

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

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

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

- 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)

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

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

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

となるように定める (学習、フィッティング)

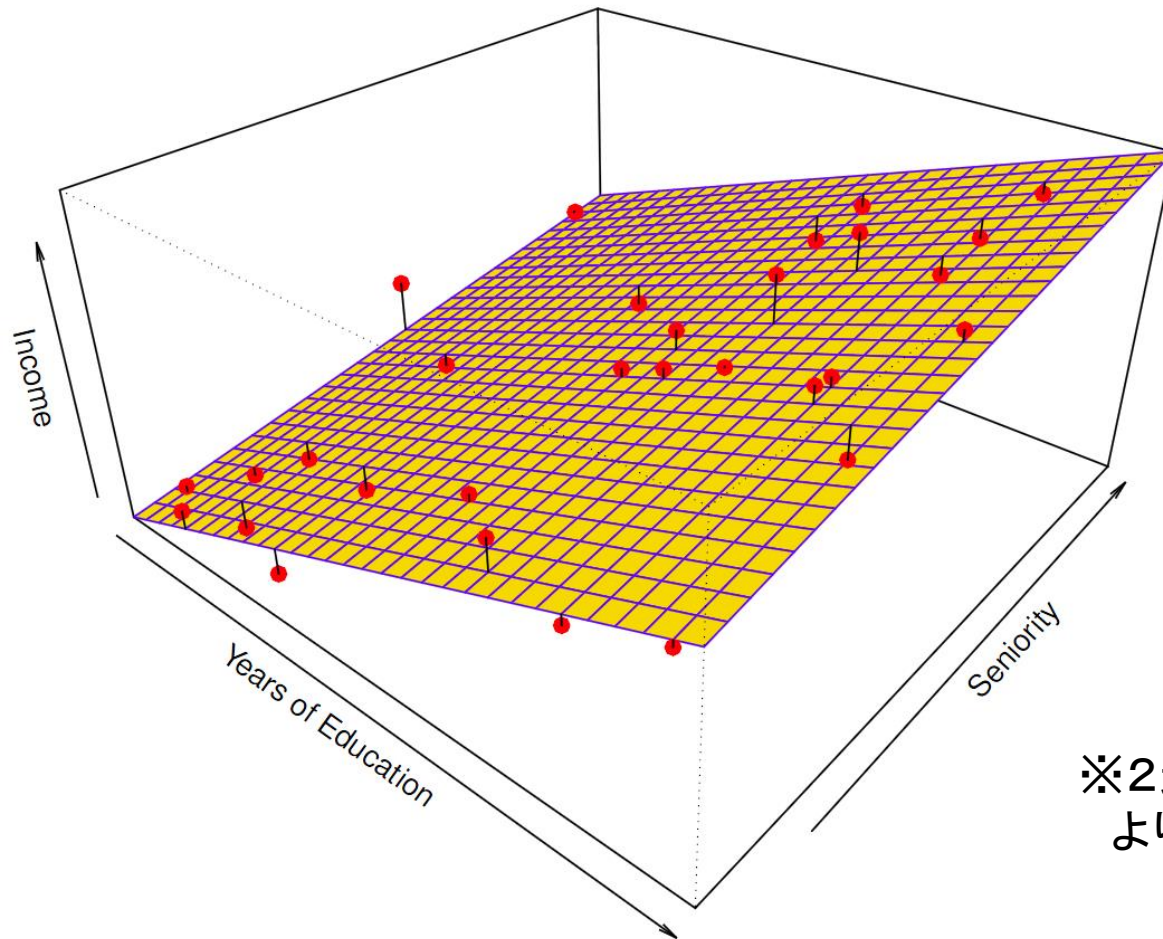
どのようにして f を推定するのか

- 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)

