

# データサイエンス 第6回

～回帰分析（2）～

情報理工学系研究科  
創造情報学専攻  
中山 英樹

# 本日の内容

- 先週の復習 (+a)
- 回帰分析
  - 非線形回帰分析

# 復習

- 機械学習の言葉でいうと…
- データから何かを発見したい → 教師なし学習

説明変数	手法
量的データ(比尺度)	主成分分析、因子分析、LPP
量的データ(間隔尺度)	クラスター分析、多次元尺度構成法、数量化Ⅳ類
質的データ	数量化Ⅲ類、対応分析

- データを使って何かを予測したい → 教師あり学習

目的変数	説明変数	手法
量的データ	量的データ	回帰分析
	質的データ	数量化Ⅰ類
質的データ	量的データ	判別分析
	質的データ	数量化Ⅱ類

# 回帰分析

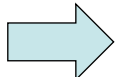
- 説明変数から目的変数を予測するモデル（関数）を構築
  - （典型的には）量的データから量的データを予測
  - 例）人口・気温から消費電力を予測

以下では

$d$  次元のベクトル入力  $\mathbf{x}$  から実数値  $y$  を出力するモデル  $y = f_{\theta}(\mathbf{x})$  を、 $N$  個の学習サンプル  $\{\mathbf{x}_i, y_i\}_{i=1}^N$  から学習する

という問題を考える

( $\theta$  はモデルのパラメータ)

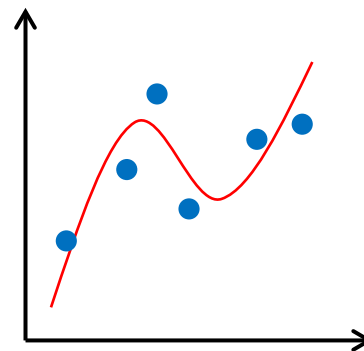
- 学習サンプルに最もフィットするモデルを求めたい
  - “フィット”するとは？  予測と真の値のズレを最小化する！

# 最小二乗法 (least squares)

- 学習サンプルにおける、モデルの出力  $\hat{y}_i = f_{\theta}(\mathbf{x}_i)$  と真値  $y_i$  の二乗誤差

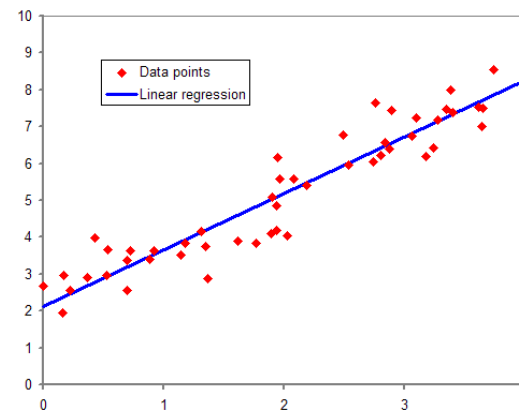
$$E(\theta) = \sum_{i=1}^N (y_i - \hat{y}_i)^2 = \sum_{i=1}^N (y_i - f_{\theta}(\mathbf{x}_i))^2 \text{ を最小化}$$

- 誤差の分布に分散一定の正規分布を仮定
  - 他にも誤差の基準はいろいろある (異なる分布に対応)
  - 二乗誤差を基準にすると実用上はとても便利
- 
- 例) 多項式フィッティング (xは1次元)



# 線形回帰: linear regression

- 説明変数の一次式（線形結合）によるモデル
  - 単回帰：説明変数が一つ  $y = ax + b$
  - 重回帰：説明変数が複数  $y = a_0x_0 + a_1x_1 + a_2x_2 + \cdots + a_dx_d + b = \mathbf{a}^T \mathbf{x} + b$ 
    - 説明変数は互いに無相関であることを仮定
- 対象が線形な関係を有する場合に力を発揮
  - 実用上は厳密に線形であることは少ない（そもそも分からないことの方が多い）
  - 予測性能が実用に耐えるかの評価が大事
  - 前処理（対数をとったり）は重要
- シンプルなモデルであり、学習はスケーラブル



