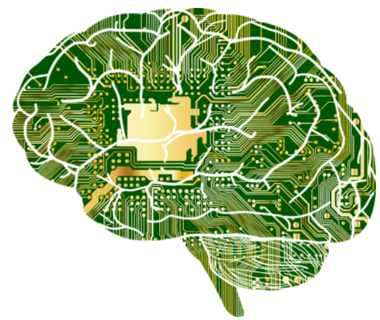
データと学習

統計的学習

- データからの学習

データ

教師付き学習
（supervised learning）



予測
（prediction）

推論
（inference）

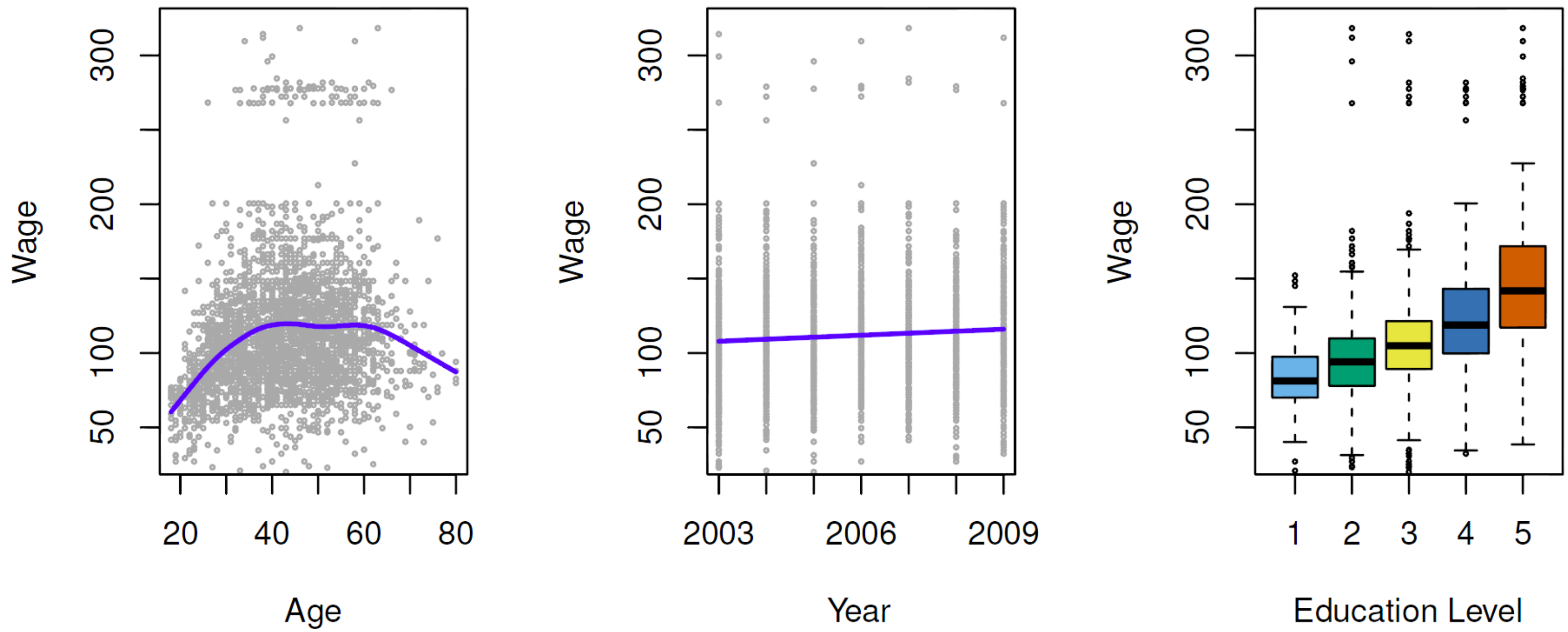
教師なし学習
（unsupervised learning）

分析
（analysis）

Wage Data

- アメリカ東海岸地域の賃金データ
- 入力
  - age: 労働者の年齢
  - education: 最終学歴（1: 高校未卒 ～ 5: 大学院）
  - year: 調査した年（2003年 ～ 2009年）
  - etc.
- 出力
  - wage: 賃金
- 教師付き学習（supervised learning）
  - 入力から出力を予測する回帰（regression）問題

Wage Data



- それぞれの属性だけで賃金を正確に予測することは難しい
- 属性と賃金の間に非線形な関係がある

Many of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.