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

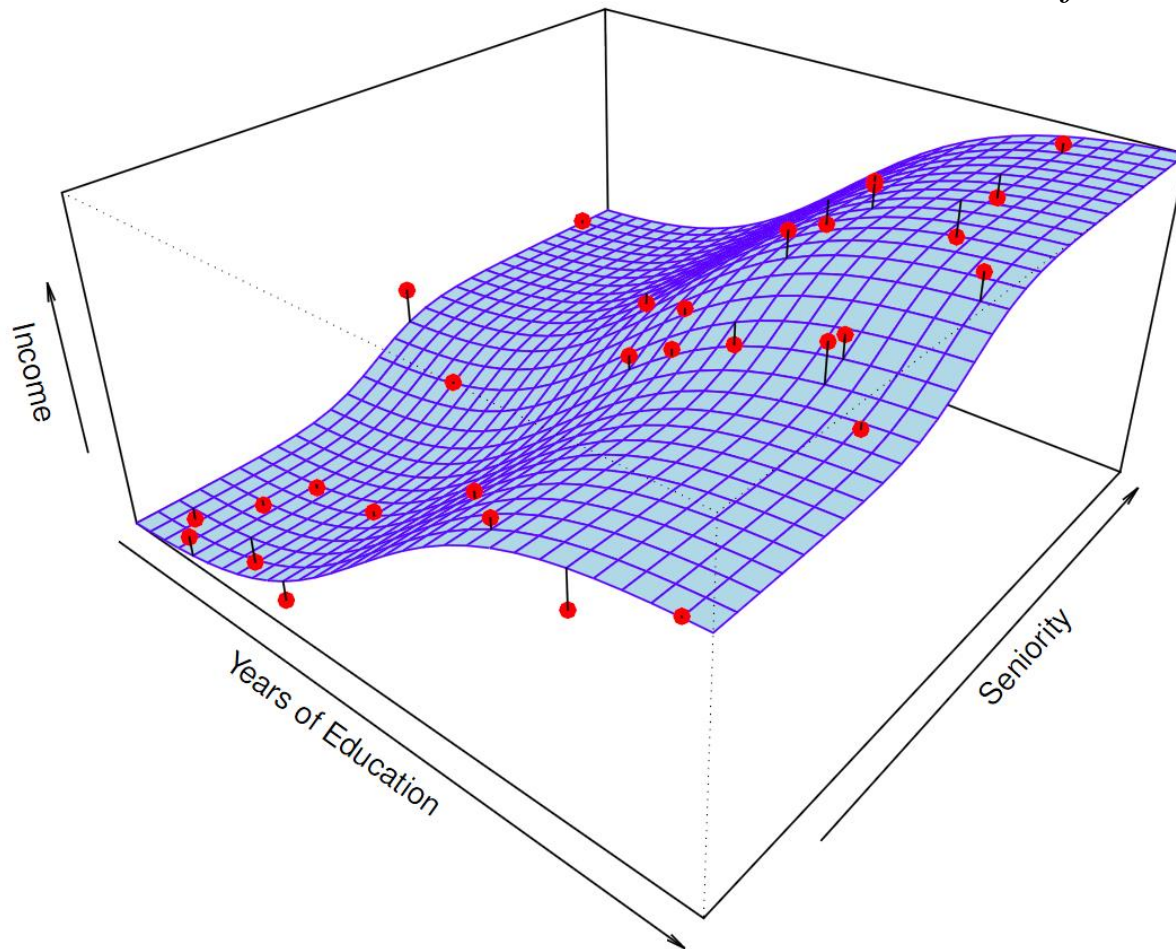


※2乗誤差の最小化により $\beta_0, \beta_1, \beta_2$ を決定

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

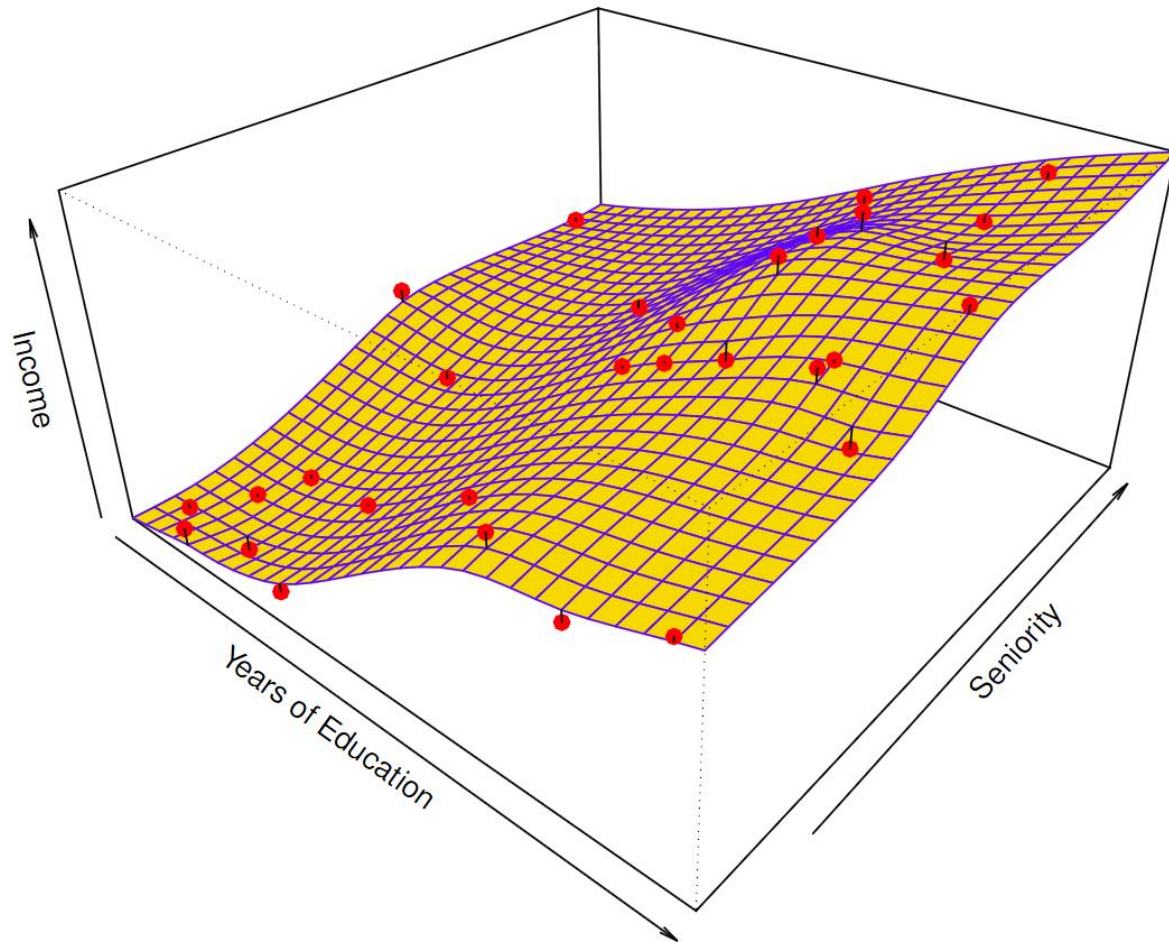


どのようにして f を推定するのか

- 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

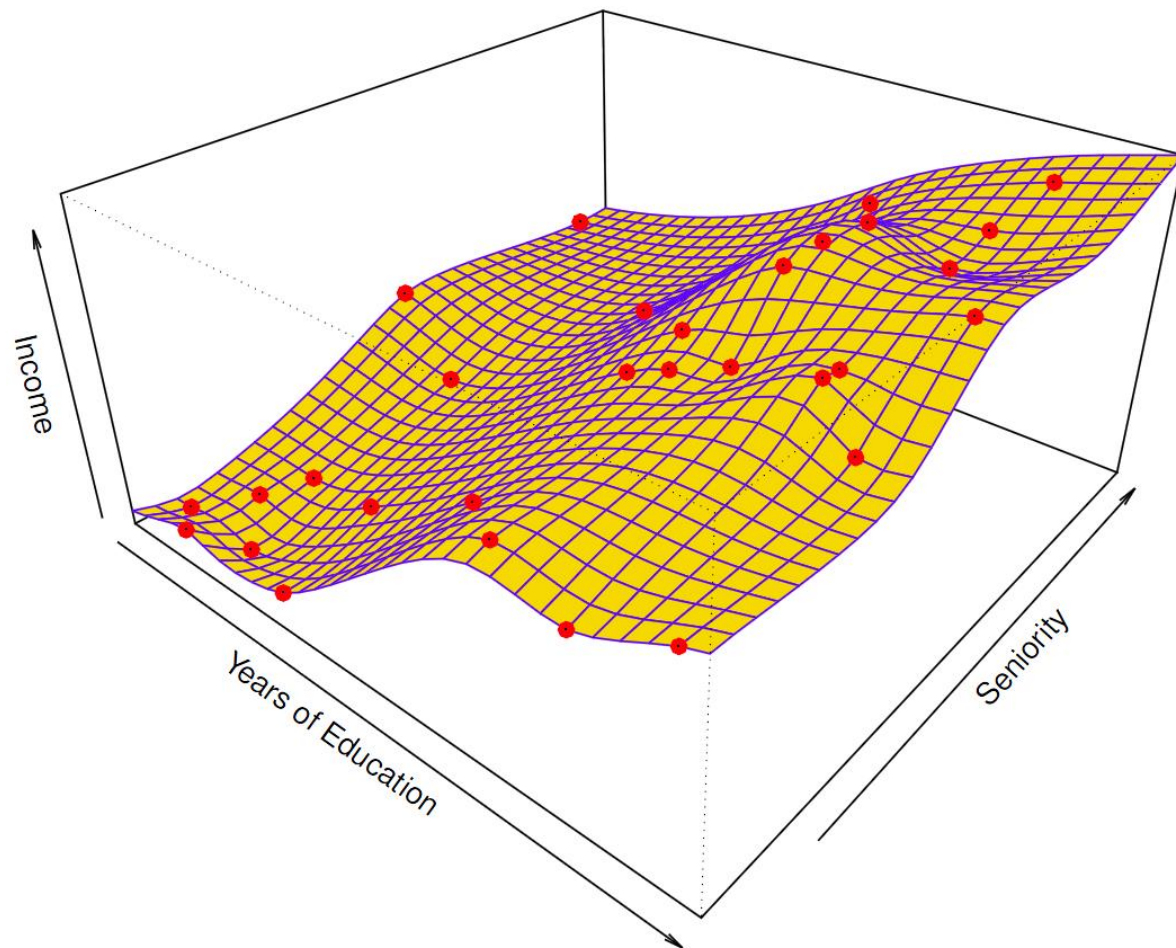
どのようにして f を推定するのか

- 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



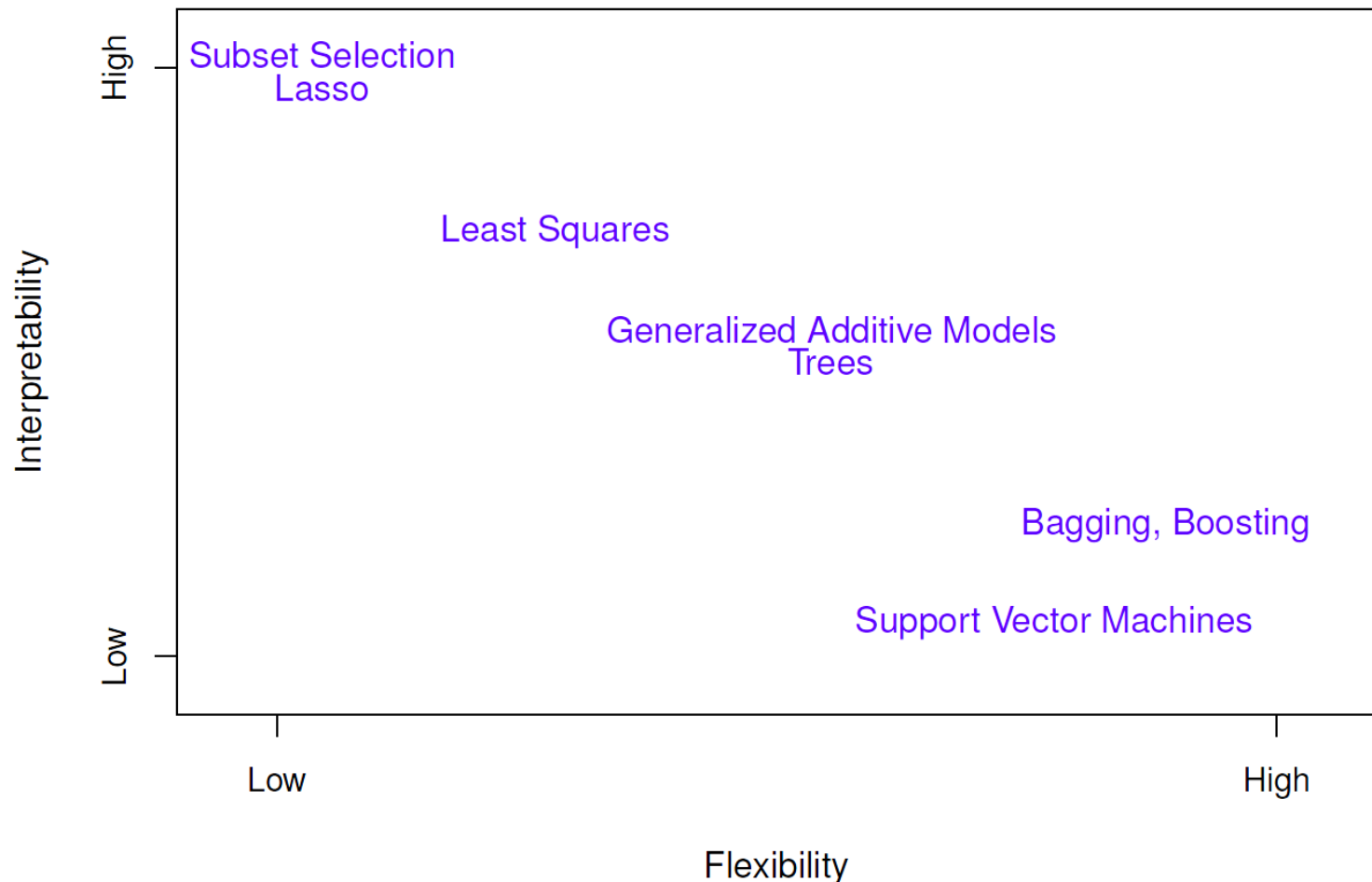
→ 過学習

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, \dots, X_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