# 線形回帰

- 最小二乗法によるモデルフィッティング

$$E(\mathbf{a}, b) = \sum_{i=1}^N (y_i - \hat{y}_i)^2 = \sum_{i=1}^N (y_i - (\mathbf{a}^T \mathbf{x}_i + b))^2$$

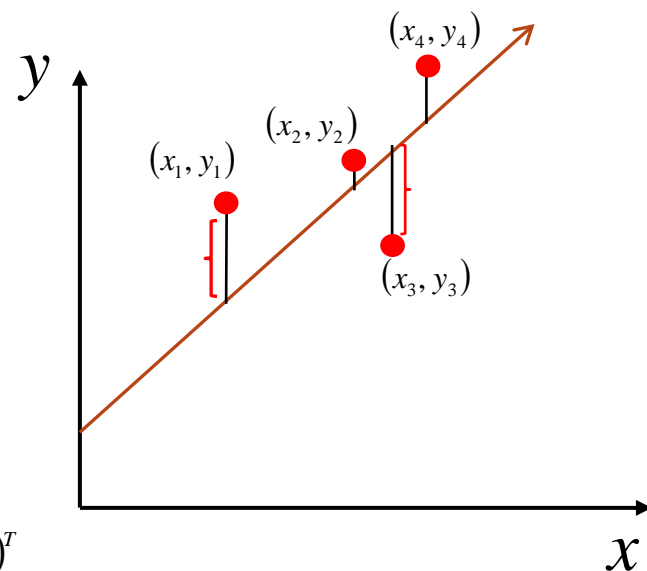
$\mathbf{a}, b$  でそれぞれ偏微分して0とおくと

$$\frac{\partial E}{\partial \mathbf{a}} = -2 \sum_{i=1}^N \mathbf{x}_i (y_i - \mathbf{a}^T \mathbf{x}_i) = 0 \quad \Rightarrow \quad \hat{\mathbf{a}} = \left( \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T \right)^{-1} \left( \sum_{i=1}^N \mathbf{x}_i y_i \right)$$
$$= \underbrace{(X X^T)^{-1}}_{\text{自己相関行列}} X \mathbf{y}$$

ただし  $X = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N)$ ,  $\mathbf{y} = (y_1, y_2, \dots, y_N)^T$

$$\frac{\partial E}{\partial b} = -2 \sum_{i=1}^N (y_i - (\mathbf{a}^T \mathbf{x}_i + b)) = 0 \quad \Rightarrow \quad \hat{b} = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{\mathbf{a}}^T \mathbf{x}_i)$$

定数項  $b$  を用意する代わりに、特徴ベクトル  $\mathbf{x}$  に常に1の要素を付与してもよい



# 最尤推定との関係性

- 尤度：ある仮説（モデル）のもとで観察されたデータが生じる確率
  - 尤度（もっともらしさ）を最大とするパラメータを求める

$$L = \prod_{i=1}^N \underline{p(y_i | \mathbf{a}, b)}$$

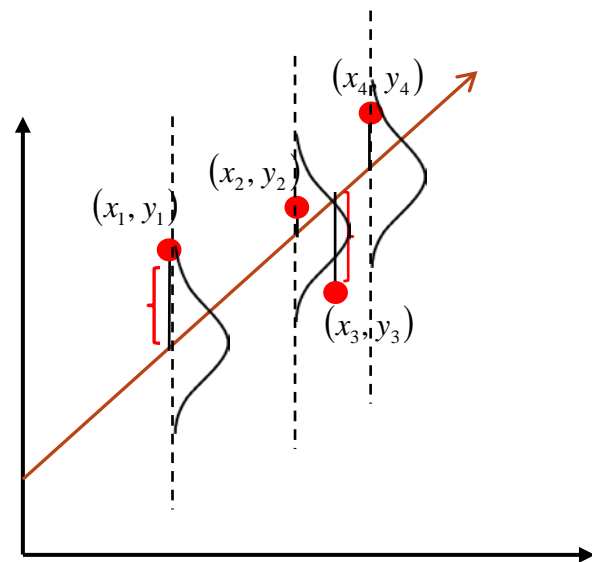
パラメータ $\mathbf{a}, b$ のもとでの $y_i$ の事後確率  
(各 $y_i$ は独立試行)

$$= \prod_{i=1}^N \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(y_i - (\mathbf{a}^T \mathbf{x}_i + b))^2}{2\sigma^2}\right)$$

