675  0.040910 -0.026155
Lag2     0.030596 -0.026294  1.000000  ... -0.003558 -0.043383 -0.010250
Lag3     0.033195 -0.010803 -0.025897  ... -0.018808 -0.041824 -0.002448
Lag4     0.035689 -0.002986 -0.010854  ... -0.027084 -0.048414 -0.006900
Lag5     0.029788 -0.005675 -0.003558  ...  1.000000 -0.022002 -0.034860
Volume   0.539006  0.040910 -0.043383  ... -0.022002  1.000000  0.014592
Today    0.030095 -0.026155 -0.010250  ... -0.034860  0.014592  1.000000
```

- プロット

- Year と Volume の間にのみ強い相関
- 取引量が年々増加

```
>>> import matplotlib.pyplot as plt
>>> plt.plot(Smarket[['Volume']])
>>> plot.show()
```

Python実習

- ロジスティック回帰
  - Logit(), fit()

```
>>> import statsmodels.discrete.discrete_model as sm
>>> from patsy import dmatrices
>>> y, X = dmatrices('Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume', Smarket,
return_type = 'dataframe')
>>> trained = sm.Logit(y['Direction[Up]'], X).fit()
>>> trained.summary()
...
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept     -0.1260      0.241     -0.523      0.601      -0.598       0.346
Lag1          -0.0731      0.050     -1.457      0.145      -0.171       0.025
Lag2          -0.0423      0.050     -0.845      0.398      -0.140       0.056
Lag3           0.0111      0.050      0.222      0.824      -0.087       0.109
Lag4           0.0094      0.050      0.187      0.851      -0.089       0.107
Lag5           0.0103      0.050      0.208      0.835      -0.087       0.107
Volume         0.1354      0.158      0.855      0.392      -0.175       0.446
==============================================================================
```

p値が一番小さいのはLag1

Python実習

- モデルの係数、確率

```
>>> trained.params
Intercept   -0.126000
Lag1        -0.073074
Lag2        -0.042301
Lag3         0.011085
Lag4         0.009359
Lag5         0.010313
Volume       0.135441
dtype: float64
>>> trained.predict()
array([0.50708413, 0.48146788, 0.48113883, ..., 0.5392683 ,
0.52611829, 0.51791656])
```

Python実習

- 予測精度を評価
  - 確率をクラスラベルに変換

```
>>> import numpy as np
>>> predict_label = pd.DataFrame(np.zeros(1250))
>>> predict_label.iloc[trained.predict() > 0.5] = 1
```

  - （学習データでの）正解率を計算

```
>>> from sklearn.metrics import confusion_matrix
>>> confusion_matrix(y.iloc[:,1], predict_label.iloc[:,0])
array([[145, 457],
       [141, 507]], dtype=int64)
>>> (507+145)/1250
0.5216
>>> np.mean(y.iloc[:,1] == predict_label.iloc[:,0])
0.5216
```

Python実習

- 学習データとテストデータを分離

```
>>> Smarket_2005 = Smarket.query('Year >= 2005')
>>> Smarket_2005.shape
(252, 9)
>>> Smarket_train = Smarket.query('Year < 2005')
>>> Smarket_train.shape
(998, 9)
```

- 学習データを使ってモデルのパラメータを決定

```
>>> y_train, X_train = dmatrices('Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + 
Lag5 + Volume', Smarket_train, return_type = 'dataframe')
>>> trained = sm.Logit(y_train['Direction[Up]'], X_train).fit()
>>> trained.summary()
...
```

Python実習

- テストデータで評価

```
>>> y_test, X_test = dmatrices('Direction ~ Lag1 + Lag2 + Lag3 + Lag4 +
Lag5 + Volume', Smarket_2005, return_type = 'dataframe')
>>> preds = trained.predict(X_test)
>>> predict_label = pd.DataFrame(np.zeros(shape=(len(X_test), 1)))
>>> mark = (preds > 0.5).reset_index(drop=True)
>>> predict_label.loc[mark] = 1
>>> confusion_matrix(y_test.iloc[:, 1], predict_label.iloc[:, 0])
array([[77, 34],
       [97, 44]], dtype=int64)
>>> (77 + 44) / 252
0.4801587301587302
>>> np.mean(y_test.iloc[:, 1].values == predict_label.iloc[:, 0].values)
0.4801587301587302
```

Python実習

- 予測変数を絞ってみる