補足

箱ひげ図（box plot）

- 要約統計量
  - Q1: 第1四分位点
  - Q3: 第3四分位点
  - IQR: Q3 – Q1

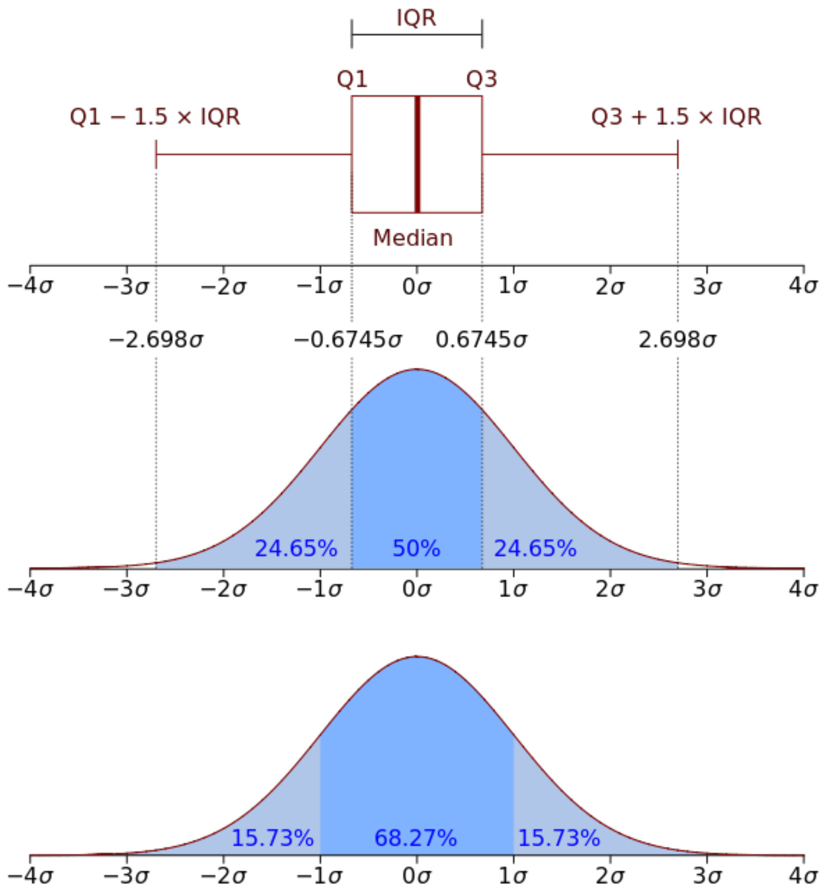


Figure by Jhguch　

Smarket Data

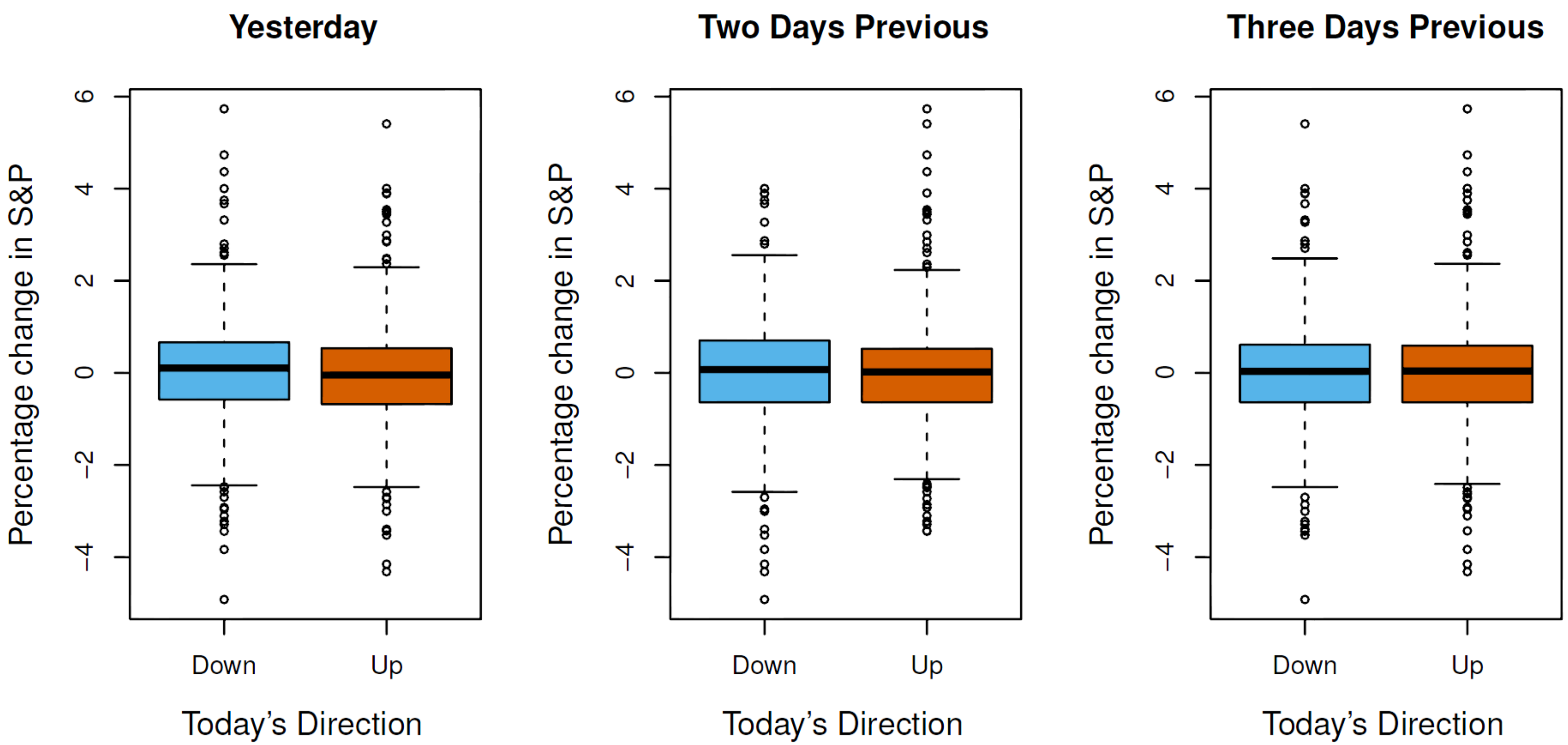
- 株式市場インデックス（S&P 500）の変動データ

- 入力
  - 直前5日間の価格変動

- 出力
  - 当日、インデックスが上昇した（Up）か下落した（Down）か

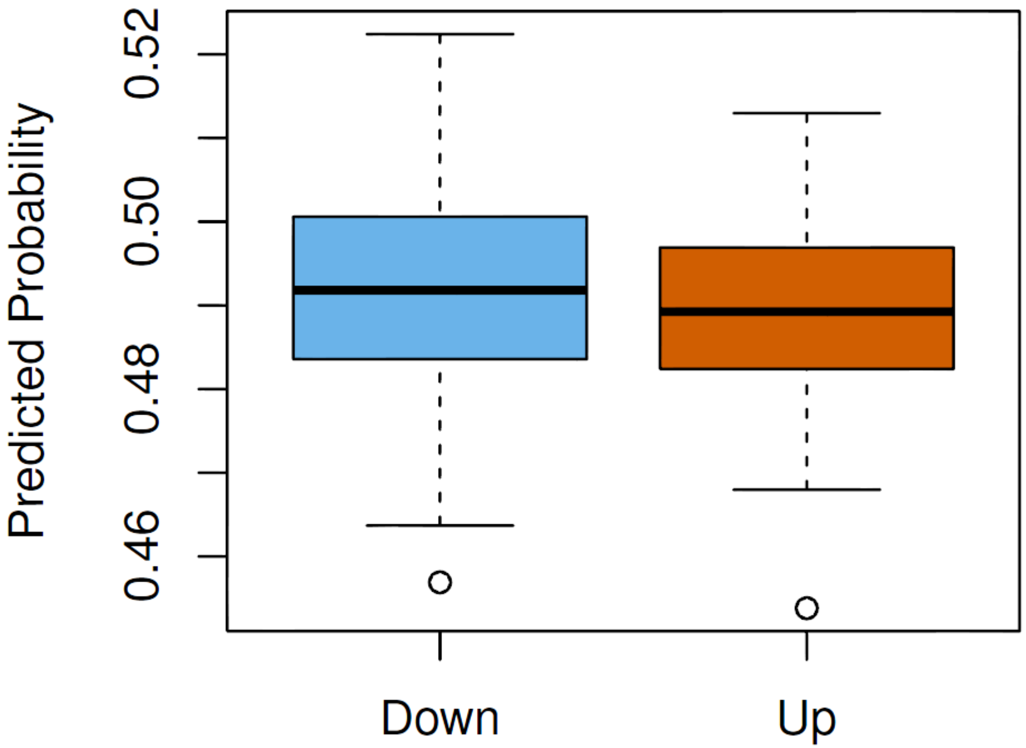
- 教師付き学習
  - カテゴリを予測する**分類（classification）**問題