対数尤度

$$\log L = -\frac{N}{2} \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^N (y_i - (\mathbf{a}^T \mathbf{x}_i + b))^2$$

結局これを最小化



# 線形回帰の弱点（１）

- 説明変数の間に強い相関や線形従属の関係があると正しい解がでない  
(= 多重共線性, multicollinearity)

$$\hat{\mathbf{a}} = \underline{(\mathbf{X}\mathbf{X}^T)^{-1}} \mathbf{X} \mathbf{y}$$

相関行列がランク落ちし、逆行列が求まらない

- 実データでは、説明変数の間に相関がある方がふつう
  - 解けないだけならまだいいが、不安定な解を出すことがある  
(行列式がゼロに近いので、係数  $\mathbf{a}$  が異常に大きな値になる)
  - あらかじめ相関の高い説明変数は除外しておくことが
- サンプルが次元数に対して少ない場合も不安定になりやすい

# リッジ回帰

- 自己相関行列に単位行列を微小な重みをつけて加算し、安定的に逆行列を計算（正則化）

$$\hat{\mathbf{a}} = (XX^T + \gamma I)^{-1} X \mathbf{y}$$

- いろんな解釈が可能
  - 過学習を抑える効果（人工的なノイズを加えている）
  - 最小化する目的関数として、誤差項に二乗ノルムを加えている  
= **L2正則化**

$$E'(\mathbf{a}, b) = \sum_{i=1}^N \left( y_i - (\mathbf{a}^T \mathbf{x}_i + b) \right)^2 + \gamma \|\mathbf{a}\|^2$$



係数(の絶対値)をできるだけ小さくするような作用

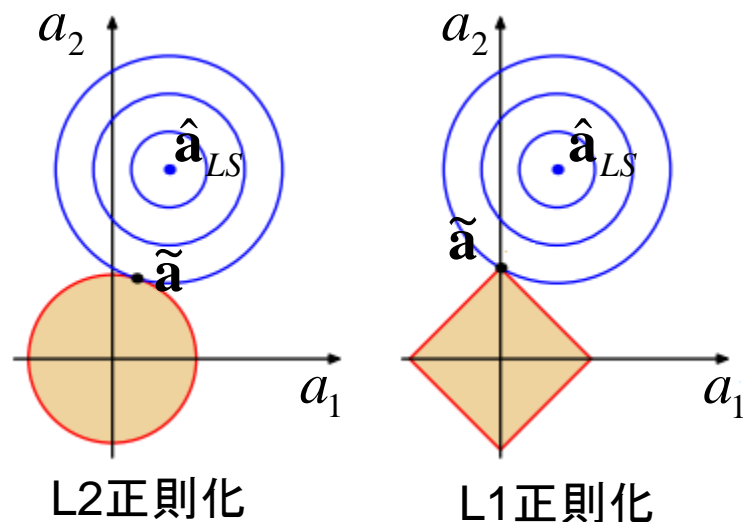
# スパース線形回帰

- L1正則化(Lasso)を用いた線形回帰

$$E''(\mathbf{a}, b) = \sum_{i=1}^N \left( y_i - (\mathbf{a}^T \mathbf{x}_i + b) \right)^2 + \underbrace{\gamma \|\mathbf{a}\|_1}_{\text{L1ノルム}} = \sum_i |a_i|$$