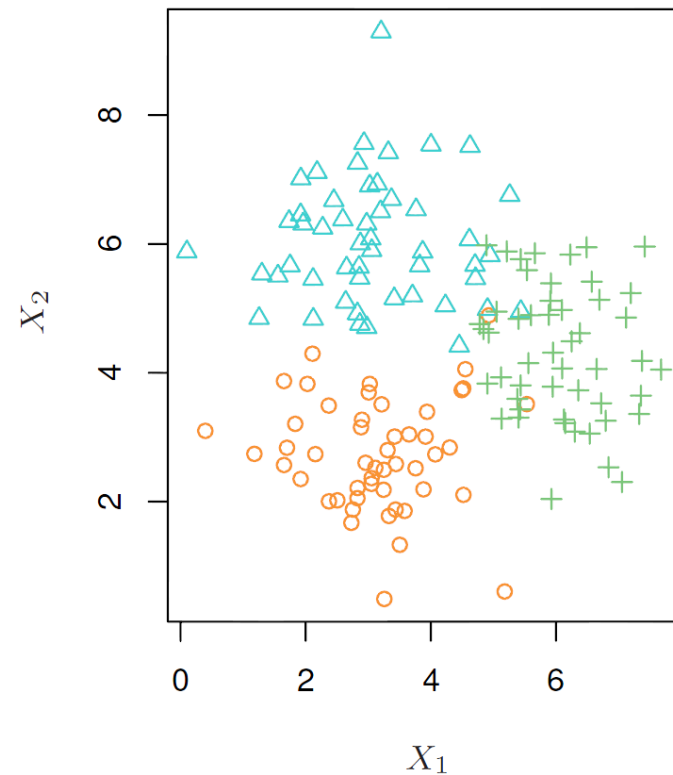
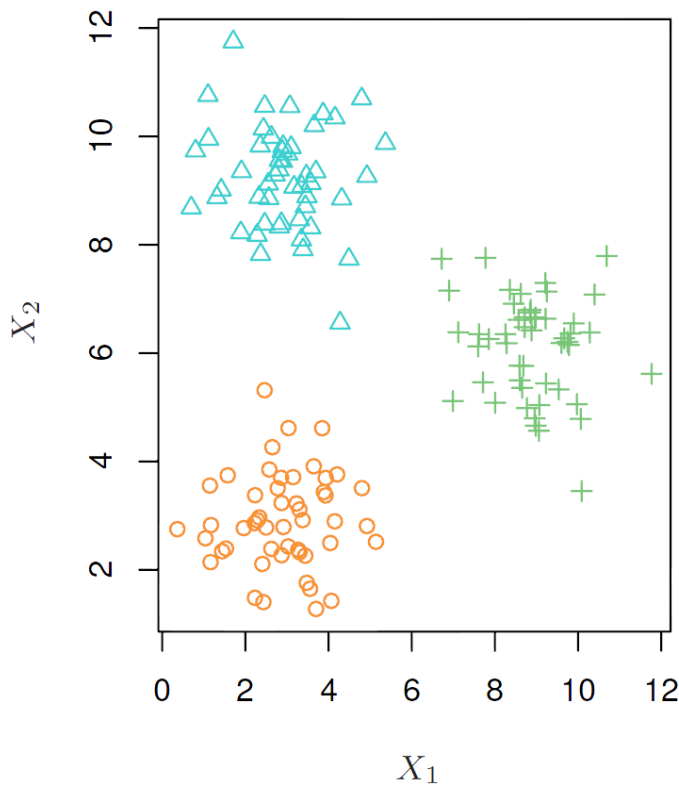


教師付き学習と教師なし学習

- 教師付き学習 (supervised learning)
 - 各観測データ $x_i, i = 1, \dots, n$ に対して応答 y_i がある
 - 目的
 - X から Y を予測
 - X と Y の関係を推論
 - 例) 顧客の属性から購買額を予測
- 教師なし学習 (unsupervised learning)
 - $x_i, i = 1, \dots, n$ はあるが y_i は無い
 - 目的
 - 変数や観測データ間の関係を明らかにする
 - 例) 顧客の属性から顧客をグループ分け (クラスタリング)

教師付き学習と教師なし学習

- クラスタリングの例



- 色は真のグループを表す(実際にクラスタリングする際は未知)
- 実際のデータはもっと高次元なので大変

回帰と分類

- 変数
 - 量的変数 (quantitative variable)
 - 実数値をとる変数
 - 例) 年齢、身長、収入、株価、etc
 - 質的変数 (qualitative variable)
 - K 個の異なるクラス(カテゴリ)のうちどれかの値をとる変数
 - 例) 性別、購入した製品、Yes/No、病気の種類、etc
- 教師付き学習の問題は大きく2つに分けられる
 - 回帰問題 (regression problem)
 - 予測対象(出力)が量的変数
 - 分類問題 (classification problem)
 - 予測対象が質的変数

モデルの選択

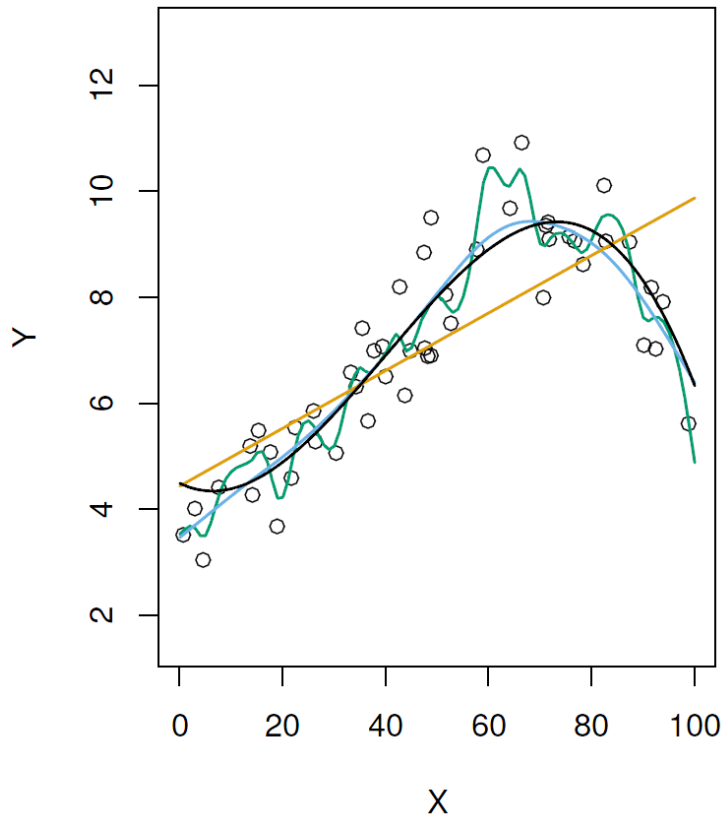
- モデルの精度(回帰問題の場合)
 - 平均二乗誤差(mean squared error, MSE)