Smarket Data



二次判別分析による予測結果

インデックスが
下落する確率
（予測）



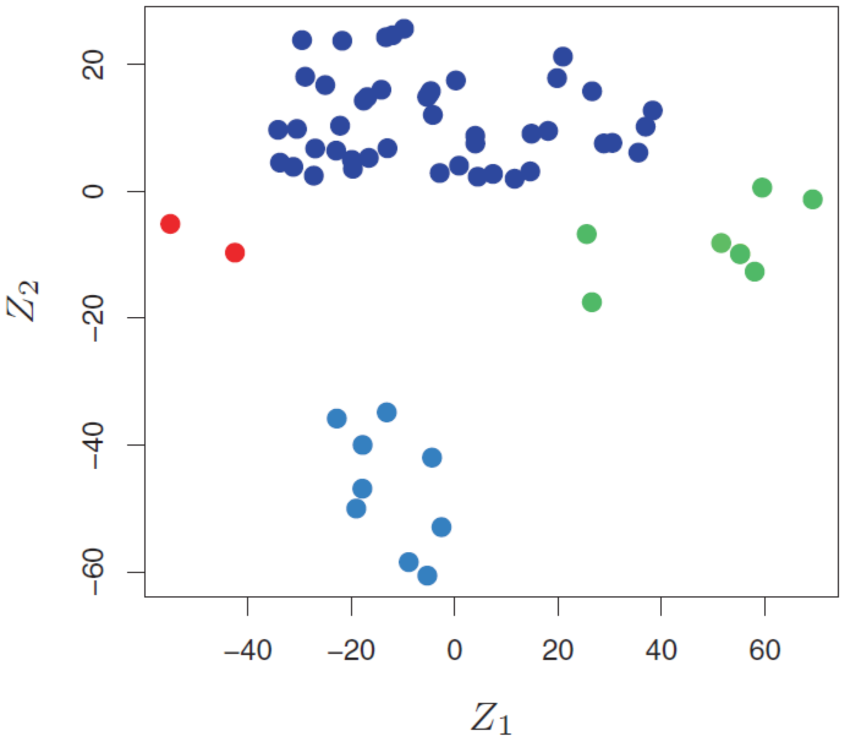
実際のインデックスの変化

NCI60 Data

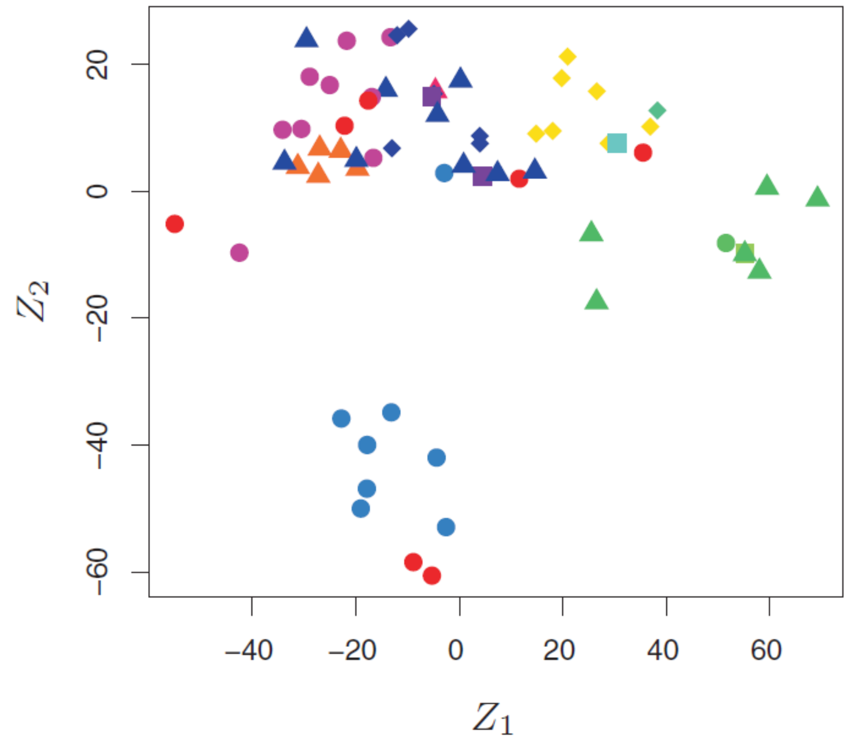
- 遺伝子の発現データ
- 64個の細胞株のそれぞれに関して、6830個の遺伝子の発現量を観測
- 教師なし学習
  - クラスタリング（clustering）

NCI60 Data

- 主成分分析による次元削減: 6830次元 → 2次元



クラスタリングの結果



癌の種類ごとに色分け

（この授業での）データの記法

- データの表現
  - n: 観測（データ点）の数（サンプルサイズ）
  - p: 予測に使える属性の数

- 例
  - Wage data
    - $n = 3000$
      - 3,000人分のデータ
    - $p = 12$
      - year, age, education, …

（この授業での）データの記法

- データの行列表現

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} \boxed{\text{観測データ1}} \\ \boxed{\text{観測データ2}} \\ \vdots \quad \vdots \quad \ddots \quad \vdots \\ \boxed{\text{観測データ}n} \end{pmatrix}$$

x_{ij}: i番の観測データのj番の属性（変数）の値

（この授業での）データの記法

- i番の観測データを表すベクトル

転置

$$x_i^T = \begin{pmatrix} x_{i1} & x_{i2} & \cdots & x_{ip} \end{pmatrix} \quad \text{あるいは} \quad x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}$$

ベクトルは基本的に
列（縦）ベクトル

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

（この授業での）データの記法

- j番の変数を表すベクトル

$$\mathbf{x}_j = \begin{pmatrix} x_{1j} \\ x_{2j} \\ \vdots \\ x_{nj} \end{pmatrix}$$

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix} \quad \Rightarrow \quad \mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_p \end{pmatrix}$$

（この授業での）データの記法

- 予測の対象
  - 例）wage

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}$$

スカラー値

- 観測データ全体

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

要素数 p のベクトル

（この授業での）データの記法

- ベクトルや行列のサイズの表記法

$$\mathbf{y} \in \mathbb{R}^n$$

→　$\mathbf{y}$ は実数値を要素に持つサイズ n のベクトル

$$\mathbf{X} \in \mathbb{R}^{n \times p}$$

→　$\mathbf{X}$ は実数値を要素に持つサイズ $n \times p$ の行列

$$\mathbb{R}^n = \underbrace{\mathbb{R} \times \mathbb{R} \times \ldots \times \mathbb{R}}_{n\text{個}}$$

実数全体の集合の直積

Python のインストール

- Windows の場合
  - https://www.python.org/ からダウンロードしてインストーラを実行
  - コマンドプロンプトからモジュールのインストール

```
> py -m pip install numpy
> py -m pip install matplotlib
> py -m pip install pandas
> py -m pip install scikit-learn
```

  - Pythonシェルの起動

```
> py
```

Pythonのインストール

- 最近の Ubuntu など
  - モジュールのインストール

```
> sudo apt-get install python3-pip
> python3 –m pip install numpy
> python3 –m pip install matplotlib
> python3 –m pip install pandas
> python3 –m pip install scikit-learn
```

  - Pythonシェルの立ち上げ

```
> python3
```

NumPy入門

- 準備
  - NumPy モジュールの読み込み

```
>>> import numpy as np
```

- ベクトルの作成
  - `array()`
  - 実行例

```
>>> x = np.array([1, 3, 2, 5])
>>> x
array([1, 3, 2, 5])
```

NumPy入門

- ベクトルのサイズ
  - `len()`
- ベクトルの和
  - +演算子

```
>>> x = np.array([1, 6, 2])
>>> y = np.array([1, 4, 3])
>>> len(x)
>>> len(y)
>>> x + y
array([ 2, 10,  5])
```

NumPy入門

- 行列の作成
  - array()
  - 属性 shape: 行列のサイズ
  - 属性 ndim: 次元数
  - 属性 size: 要素数

```
>>> x = np.array([[1, 3], [2, 4]])
>>> x
array([[1, 3],
       [2, 4]])
>>> x.shape
(2, 2)
>>> x.ndim
2
>>> x.size
4
```

NumPy入門

- 行列の転置
  - 属性 T

```
>>> x.T
array([[1, 2],
       [3, 4]])
```

NumPy入門

- 要素ごとの演算
  - sqrt()

```
>>> np.sqrt(x)
array([[ 1.        ,  1.73205081],
       [ 1.41421356,  2.        ]])
```

  - 2乗

```
>>> x**2
array([[ 1,  9],
       [ 4, 16]])
```

NumPy入門

- 乱数(正規分布)によるベクトルの生成
  - normal(平均, 標準偏差, 要素数)

```
>>> x = np.random.normal(0, 1, 50)
>>> from numpy.random import *
>>> x = normal(0, 1, 50)           ← モジュール名の省略
```

- 相関係数
  - corrcoef()

```
>>> y = x + normal(50, 0.1, 50)
>>> np.corrcoef(x, y)
array([[ 1.        ,  0.9964696],
       [ 0.9964696,  1.        ]])
```