- スパース（ゼロ要素が多い）な係数ベクトルが出やすい
- 汎化誤差が向上する場合がある。特徴選択にも使われる。
- データ容量や計算コストの削減につながる

sklearn.linear\_model.Lasso など



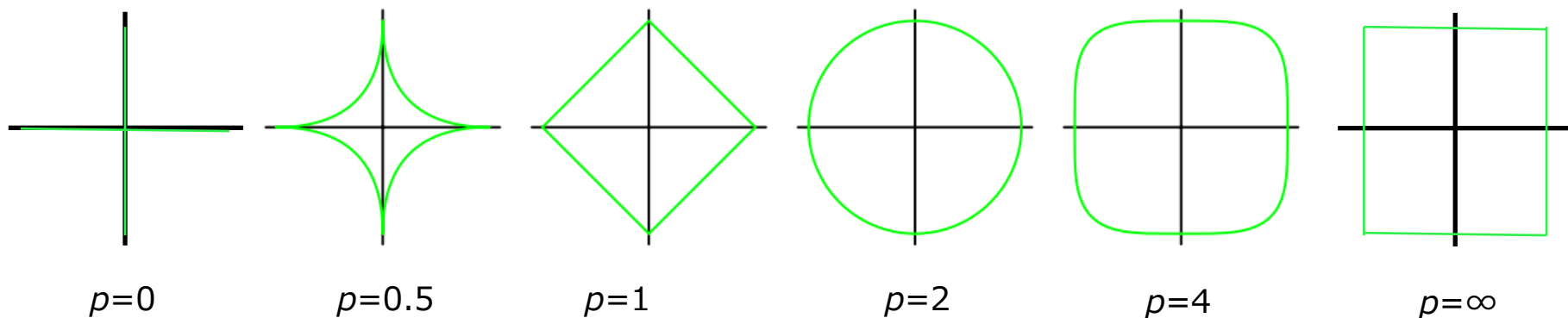
# Lp正則化

- スパースネスの程度をコントロールできる

$$\|\mathbf{a}\|_p = \left( |a_1|^p + |a_2|^p + \cdots + |a_d|^p \right)^{\frac{1}{p}}$$

ただし  $\|\mathbf{a}\|_0 = \sum_{i=1}^d \delta(a_i) \quad \delta(a_i) = \begin{cases} 1 & (a_i \neq 0) \\ 0 & (a_i = 0) \end{cases}$

$$\|\mathbf{a}\|_\infty = \max(|a_1|, |a_2|, \dots, |a_d|)$$



c.f. Yukawa & Amari, “Lp-regularized least squares ( $0 < p < 1$ ) and critical path”, 2013.

# Elastic Net [Zhou et al., 2005]

- L1正則化(Lasso) + L2正則化(Ridge regression)

$$E''(\mathbf{a}, b) = \sum_{i=1}^N \left( y_i - (\mathbf{a}^T \mathbf{x}_i + b) \right)^2 + \gamma_1 \|\mathbf{a}\|_1 + \gamma_2 \|\mathbf{a}\|_2$$