$$MSE = \frac{1}{n} \sum_{i=1}^n \left(y_i - \hat{f}(x_i) \right)^2$$

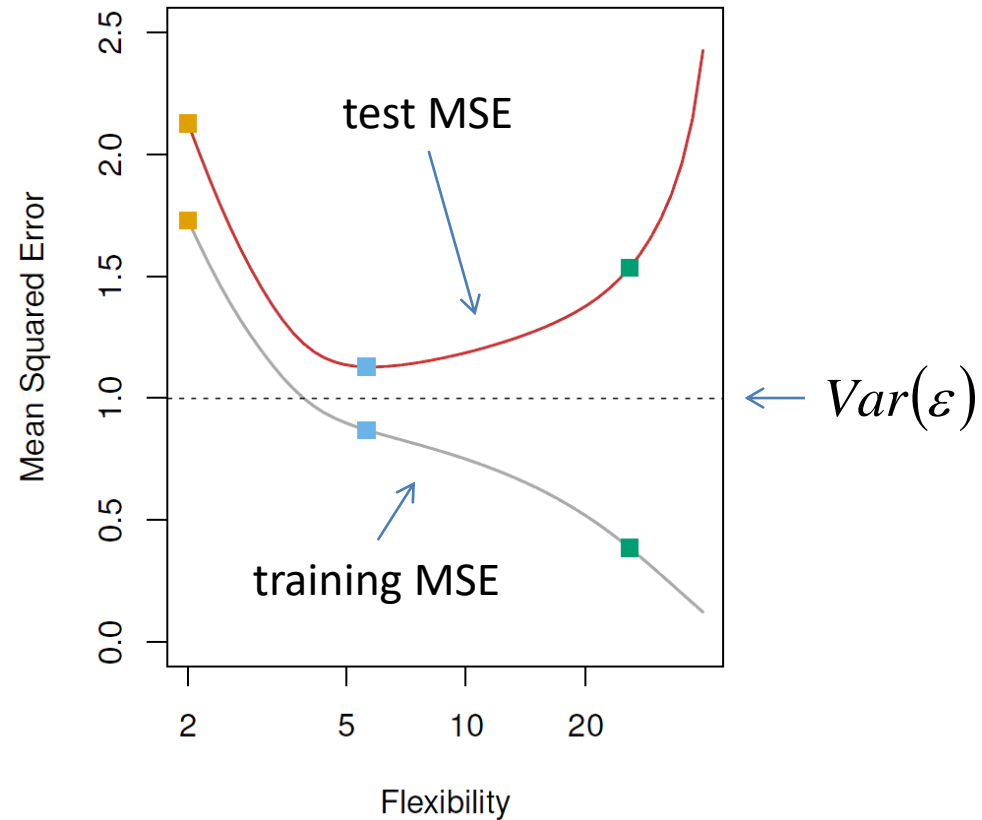
- この式は学習データでの誤差(training MSE)だが、本当に最小化したいのは、未知のデータ(テストデータ)での誤差(test MSE)

$$\text{Ave} \left(\left(y_0 - \hat{f}(x_0) \right)^2 \right)$$

モデルの選択



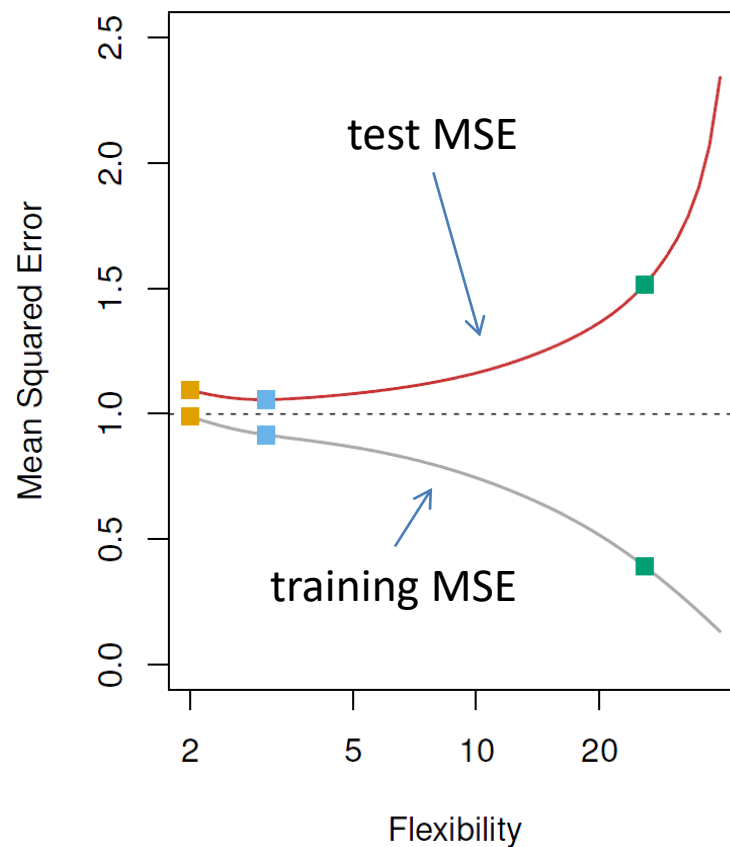
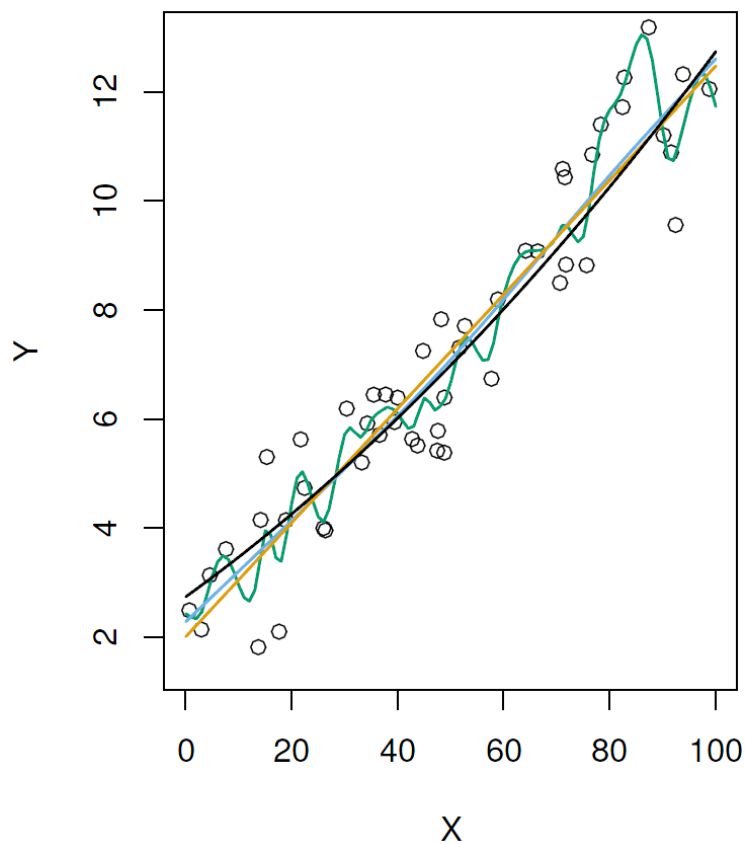
黒: 真の f
オレンジ: 線形回帰
青、緑: 平滑化スプライン