```
>>> y_train, X_train = dmatrices('Direction ~ Lag1 + Lag2', Smarket_train,
return_type = 'dataframe')
>>> trained = sm.Logit(y_train['Direction[Up]'], X_train).fit()
>>> y_test, X_test = dmatrices('Direction ~ Lag1 + Lag2', Smarket_2005,
return_type = 'dataframe')
>>> preds = trained.predict(X_test)
>>> predict_label = pd.DataFrame(np.zeros(shape=(len(X_test), 1)))
>>> mark = (preds > 0.5).reset_index(drop=True)
>>> predict_label.loc[mark] = 1
>>> np.mean(y_test.iloc[:, 1].values == predict_label.iloc[:, 0].values)
...
```

Python実習

- 入力変数の値を指定して予測

```
>>> newdata = pd.DataFrame({'Direction': np.zeros(2),
'Lag1': [1.2, 1.5], 'Lag2': [1.1, -0.8]})
>>> X_test = dmatrices('Direction ~ Lag1 + Lag2', newdata,
return_type = 'dataframe')[1]
>>> trained.predict(X_test)
0     0.479146    ← 入力: Lag1=1.2, Lag2=1.1
1     0.496094    ← 入力: Lag1=1.5, Lag2=-0.8
dtype: float64
```

線形判別分析

- 線形判別分析（linear discriminant analysis, LDA）
  - ロジスティック回帰のように、直接 $\Pr(Y = k \mid X = x)$ をモデル化するのではなく、 $\Pr(X = x \mid Y = k)$ を正規分布でモデル化し、ベイズの定理によって $\Pr(Y = k \mid X = x)$ を計算

- ロジスティック回帰との比較
  - ロジスティック回帰はクラス間のデータ分布が完全に分かれている場合、パラメータ推定が不安定になる
  - データ分布がクラスごとに正規分布に従っている場合、線形判別分析のほうがパラメータ推定が安定する

線形判別分析

- ベイズの定理による事後確率の計算