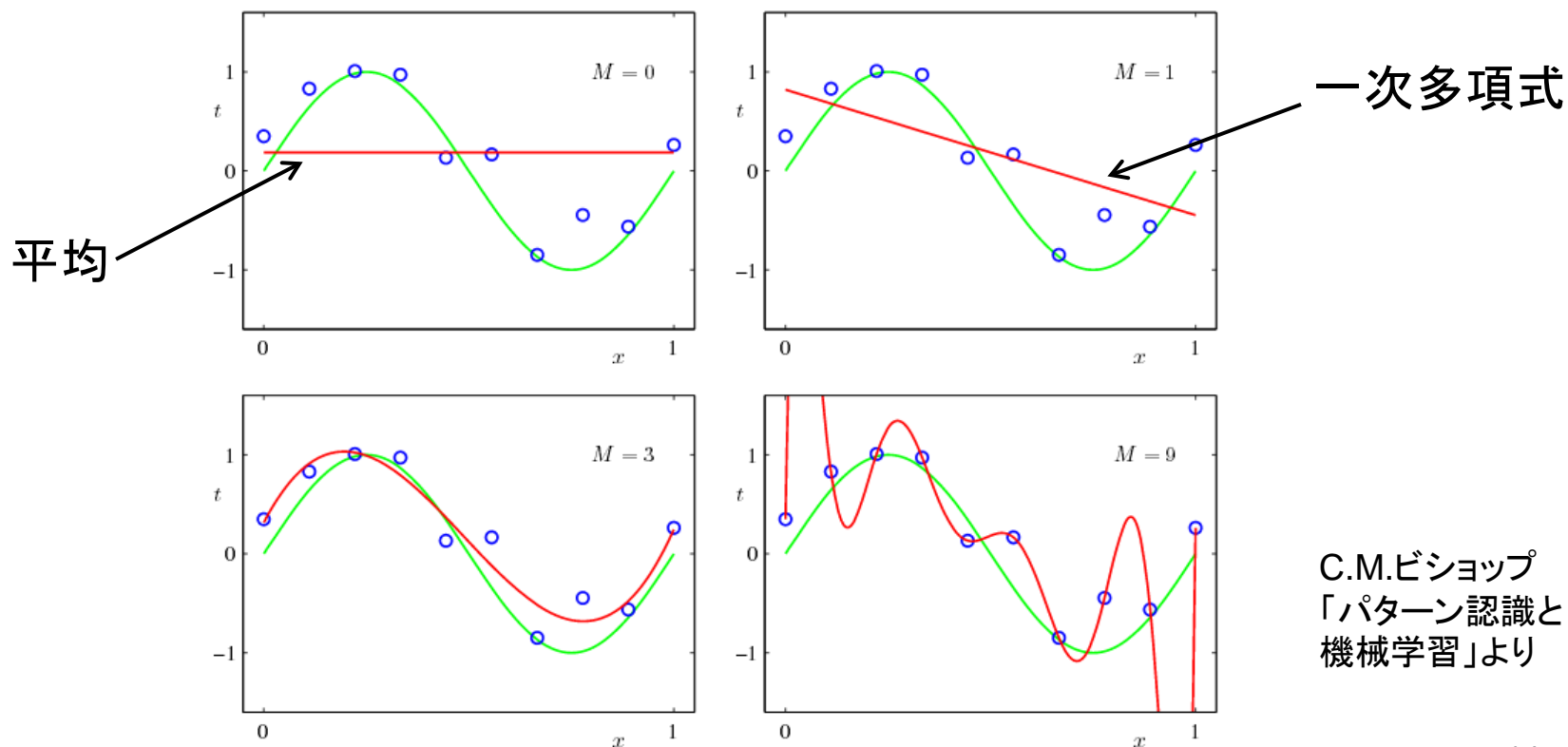
- リッジ回帰を解いたあと、Lassoの最適化を行う

sklearn.linear\_model.ElasticNet など

[http://scikit-learn.org/stable/modules/generated/sklearn.linear\\_model.ElasticNet.html](http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.ElasticNet.html)

# 過学習（オーバーフィッティング）

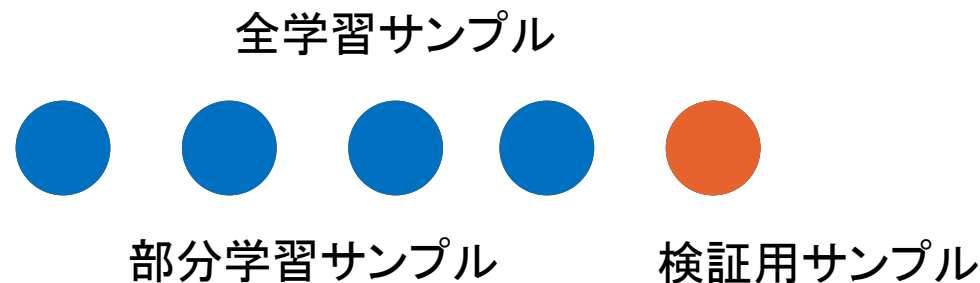
- 訓練誤差は小さいが、汎化誤差（予測誤差）が著しく大きい状態
  - データ数や実際の構造に対して複雑すぎるモデルを用いることで生じる
- 多項式回帰の次数を上げていけば、全ての学習データを通る線は引ける
  - 実際には意味のない解！（訓練誤差はゼロだが汎化誤差は非常に大きくなる）
  - 学習サンプルは有限