学習データでのMSEが小さいほど
テスト時のMSEが小さくなるとは限
らない

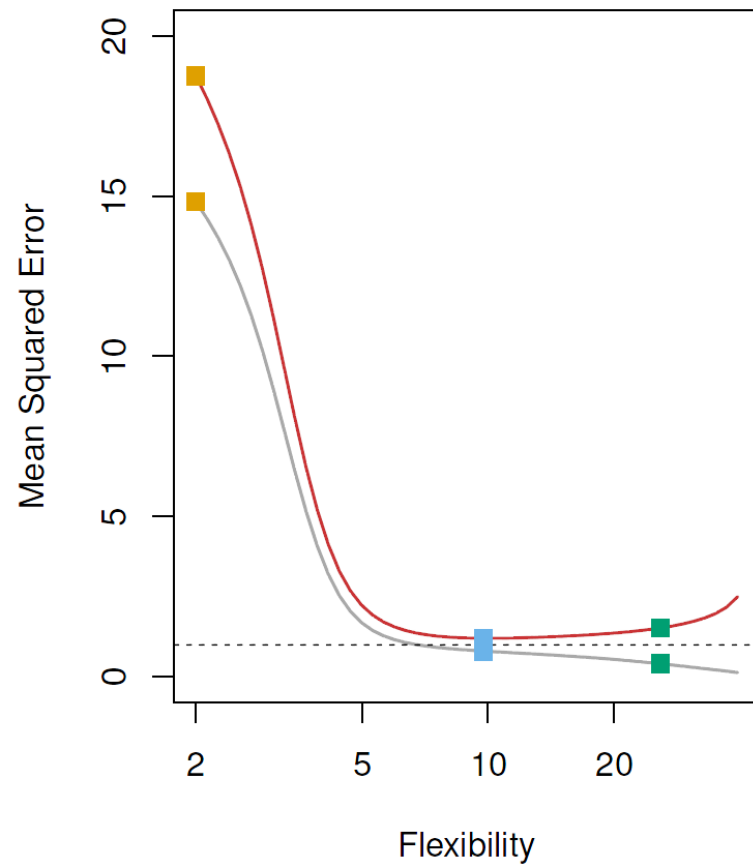
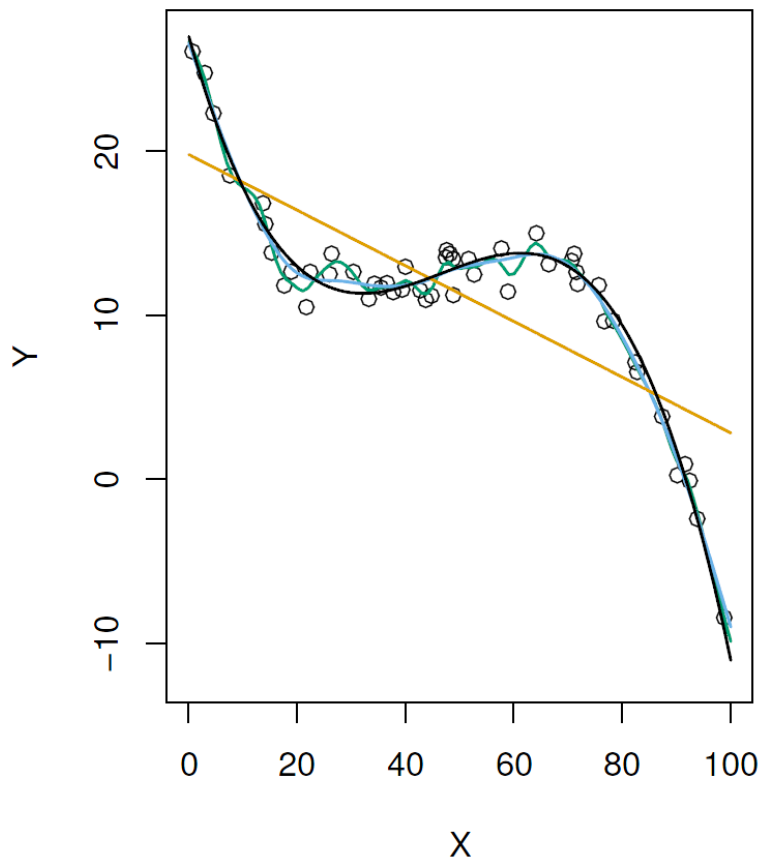
モデルの選択

- 真の f がほぼ直線の場合



モデルの選択

- 真の f の非線形性が強い場合



Python実習

- 準備

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> import math
>>> x = np.linspace(-math.pi, math.pi, 50)
>>> y = x
>>> X, Y = np.meshgrid(x, y)
>>> f = np.cos(Y) / (1 + np.square(X))
```

参考) ISL_python

https://github.com/qx0731/ISL_python

Python実習

- ヒートマップ

```
>>> fa = (f - f.T) / 2
>>> plt.imshow(fa, extent=(x[0], x[-1], y[0], y[-1]))
>>> plt.show()
```

- 3次元プロット

```
>>> from mpl_toolkits.mplot3d import axes3d
>>> fig = plt.figure()
>>> ax = fig.add_subplot(111, projection='3d')
>>> ax.plot_wireframe(X, Y, fa)
>>> plt.show()
```

Python実習

- 行列の要素

- [行-1, 列-1] ※indexがゼロから始まるので

```
>>> A = np.arange(1, 17, 1)
>>> A = A.reshape(4, 4)
>>> A = A.T
>>> A
array([[ 1,  5,  9, 13],
       [ 2,  6, 10, 14],
       [ 3,  7, 11, 15],
       [ 4,  8, 12, 16]])
>>> A[1, 2]
10
```

Python実習

- 行列の一部
 - [抽出する行番号(の並び), 抽出する列番号(の並び)]

```
>>> A[0:3, 1:4]
array([[ 5,  9, 13],
       [ 6, 10, 14],
       [ 7, 11, 15]])
>>> A[[0],[2], [1, 3]]
array([[ 5, 13],
       [ 7, 15]])
```

```
>>> A[0:2]
array([[ 1,  5,  9, 13],
       [ 2,  6, 10, 14]])
>>> A[:, 0:2]
array([[1, 5],
       [2, 6],
       [3, 7],
       [4, 8]])
```

Python実習

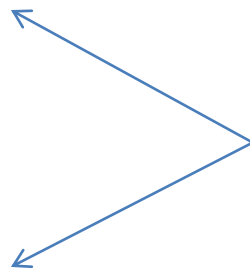
- 行列の一部
 - 特定の行

```
>>> A[1]
array([ 2,  6, 10, 14])
```

- 特定の列

```
>>> A[:, 1]
array([5,  6,  7,  8])
```

※どちらも得られるのはベクトル



Python実習

- 行列の一部
 - 特定の行(列)を除きたい場合

```
>>> ind = np.ones(4, bool)
>>> ind
array([ True,  True,  True,  True])
>>> ind[[0, 2]] = False
>>> ind
array([False,  True, False,  True])
>>> A[ind]
array([[ 2,  6, 10, 14],
       [ 4,  8, 12, 16]])
>>> A[:, ind]
...
```

Python実習

- 外部データ(csv形式)の読み込み
 - ダウンロード
 - サンプルファイル Auto.csv
 - <http://www-bcf.usc.edu/~gareth/ISL/Auto.csv>
 - 上記ファイルをホームディレクトリに保存(Windowsの場合)
 - ホームディレクトリに移動

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

- `pandas.read_csv()` 関数でデータフレームとして読み込み

```
>>> import pandas as pd
>>> Auto = pd.read_csv('Auto.csv', header=0, na_values='?')
>>> Auto
...
```

Auto.dat

mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
18	8	307	130	3504	12	70	1	chevrolet chevelle malibu
15	8	350	165	3693	11.5	70	1	buick skylark 320
18	8	318	150	3436	11	70	1	plymouth satellite
16	8	304	150	3433	12	70	1	amc rebel sst
17	8	302	140	3449	10.5	70	1	ford torino
:	:	:	:	:	:	:	:	

Python実習

- データの参照

- `iloc[]`: 整数で行、列を指定 (integer-location)

```
>>> Auto.iloc[32]
mpg                25
cylinders           4
displacement       98
horsepower          NaN ← 欠損値
weight            2046
acceleration       19
year              71
origin             1
name              ford pinto
Name: 32, dtype: object
```

Python実習

- 欠損値を含むデータの削除

- `dropna()`

```
>>> Auto = Auto.dropna()  
>>> Auto.shape  
(392, 9)
```

- データフレームの列名

- `list()`

```
>>> list(Auto)  
['mpg', 'cylinders', 'displacement', 'horsepower',  
'weight', 'acceleration', 'year', 'origin', 'name']
```

Python実習

- データの参照

```
>>> Auto.iloc[0:3, 1:3]
cylinders    displacement
0             8          307.0
1             8          350.0
2             8          318.0
>>> Auto[0:4]
...
```

Python実習

- データの参照
 - データフレーム名.列名

```
>>> plt.plot(Auto.cylinders, Auto.mpg, 'ro')  
>>> plt.show()
```

- 行列と同じような方法

```
>>> plt.plot(Auto.iloc[:, 1], Auto.iloc[:, 0], 'ro')
```


Python実習

- データの統計情報

- describe()

```
>>> Auto.describe()  
...  
>>> Auto.describe(include= 'all')  
...
```

- ヒストグラム

- hist()