NumPy入門

- 乱数のseedの指定
  - seed()
  - 同じ系列の疑似乱数を発生させられる

```
>>> seed(1303)
>>> normal(0, 1, 3)
array([-0.03425693,  0.06035959,  0.45511859])
>>> normal(0, 1, 3)
array([-0.36593175, -1.6773304 ,  0.5910023 ])
>>> seed(1303)
>>> normal(0, 1, 3)
array([-0.03425693,  0.06035959,  0.45511859])
```

NumPy入門

- 統計量の計算(平均、分散、標準偏差)
  - mean()
  - var()
  - std()

```
>>> seed(3)
>>> y = normal(0, 1, 100)
>>> np.mean(y)
-0.10863707440606224
>>> np.var(y)
1.132081888283007
>>> np.sqrt(np.var(y))
1.0639933685333791
>>> np.std(y)
1.0639933685333791
```

Matplotlib入門

- 準備

```
>>> import matplotlib.pyplot as plt
```

- プロット(散布図)
  - scatter()

```
>>> x = normal(0, 1, 100)
>>> y = normal(0, 1, 100)
>>> plt.scatter(x, y)
>>> plt.show()
```

Matplotlib入門

- 出力の保存
  - savefig()

```
>>> plt.scatter(x, y, 10)
>>> plt.savefig("fig1.png")
```

Matplotlib入門

- プロット(折れ線グラフ)
  - plot()

```
>>> x = np.linspace(0, 2, 100)
>>>
>>> plt.plot(x, x, label='linear')
>>> plt.plot(x, x**2, label='quadratic')
>>> plt.plot(x, x**3, label='cubic')
>>>
>>> plt.xlabel('x label')
>>> plt.ylabel('y label')
>>>
>>> plt.title("Simple Plot")
>>>
>>> plt.legend()
>>>
>>> plt.show()
```

https://matplotlib.org/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py

NumPy入門

- 数の系列の作成
  - arange()
  - linspace()

```
>>> x = np.arange(1, 11)
>>> x
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
>>> import math
>>> x = np.linspace(-math.pi, math.pi, 50)
>>> x
```

Matplotlib入門

- 等高線プロット
  - contour()

```
>>> import matplotlib.cm as cm
>>> import matplotlib.mlab as mlab
>>> y = x
>>> X, Y = np.meshgrid(x, y)
>>> f = np.cos(Y) / (1 + np.square(X))
>>> CS = plt.contour(X, Y, f)
>>> plt.show()
```

統計的学習の基礎概念

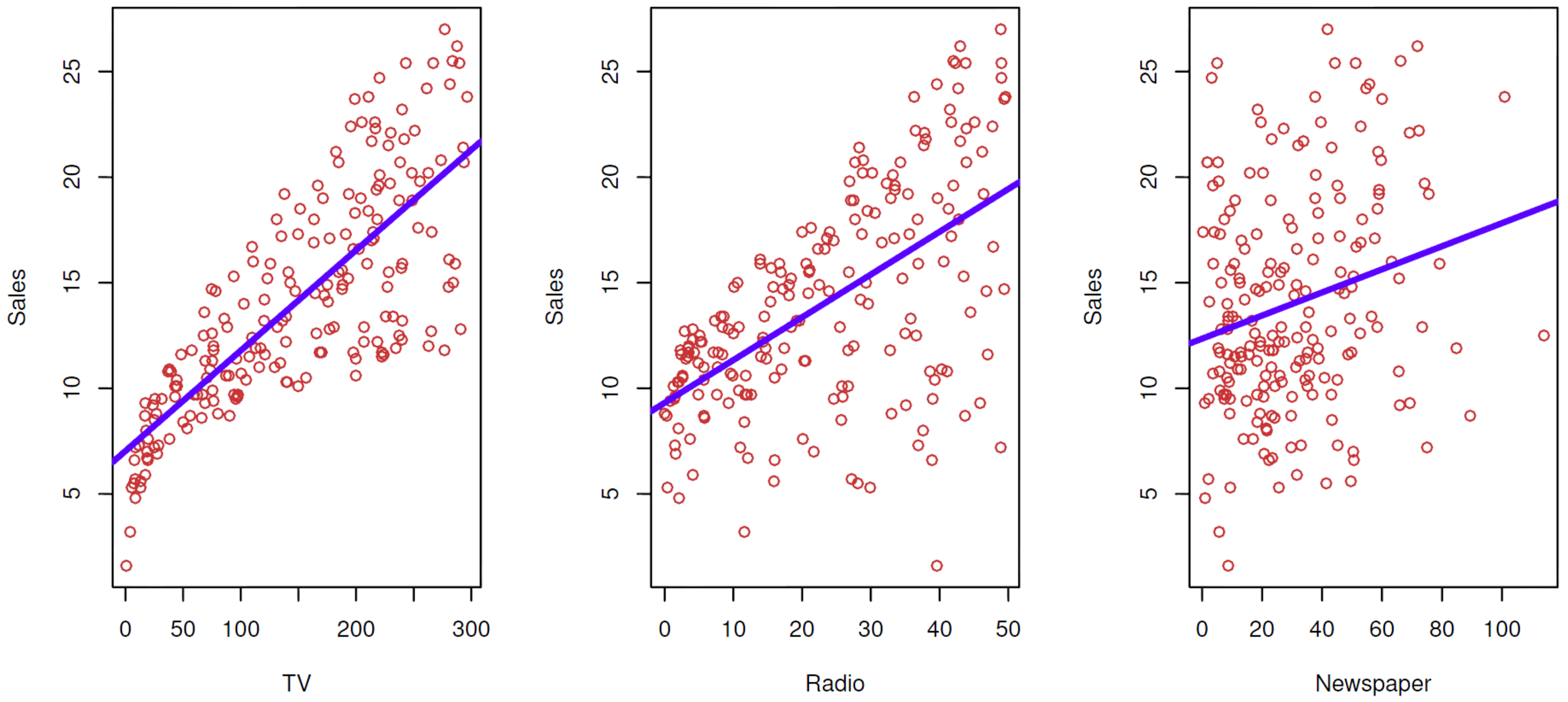
統計的学習とは

- Advertising データセット

	メディアでの広告宣伝費			製品の売上
	TV	Radio	Newspaper	Sales
市場1	230.1	37.8	69.2	22.1
市場2	44.5	39.3	45.1	10.4
市場3	17.2	45.9	69.3	9.3
市場4	151.5	41.3	58.5	18.5
:	:	:	:	:
市場200	232.1	8.6	8.7	13.4