# パラメータのチューニング方法

- 交差検定法（クロスバリデーション）

1. もともとの学習サンプルを分割し、新しい学習サンプルと検証用サンプルに分割する（例えば4:1などに）
2. 新しい学習サンプルで、候補となるモデル（この場合パラメータごとに）を学習し、検証用データの予測誤差を調べる
3. 学習サンプルと検証用サンプルを順番に入れ替え、2を繰り返す
4. 全試行の平均予測誤差が最も小さかったものを採用する



# 情報量規準

- 赤池情報量規準 (AIC) (×基準)
  - 以下を最小とするモデルを選択する

$$-2\ln(L) + 2M$$

対数尤度

パラメータ数

- BIC (Bayesian information criterion)

$$-2\ln(L) + M \ln(N)$$

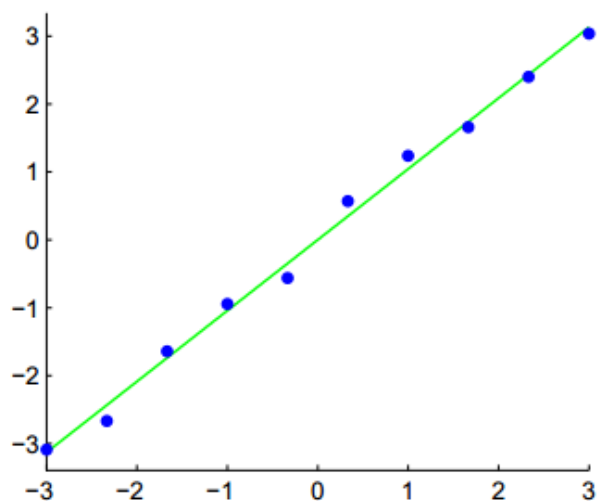
- MDL (minimal description length)

$$-\ln(L) + \frac{M \ln(N)}{2}$$

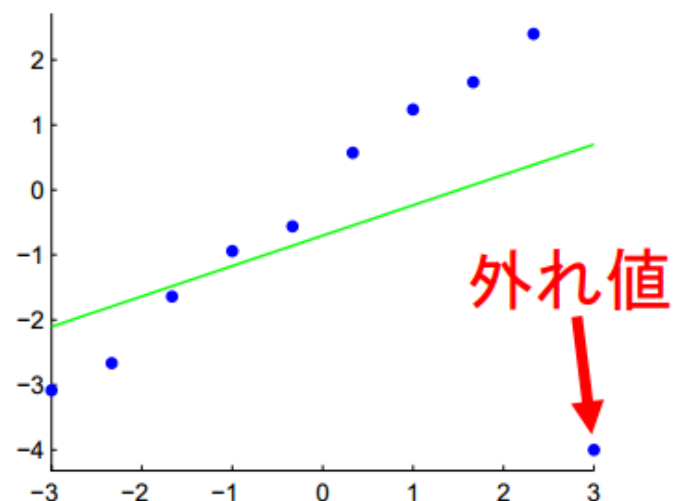
- 訓練データだけから、「そこそこいいモデル」が得られる
  - 実際には、過度に単純なものが得られる場合が多いらしい
  - 変数選択などに使える

# 線形回帰の弱点（２）

- 二乗誤差基準の場合、外れ値（異常値）に弱い
  - 測定のノイズなど
  - 値のズレが二乗で効いてくる！



通常の線形回帰  
(外れ値なし)