$$\Pr\left(Y=k \middle| X=x\right)=\frac{\Pr\left(Y=k\right)\Pr\left(X=x \middle| Y=k\right)}{\Pr\left(X=x\right)}$$

$$=\frac{\Pr\left(Y=k\right)\Pr\left(X=x \middle| Y=k\right)}{\sum_{l=1}^{K}\Pr\left(X=x,Y=l\right)}$$

$$=\frac{\Pr\left(Y=k\right)\Pr\left(X=x \middle| Y=k\right)}{\sum_{l=1}^{K}\Pr\left(Y=l\right)\Pr\left(X=x \mid Y=l\right)}$$

$$p_k\left(x\right)=\frac{\pi_k f_k\left(x\right)}{\sum_{l=1}^{K}\pi_l f_l\left(x\right)}$$

K: クラスの数
π_k: クラス k の事前確率
$f_k(x)$: クラス k のデータの密度関数

線形判別分析

- 予測変数がひとつ（p = 1）の場合
  - $f_k(x)$ が正規分布だと仮定

$$f_k(x) = \frac{1}{\sqrt{2\pi}\sigma_k} \exp\left(-\frac{1}{2\sigma_k^2} (x - \mu_k)^2 \right)$$

  - 分散がすべてのクラスで等しいとすると

$$p_k(x) = \frac{\pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2} (x - \mu_k)^2 \right)}{\sum_{l=1}^{K} \pi_l \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2} (x - \mu_l)^2 \right)}$$

線形判別分析

- 分類

$$p_k(x) \propto \pi_k \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{1}{2\sigma^2}(x-\mu_k)^2 \right)$$

が最も大きいクラス k に分類すればよい。右辺の対数をとってクラスに依存しない項を削除した、

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

を考えても良い。

線形判別分析

- 分類

$K = 2, \pi_1 = \pi_2$ の場合、分離境界は

$$x \cdot \frac{\mu_1}{\sigma^2} - \frac{\mu_1^2}{2\sigma^2} + \log(\pi_1) = x \cdot \frac{\mu_2}{\sigma^2} - \frac{\mu_2^2}{2\sigma^2} + \log(\pi_2)$$