- 売上を増やすにはどうしたら良いか？
  - どのメディアでの宣伝を増やすべきか？

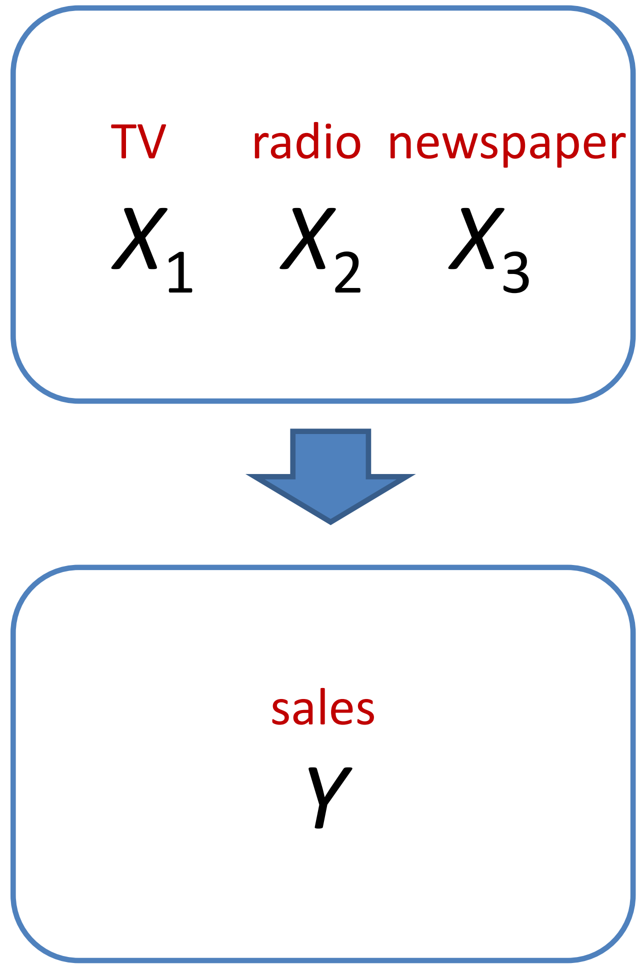
統計的学習とは



※青線は最小二乗法によって求めた回帰直線

統計的学習とは

- 入力: TV, radio, newspaper
  - 呼び方
    - 入力変数(input variable)
    - 独立変数(independent variable)
    - 予測変数(predictor)
    - 説明変数(explanatory variable)
    - 特徴量／素性(そせい)(feature)
    - etc
- 出力: sales
  - 呼び方
    - 出力変数(output variable)
    - 従属変数(dependent variable)
    - 応答変数(response)
    - etc



統計的学習とは

- 入力 $X = (X_1, X_2, \ldots, X_p)$ と出力 Y の間に何らかの関係が存在

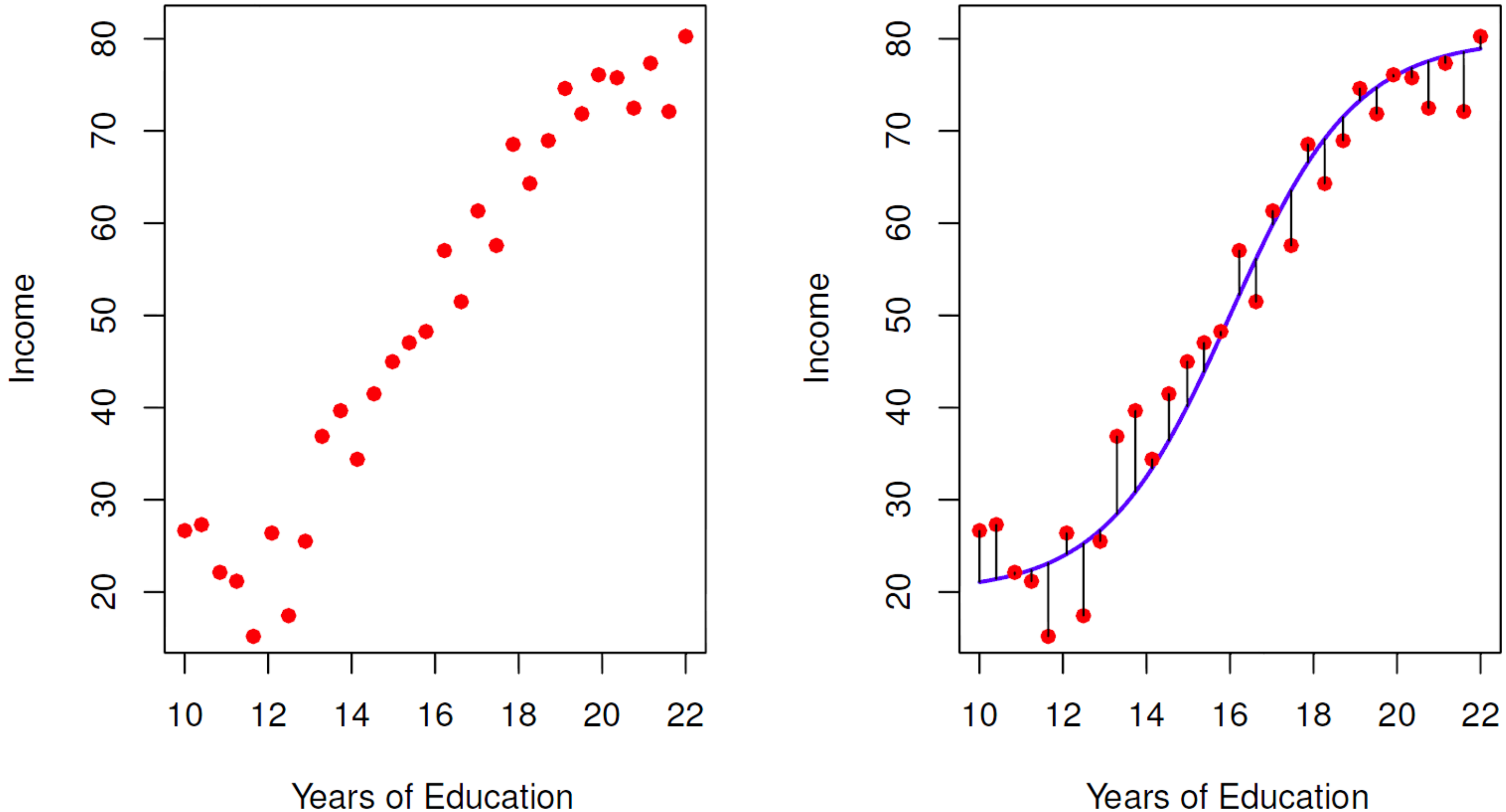
$$Y = f\left(X\right) + \varepsilon$$

- f は $X_1, X_2, \ldots, X_p$ の関数
  - X と Y の系統的な関係を表す
  - データから f を推定したい
- ε は確率的な誤差（error）
  - 平均はゼロ