```
>>> Auto.hist(column = ['cylinders'])  
>>> plt.show()
```

モデルの精度

- The bias-variance trade-off
 - テストMSEの期待値は3つの項の和に分解できる
 - $\hat{f}(x_0)$ のバリエーション(分散)
 - $\hat{f}(x_0)$ のバイアスの二乗
 - ϵ の分散

$$E \left[\left(y_0 - \hat{f}(x_0) \right)^2 \right] = \text{Var} \left(\hat{f}(x_0) \right) + \left[\text{Bias} \left(\hat{f}(x_0) \right) \right]^2 + \text{Var}(\epsilon)$$

$$\text{Var} \left(\hat{f}(x_0) \right) = E \left[\left(\hat{f}(x_0) - E \left(\hat{f}(x_0) \right) \right)^2 \right] = E \left[\hat{f}(x_0)^2 \right] - E \left[\hat{f}(x_0) \right]^2$$

$$\text{Bias} \left(\hat{f}(x_0) \right) = E \left[\hat{f}(x_0) - f(x_0) \right]$$

学習データが変化すること
による $\hat{f}(x_0)$ のゆらぎ

参考

Bias-variance trade-off の導出

$$E \left[\left(y_0 - \hat{f}(x_0) \right)^2 \right] = E[y_0^2 + \hat{f}(x_0)^2 - 2y_0\hat{f}(x_0)]$$

$$= E[y_0^2] + E[\hat{f}(x_0)^2] - 2E[y_0\hat{f}(x_0)]$$

$$\text{※ } E[X^2] = \text{Var}[X] + E[X]^2 \text{ より}$$

$$= \text{Var}[y_0] + E[y_0]^2 + \text{Var}[\hat{f}(x_0)] + E[\hat{f}(x_0)]^2 - 2E[y_0\hat{f}(x_0)]$$

$$\text{※ } y_0 = f(x_0) + \epsilon \text{ より}$$

$$= \text{Var}[f(x_0) + \epsilon] + E[f(x_0) + \epsilon]^2 + \text{Var}[\hat{f}(x_0)] + E[\hat{f}(x_0)]^2 - 2E[(f(x_0) + \epsilon)\hat{f}(x_0)]$$

$$\text{※ } f(x_0) \text{ は定数なので}$$

$$= \text{Var}[f(x_0)] + \text{Var}[\epsilon] + E[f(x_0)]^2 + \text{Var}[\hat{f}(x_0)] + E[\hat{f}(x_0)]^2 - 2f(x_0)E[\hat{f}(x_0)]$$