より

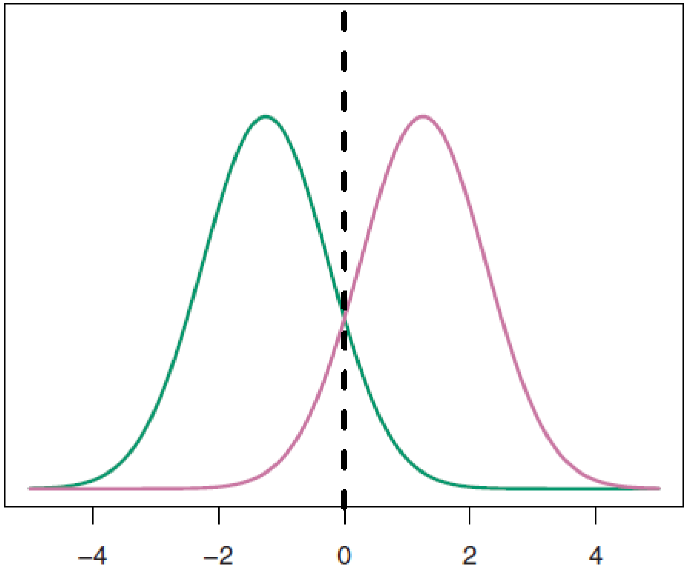
$$x = \frac{\mu_1^2 - \mu_2^2}{2(\mu_1 - \mu_2)} = \frac{\mu_1 + \mu_2}{2}$$

線形判別分析

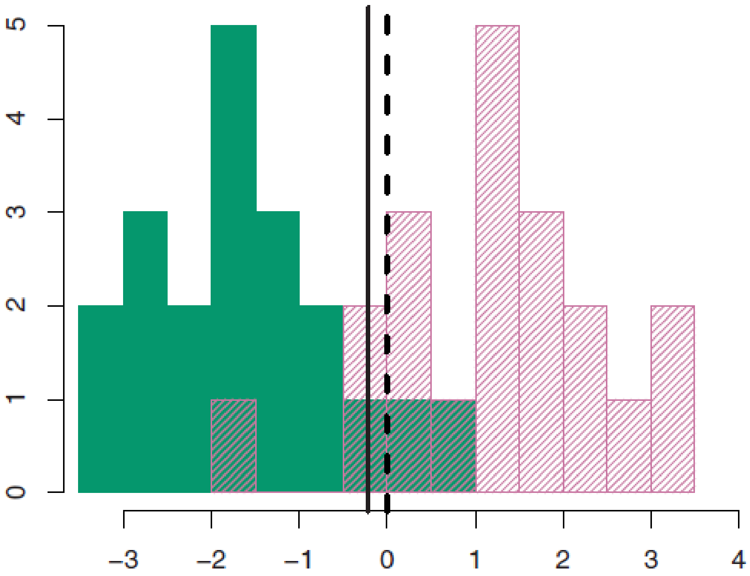
- 例（人工データ）

それぞれの正規分布から20個ずつサンプリング

点線：ベイズ決定境界



実線：LDA決定境界



$\mu_1 = -1.25$　　$\mu_2 = 1.25$

$\sigma_1^2 = 1$　　$\sigma_2^2 = 1$

ベイズエラー率：10.6%

LDAのテストエラー率：11.1%

線形判別分析

- パラメータ推定
  - 実際には μ_k, σ^2 は未知なのでデータから推定

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i \qquad \hat{\sigma}^2 = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} \left(x_i - \hat{\mu}_k \right)^2$$

n: 学習データの全事例数
n_k: クラス k に属する事例の数

線形判別分析

- パラメータ推定
  - π_k も未知の場合はデータから推定

$$\hat{\pi}_k = \frac{n_k}{n}$$

  - これらの推定値を用いて

$$\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)$$

がもっとも大きくなるクラスに分類

  - 判別関数 $\hat{\delta}_k(x)$ は x の線形関数

線形判別分析

- 予測変数が2個以上（$p > 1$）の場合
  - X（p次元のベクトル）が多変量正規分布（multivariate Gaussian distribution）に従うと仮定

$$X \sim N(\mu, \Sigma)$$

$$E(X) = \mu$$ ・・・ X の平均

$$\mathrm{Cov}(X) = \Sigma$$ ・・・ X の分散共分散行列（$p \times p$）

  - 多変量正規分布の確率密度関数

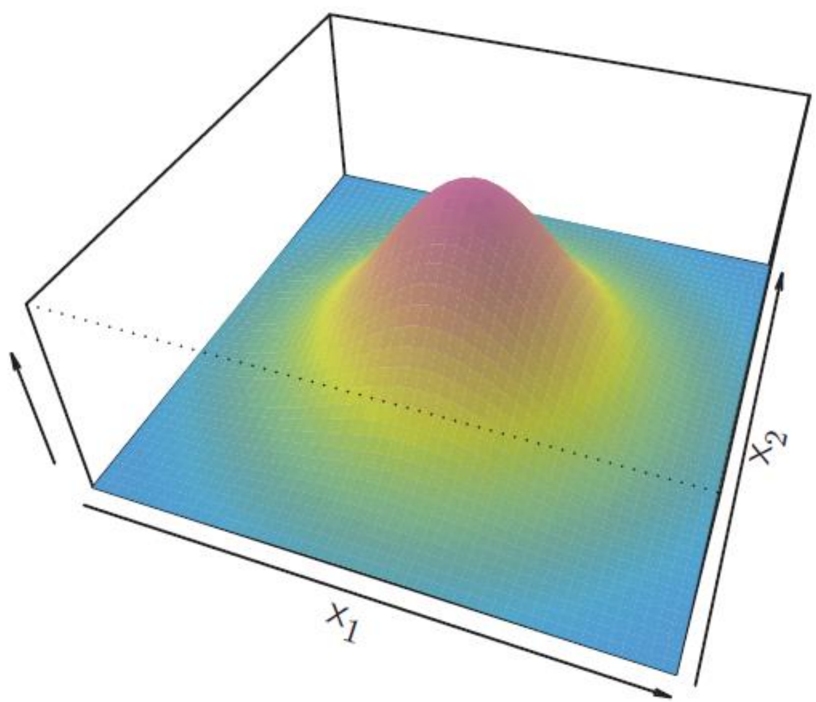
$$f(x) = \frac{1}{(2\pi)^{p/2} |\Sigma|^{1/2}} \exp\left(-\frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) \right)$$