統計的学習とは

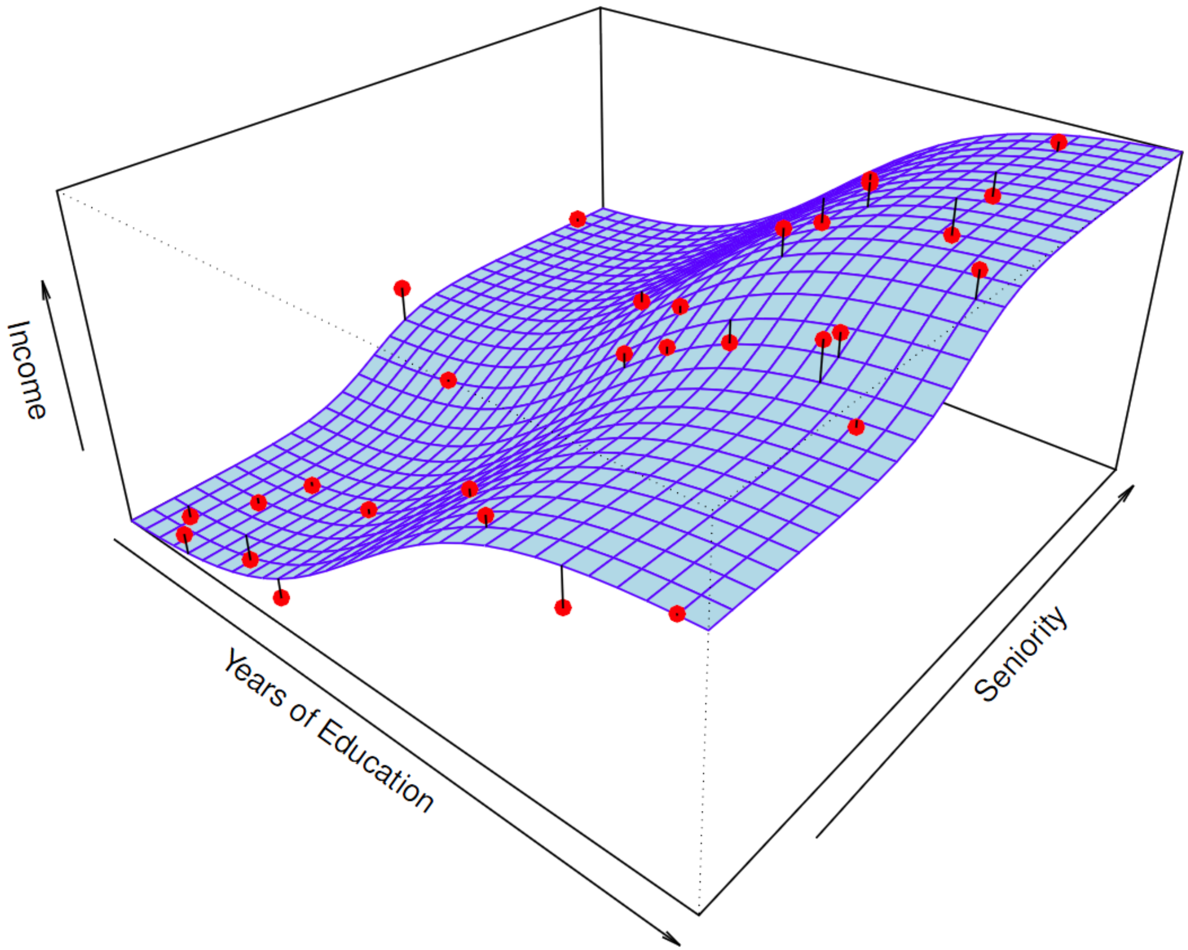
- Income データセット

青い曲線が f（実際は未知）
黒い縦線が ε



統計的学習とは

- 入力変数が2つの場合



なぜ f を推定したいのか

- 目的1：予測（prediction）
  - Y が容易に得られない場合に X から予測する

$$\hat{Y} = \hat{f}(X)$$

Y の推定値　　推定された f

- 例）薬の副作用の予測
  - $X_1, X_2, \ldots, X_p$：患者の血液の様々な特性値
  - Y：ある薬に対して重篤な副作用が起きるリスク

なぜ f を推定したいのか

- 予測が目的の場合
  - 推定が正確でありさえすれば良い
  - $\hat{f}$ はブラックボックスでも構わない

- 予測の精度
  - 削減可能な誤差（reducible error）
    - $\hat{f}$ が適切でないために生じる誤差
  - 削減不可能な誤差（irreducible error）
    - ε によって生じる誤差
      - 予測に有用な情報が入力として使われていない
      - 測定不能なゆらぎの影響

なぜ f を推定したいのか

- 予測誤差の大きさ
  - さしあたり $\hat{f}$ と X は固定されているものとすると

$$
\underset{\text{期待値}}{E}\left[\left(Y-\hat{Y}\right)^2\right] = E\left[\left(f(X)+\varepsilon-\hat{f}(X)\right)^2\right]
$$

$$
= E\left[\left(f(X)-\hat{f}(X)\right)^2 + 2\left(f(X)-\hat{f}(X)\right)\varepsilon + \varepsilon^2\right]
$$

$$
= E\left[\left(f(X)-\hat{f}(X)\right)^2\right] + 2E\left[\left(f(X)-\hat{f}(X)\right)\varepsilon\right] + E\left[\varepsilon^2\right]
$$

$$
= \left(f(X)-\hat{f}(X)\right)^2 E[1] + 2f(X)E[\varepsilon] - 2\hat{f}(X)E[\varepsilon] + \underset{\varepsilon\,\text{の分散}}{\mathrm{Var}(\varepsilon)}
$$

$$
= \underbrace{\left(f(X)-\hat{f}(X)\right)^2}_{\text{削減可能な誤差}} + \underbrace{\mathrm{Var}(\varepsilon)}_{\text{削減不可能な誤差}}
$$

なぜ f を推定したいのか

- 目的2：推論（inference）
  - $X_1, X_2, \ldots, X_p$ が変化したときに Y がどのように変わるかを知りたい
  - 具体例
    - どの入力変数が Y に関係しているのか？
    - それぞれの入力変数が出力とどのように関係しているのか？
      - 例）X_2 が大きくなるの Y はどうなるのか？
    - X と Y の関係は線形モデルで表現できるのか？

なぜ f を推定したいのか

- 推論が目的であるような場合の例
  - Advertising データセット
    - どのメディアが売上向上に貢献するのか
    - どのメディアが売上向上に最も貢献するのか
    - TV の宣伝費をある値だけ増やした時にどれだけ売上が増えるのか

- 予測と推論の両方が目的であるような場合の例
  - 不動産価格の推定
    - 家の大きさ、設備、駅までの距離、治安などから住宅の価格を予測
    - 例）設備を良くすることでどれだけ家の価値が上がるのか？（推論）
    - 例）ある住宅が高すぎるのか安すぎるのかを知りたい（予測）

なぜ f を推定したいのか

- 目的によって適切なモデルは異なる
  - 線形モデル
    - 推定した関数の中身の解釈が比較的容易なため推論に使いやすい
    - X と Y の関係が複雑・非線形な場合、高い予測精度が得られない
  - 非線形モデル
    - 対象が複雑な現象であっても高い予測精度が得られる可能性がある
    - モデルの中身の解釈が難しく、推論には使いにくい