$$= \text{Var}[\epsilon] + \text{Var}[\hat{f}(x_0)] + f(x_0)^2 - 2f(x_0)E[\hat{f}(x_0)] + E[\hat{f}(x_0)]^2$$

$$= \text{Var}[\epsilon] + \text{Var}[\hat{f}(x_0)] + (f(x_0) - E[\hat{f}(x_0)])^2$$

$$= \text{Var}[\epsilon] + \text{Var}[\hat{f}(x_0)] + (E[f(x_0) - \hat{f}(x_0)])^2$$

$$= \text{Var}[\epsilon] + \text{Var}[\hat{f}(x_0)] + \text{Bias}[\hat{f}(x_0)]^2$$

確率変数の期待値、分散など

- ・ 確率変数の和の期待値

$$E[X + Y] = E[X] + E[Y]$$

- ・ 分散

$$\begin{aligned}\text{Var}[X] &= E[(X - E[X])^2] \\ &= E[X^2 - 2E[X]X + E[X]^2] \\ &= E[X^2] - 2E[X]E[X] + E[X]^2 \\ &= E[X^2] - E[X]^2\end{aligned}$$

- ・ X と Y が独立な場合

$$E[XY] = E[X]E[Y]$$

$$\begin{aligned}\text{Var}[X + Y] &= E[(X + Y)^2] - E[X + Y]^2 \\ &= E[X^2 + 2XY + Y^2] - (E[X] + E[Y])^2 \\ &= E[X^2] + 2E[X]E[Y] + E[Y^2] - (E[X] + E[Y])^2 \\ &= E[X^2] - E[X]^2 + E[Y^2] - E[Y]^2 \\ &= \text{Var}[X] + \text{Var}[Y]\end{aligned}$$

$$\begin{aligned}E[XY] &= \sum_i \sum_j p(x_i, y_j) x_i y_j \\ &= \sum_i \sum_j p(x_i) p(y_j) x_i y_j \\ &= \sum_i p(x_i) x_i \sum_j p(y_j) y_j \\ &= \sum_i p(x_i) x_i E[Y] \\ &= E[Y] \sum_i p(x_i) x_i \\ &= E[Y] E[X]\end{aligned}$$

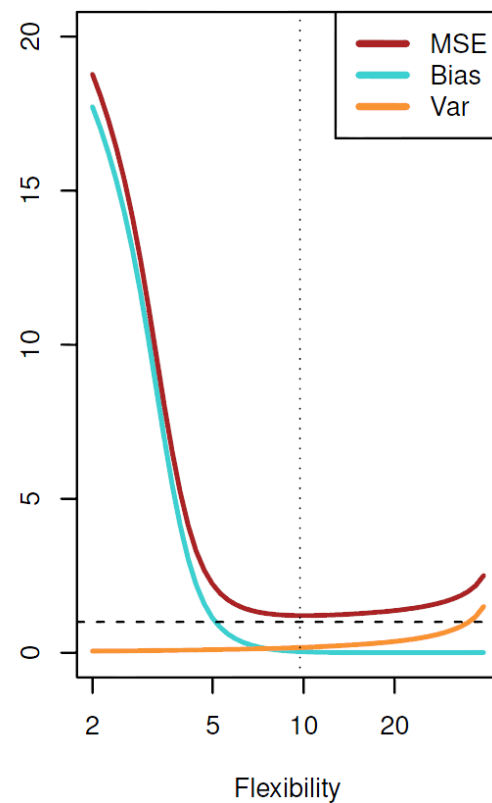
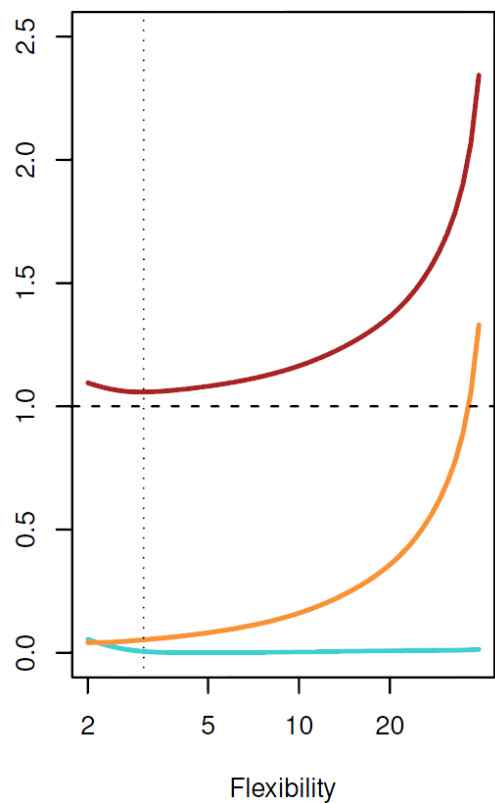
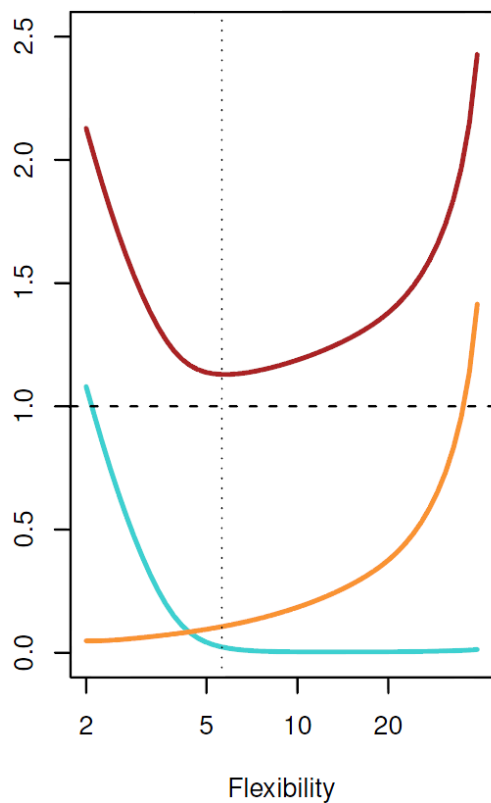
モデルの精度

- バリアンス (variance)
 - 学習データの違いによってどれだけ \hat{f} が変化するか
- バイアス (bias)
 - \hat{f} が真の f とどれだけずれているか
- モデルの表現力 (柔軟さ) との関係

	バイアス	バリアンス
表現力の高いモデル (例、平滑化スプライン)	小	大
表現力の低いモデル (例、線形回帰)	大	小

モデルの精度


- モデルの表現力との関係