Σの行列式

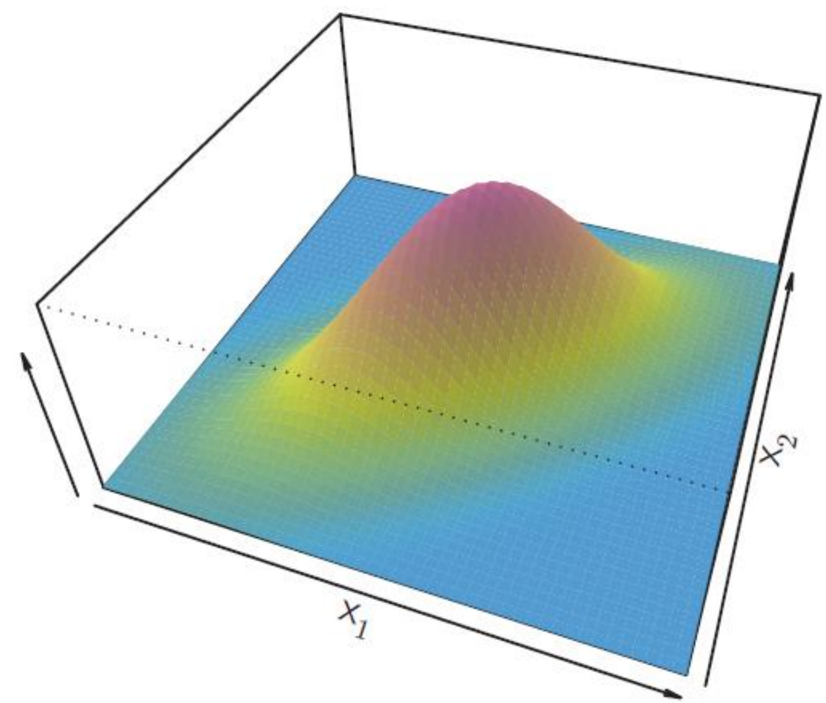
線形判別分析

- 多変量正規分布

２つの変数に相関がない場合



相関係数0.7



線形判別分析

- 分類

$$\pi_k f(x) = \frac{\pi_k}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu_k)^T \Sigma^{-1} (x-\mu_k) \right)$$

が最大のクラスに分類すればよい

対数をとって k に依存しない項を削除すると

$$\delta_k(x) = x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k$$

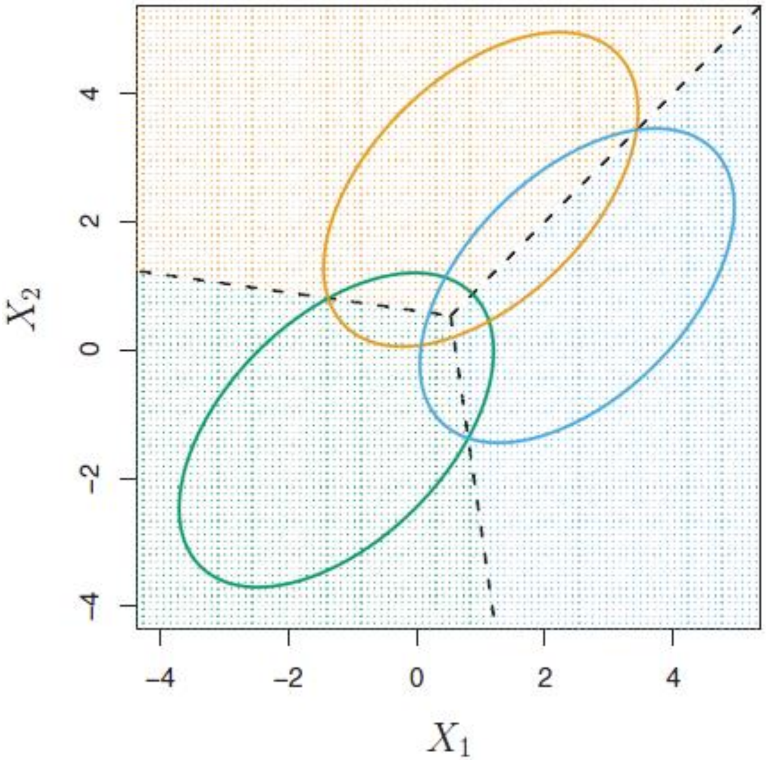
→ x の線形関数

線形判別分析

$K = 3$, $p = 2$ の例

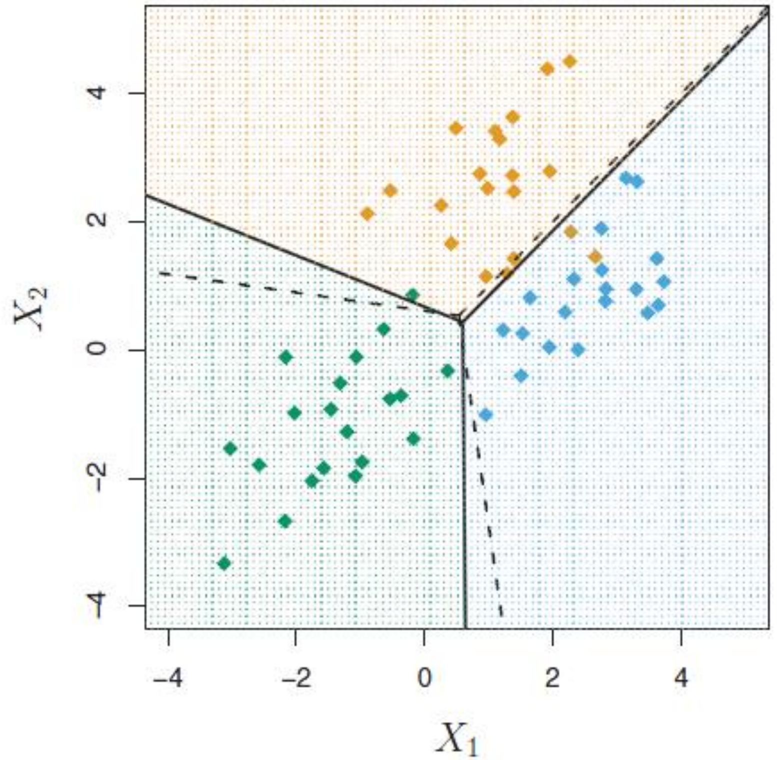
楕円：各多変量正規分布の95%の範囲

点線：ベイズ決定境界



学習データ：各クラス20事例

実線：LDA決定境界



ベイズエラー率：7.46%

LDAのテストエラー率：7.7%

線形判別分析

- LDA決定境界
  - クラス k とクラス l の境界

$$\delta_k(x) = \delta_l(x)$$

$$x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k + \log \pi_k = x^T \Sigma^{-1} \mu_l - \frac{1}{2} \mu_l^T \Sigma^{-1} \mu_l + \log \pi_l$$

  - 前ページの例の場合 $\pi_k = \pi_l$ なので

$$x^T \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^T \Sigma^{-1} \mu_k = x^T \Sigma^{-1} \mu_l - \frac{1}{2} \mu_l^T \Sigma^{-1} \mu_l$$

→ 直線（$p > 2$ の場合は超平面）

線形判別分析

- パラメータ推定

$$\hat{\pi}_k = \frac{n_k}{n}$$

$$\hat{\mu}_k = \frac{1}{n_k} \sum_{i:y_i=k} x_i$$

$$\hat{\Sigma} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:y_i=k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$