どのようにして f を推定するのか

- 学習データ（training data）
  - n 個の観測データ

$$\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$$

$$x_i = \begin{pmatrix} x_{i1} \\ x_{i2} \\ \vdots \\ x_{ip} \end{pmatrix}$$

- 統計的推定
  - 学習データから、個々の観測データ (X,Y) に対して

$$Y \approx \hat{f}(X)$$

となるような $\hat{f}$ を推定

どのようにして f を推定するのか

- パラメトリックな手法（parametric methods）
  - f の関数形に対して何らかの仮定をおく

    例）線形モデル（linear model）

$$f(X) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

  - 係数 $\beta_0, \beta_1, \beta_2, \ldots, \beta_p$ を

$$Y \approx \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p$$

    となるように定める（学習、フィッティング）

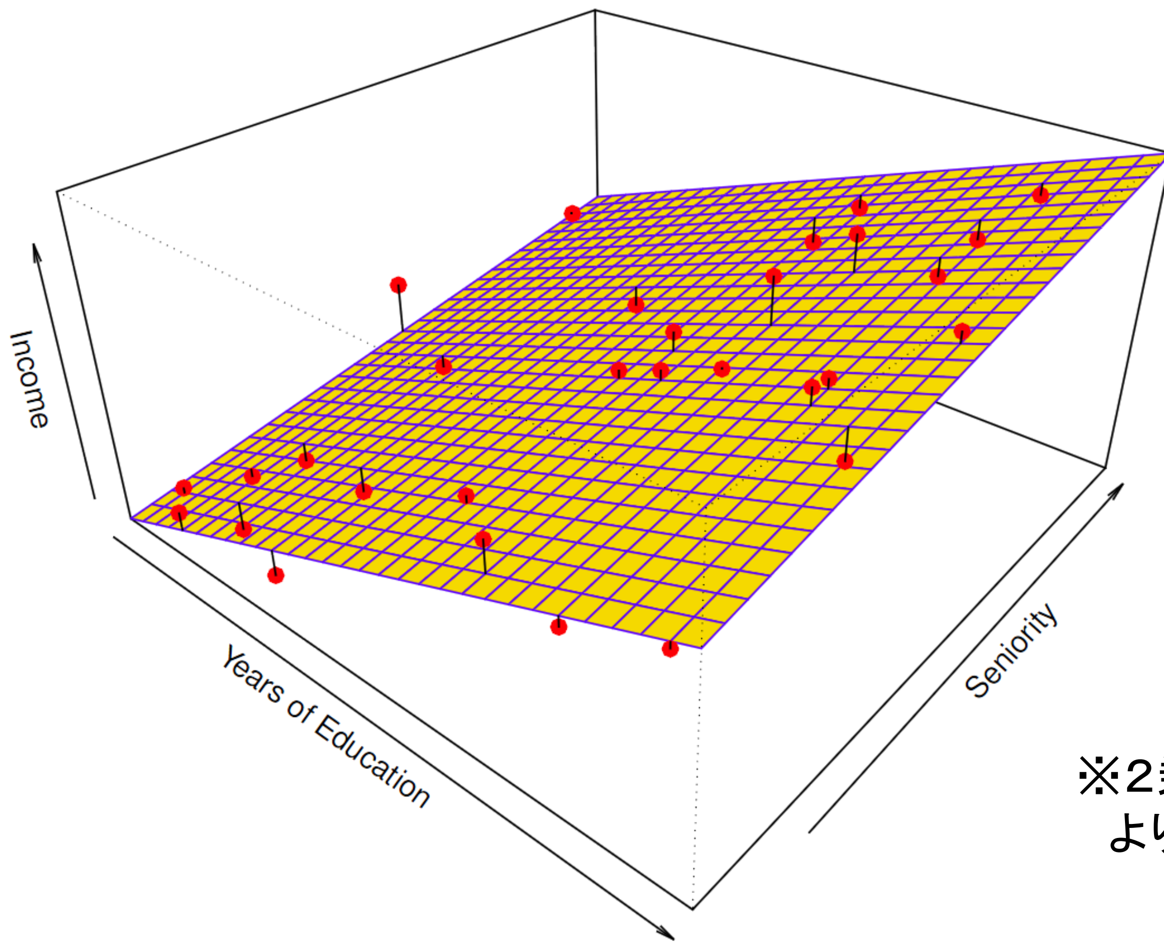
どのようにして f を推定するのか

- パラメトリックな手法（parametric methods）
  - 利点
    - f を推定する問題が、$p+1$ 個のパラメータの値を求めるという簡単な問題に帰着された
  - 欠点
    - 仮定したモデルが真の f を表現できないかもしれない
    - 表現力の大きなモデルを仮定すればその問題は軽減できるが、学習データ中のノイズに過剰に適合する過学習（overfitting）の問題が起こる

どのようにして f を推定するのか

- 例）線形モデルによる推定（Income データセット）

$$\texttt{income} \approx \beta_0 + \beta_1 \times \texttt{education} + \beta_2 \times \texttt{seniority}$$