モデルの精度

- 分類問題の場合
 - 学習データでの誤り率 (error rate)

$$\frac{1}{n} \sum_{i=1}^n I(y_i \neq \hat{y}_i)$$

 $y_i \neq \hat{y}_i$ の場合は1、そうでなければ0をとる

- 未知のデータ(テストデータ)での誤り率

$$\text{Ave}(I(y_0 \neq \hat{y}_0))$$

モデルの精度

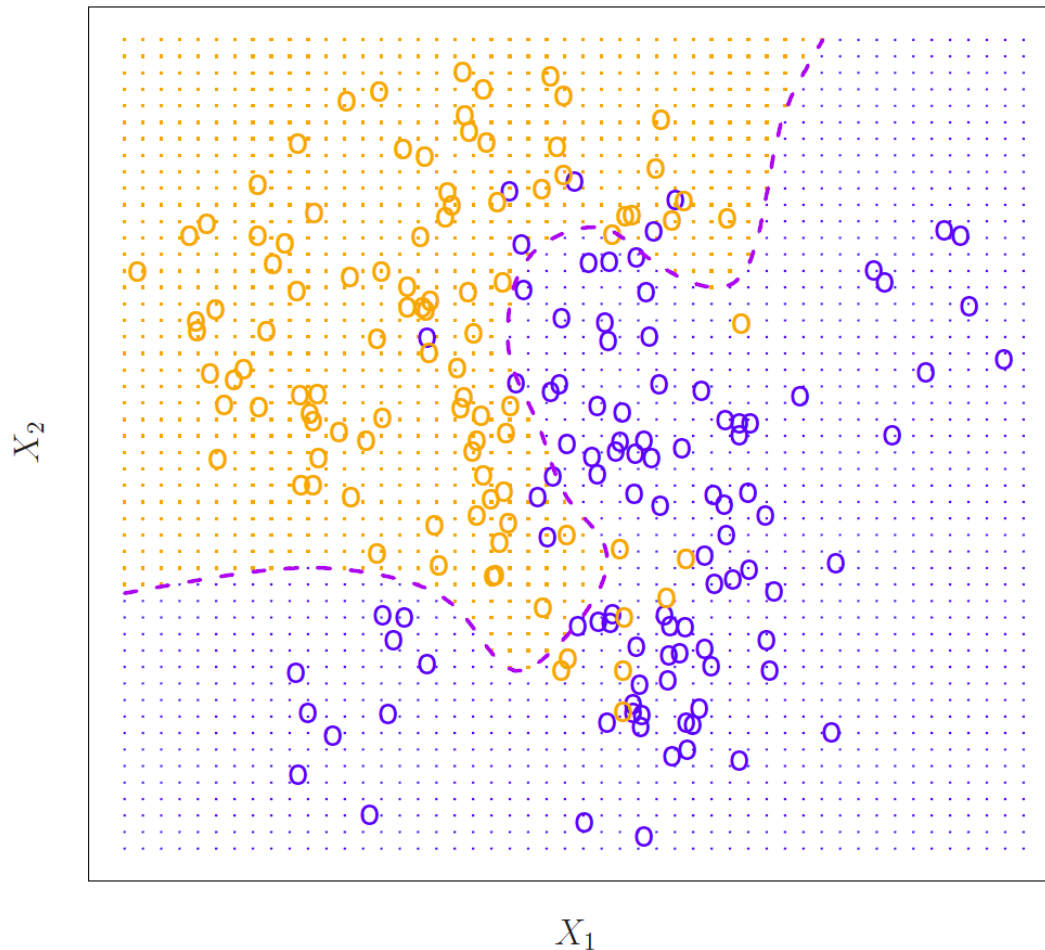
- ベイズ分類器 (Bayes classifier)
 - 条件付確率が最大になるクラスを選ぶ

$$\Pr(Y = j | X = x_0)$$

- (期待)誤り率が最小になる
- ベイズ最適決定 (Bayes optimal decision) とも
- 実際には条件付確率はわからないので、あくまでも理論上の分類器

モデルの精度

- ベイズ決定境界 (Bayes decision boundary)



モデルの精度

- ベイズ誤り率 (Bayes error rate)
 - ベイズ分類器の誤り率

$$1 - E \left[\max_j \Pr(Y = j | X) \right]$$

- 回帰問題での「削減不可能な誤差」(irreducible error)に相当

K最近傍法

(K-nearest neighbors method)

- クラスの条件付確率を推定

$$\Pr(Y = j | X = x_0) = \frac{1}{K} \sum_{i \in N_0} I(y_i = j)$$

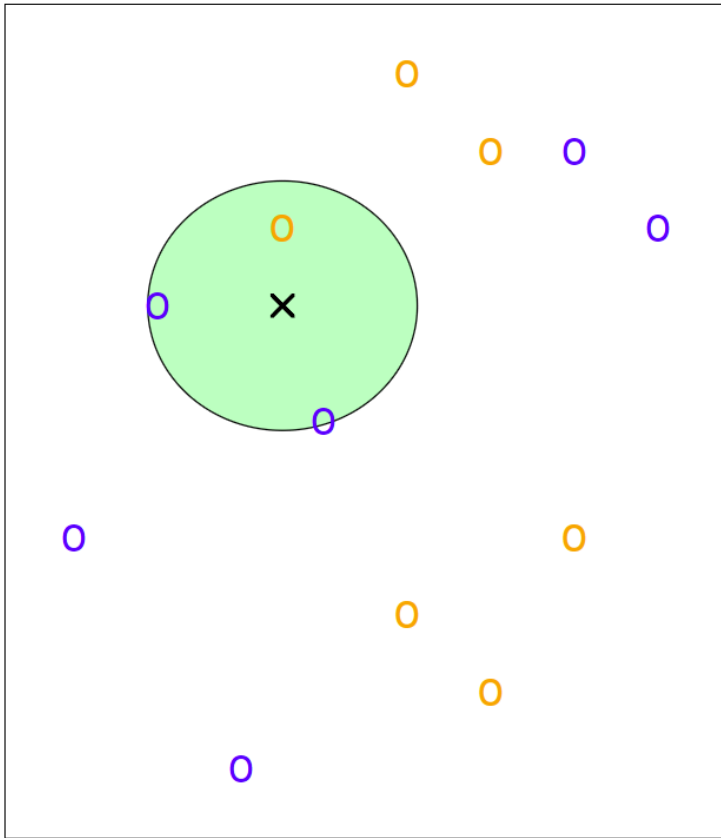
N_0 : 学習データ中で x_0 に最も近い K 個の観測データ

– 確率の最も大きいクラスに分類

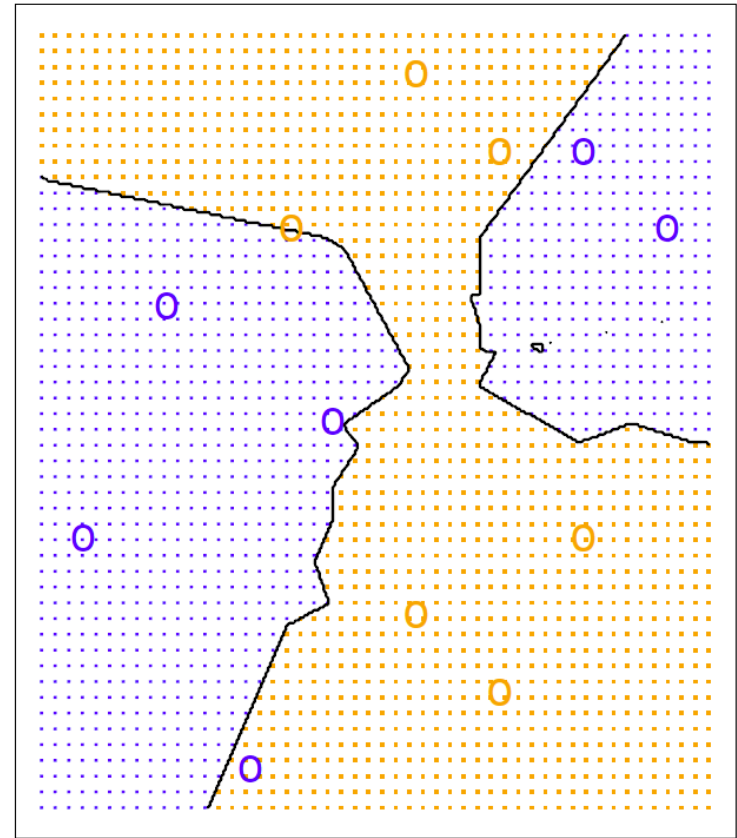
K最近傍法

(K-nearest neighbors method)

- $K = 3$ の例

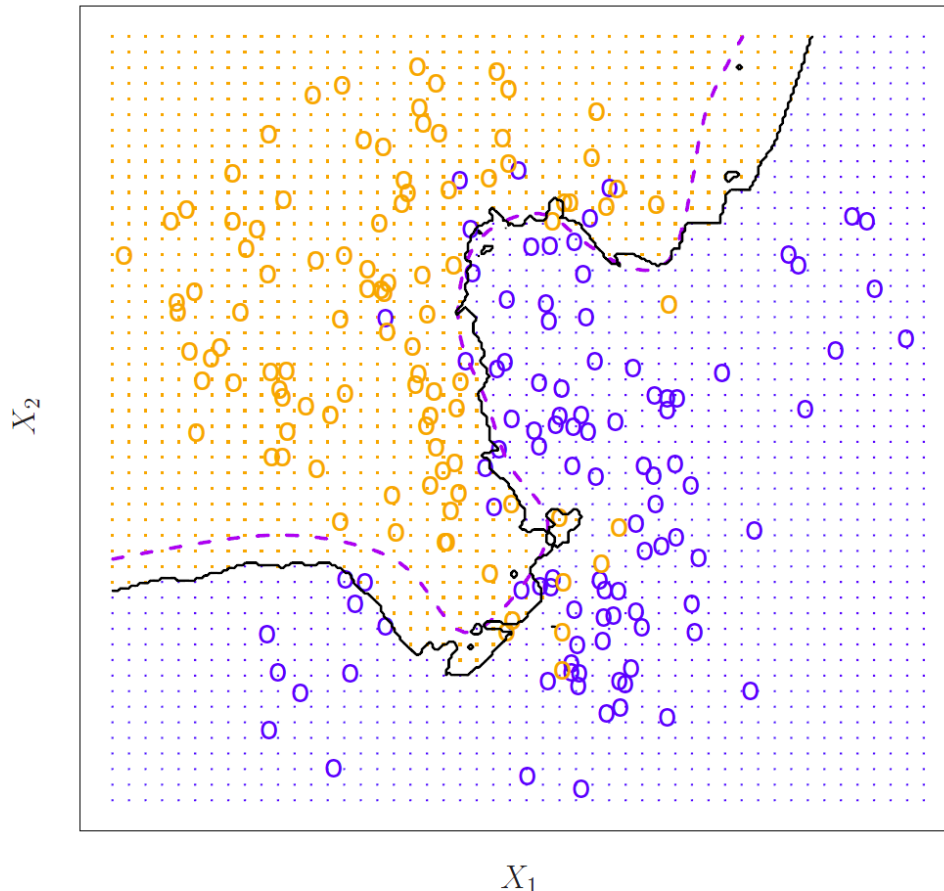


決定境界
(decision boundary)



K最近傍法 (K-nearest neighbors method)

KNN: K=10

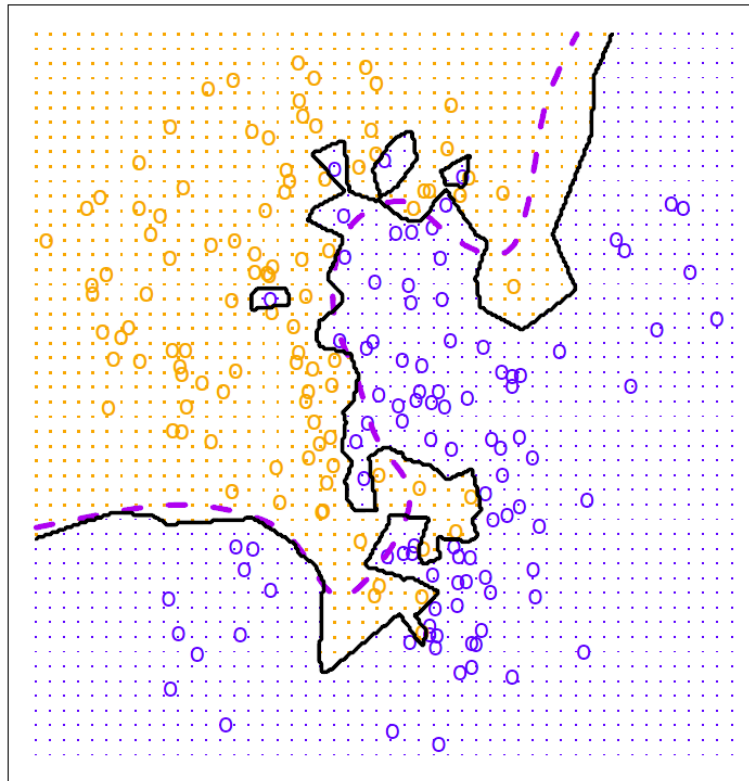


- KNN決定境界
 - 図中の黒い線
 - テスト誤り率: 0.1363
- ベイズ決定境界
 - 図中の紫の点線
 - テスト誤り率: 0.1304

K最近傍法

(K-nearest neighbors method)

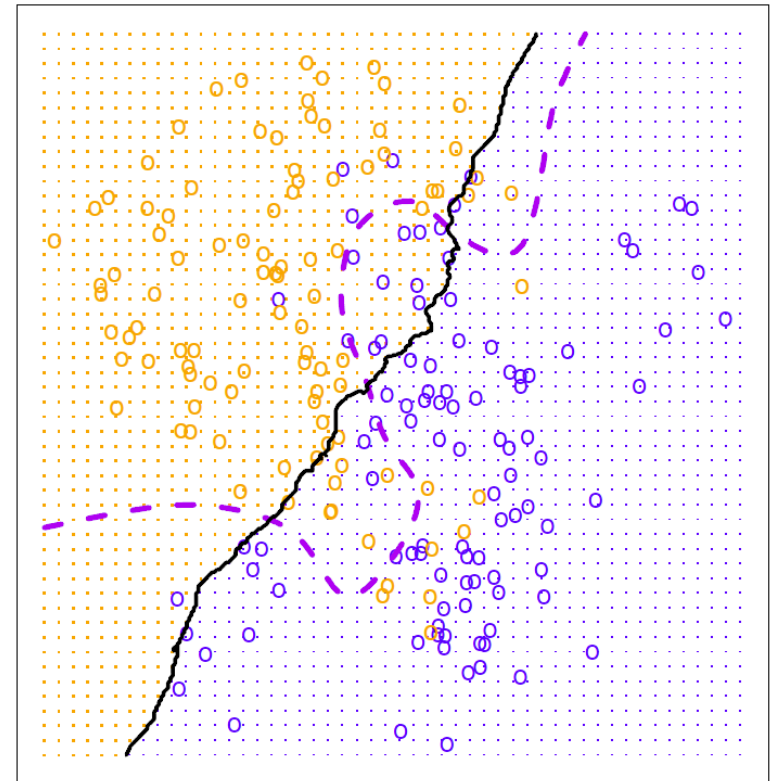
KNN: K=1



テスト誤り率: 0.1695

学習データでの誤り率はゼロ(!)

KNN: K=100

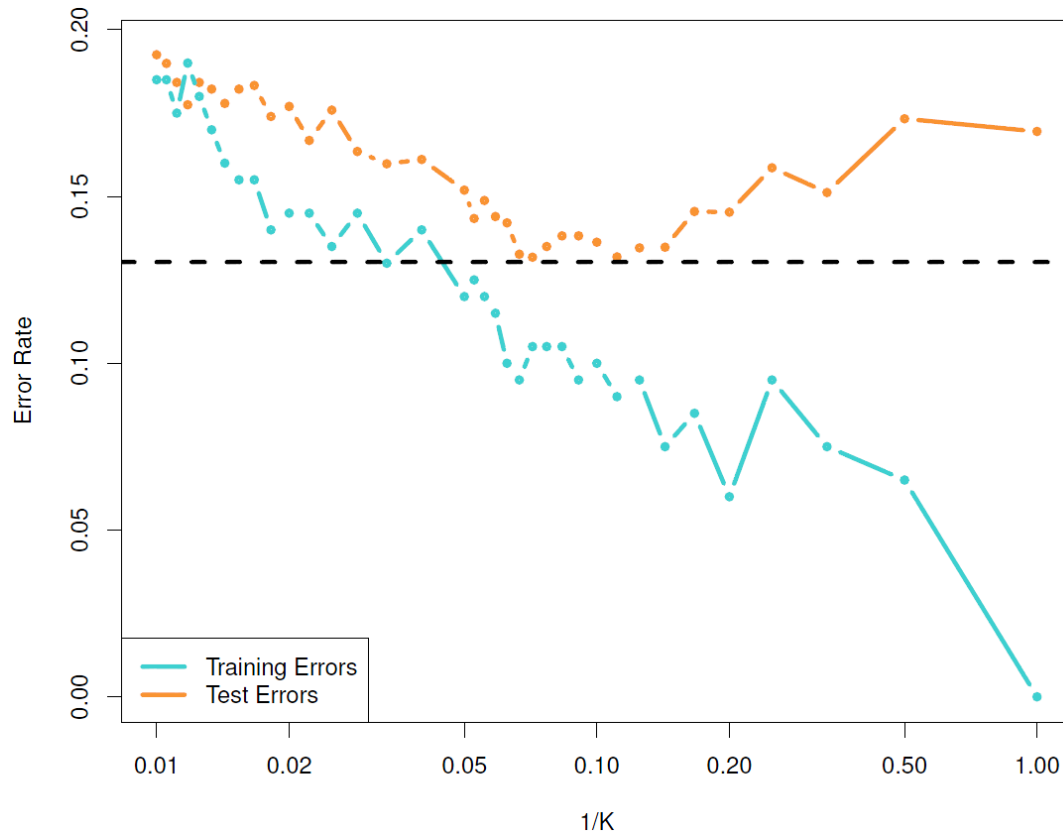


テスト誤り率: 0.1925

K最近傍法

(K-nearest neighbors method)

- モデルの表現力と誤り率



→ モデルの表現力は高すぎても低すぎてもダメ