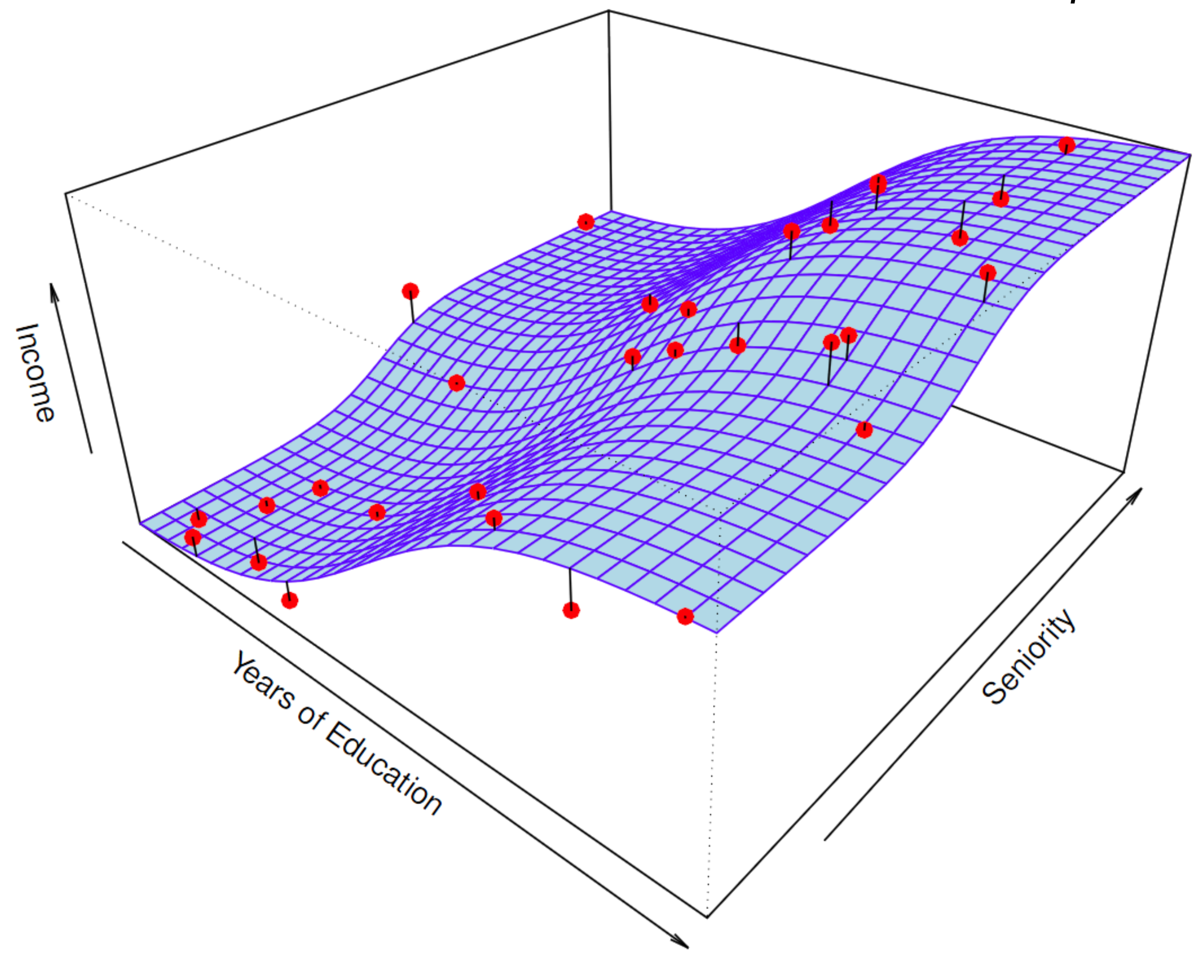


※2乗誤差の最小化により $\beta_0, \beta_1, \beta_2$ を決定

どのようにして f を推定するのか

- 真の f（再掲）

※ Income データセットは人工的に作ったデータなので真の f がわかっている

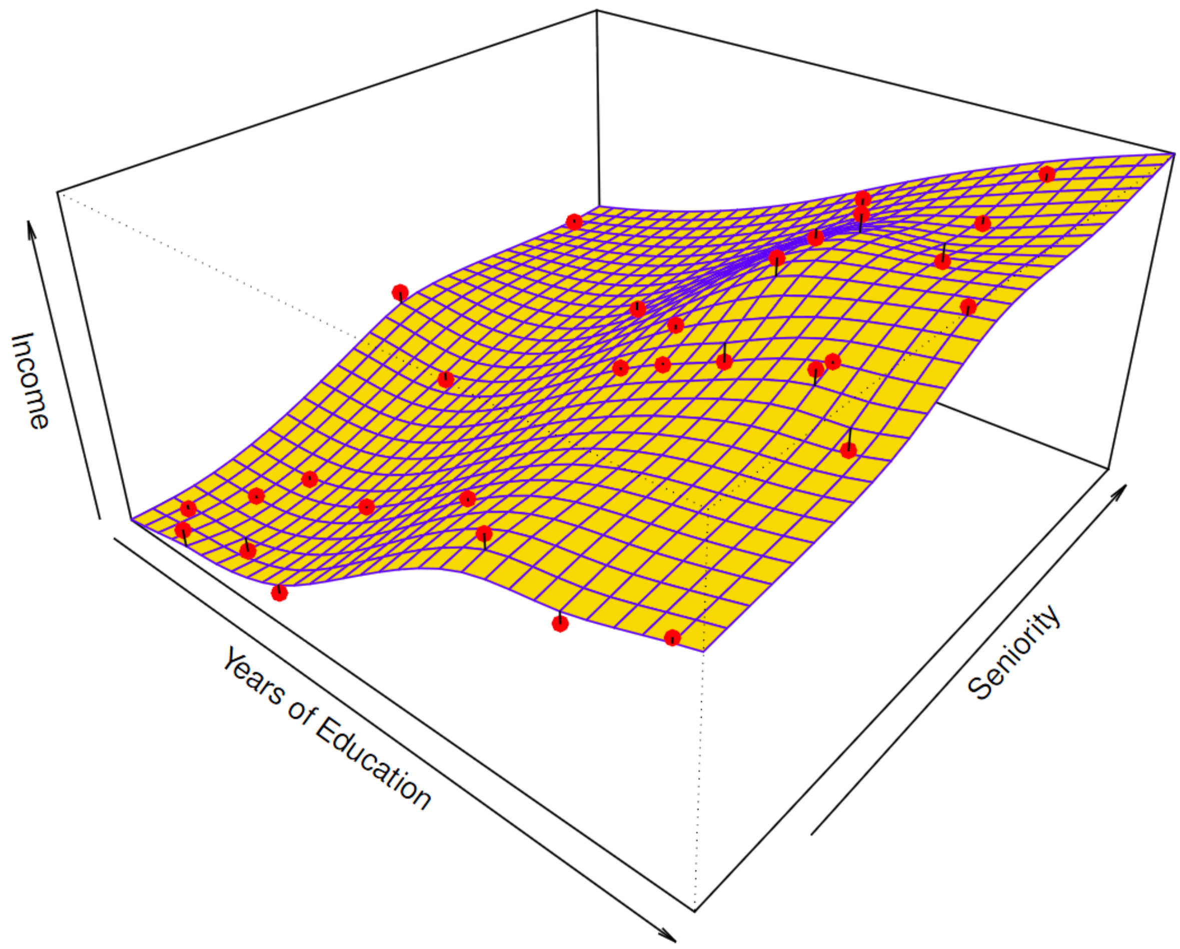


どのようにして f を推定するのか

- ノンパラメトリックな手法（non-parametric methods）
  - f の関数形に関して明示的な仮定を置かない
  - 利点
    - 複雑で多様な f を表現できる可能性がある
  - 欠点
    - 高精度な推定のためには大量の観測データが必要

どのようにして f を推定するのか

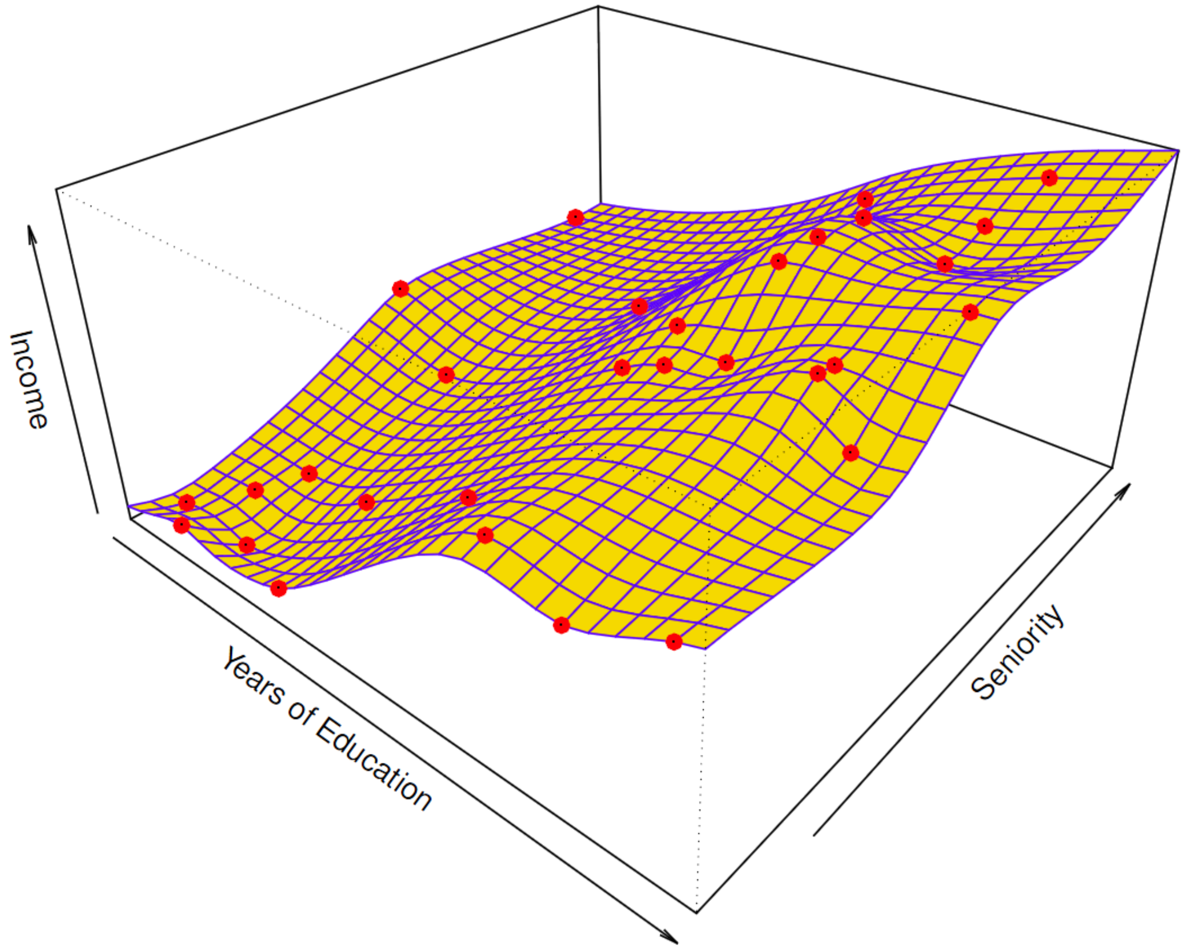
- 例）thin-plate spline による推定（Income データセット）



どのようにして f を推定するのか

- 例）thin-plate spline による推定
  - 曲面の滑らかさに関するペナルティを甘くした場合

→ 過学習



どのようにして f を推定するのか

- 予測精度と解釈性のトレードオフ
  - 表現力の低いモデル
    - 複雑な形の f をうまく近似できない
    - Y と $X_1, X_2, \ldots, X_\mathrm{p}$ の関係がわかりやすい
  - 表現力の高いモデル
    - 複雑で多様な f に対応できる
    - 解釈性が低い

どのようにして f を推定するのか

- 予測精度と解釈性のトレードオフ