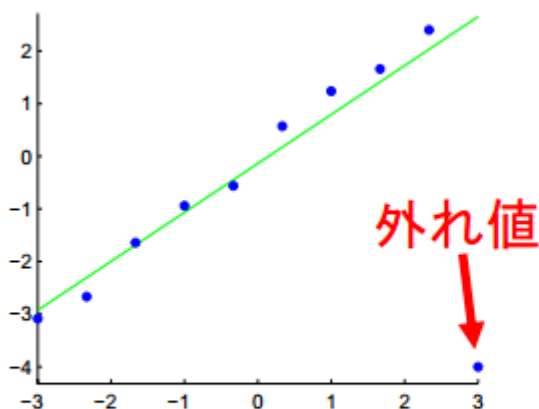
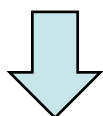
通常の線形回帰  
(外れ値あり)

(図: 杉山将)

# ロバスト線形回帰

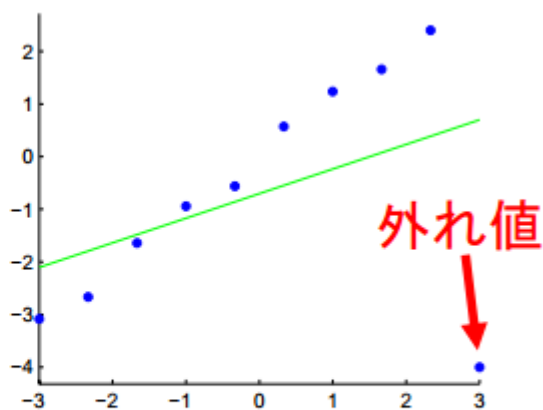
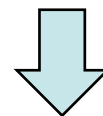
- L1損失関数による誤差基準
  - 線形計画問題で解を求める

$$\hat{\mathbf{a}}_{l1} = \arg \min_{\mathbf{a}} \sum_{i=1}^N |y_i - (\mathbf{a}^T \mathbf{x}_i + b)|$$



ロバスト学習

$$\hat{\mathbf{a}}_{l2} = \arg \min_{\mathbf{a}} \sum_{i=1}^N (y_i - (\mathbf{a}^T \mathbf{x}_i + b))^2$$

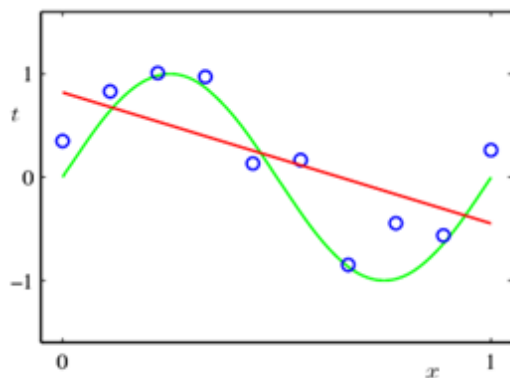


通常の  
最小二乗学習

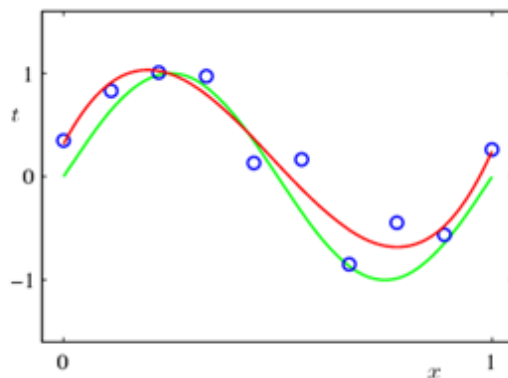


# 非線形回帰分析

- 目的変数と説明変数が非線形な関係である場合の回帰分析
  - 一般化線形モデル
  - 多層パーセプトロン
  - カーネル回帰分析（後日）



線形回帰



多項式回帰（次数3）

# 一般化線形モデル (generalized linear model)

- 残差が正規分布以外の指数分布族に従うモデル
  - 目的変数にある関数で非線形変換を加えると、説明変数の線形結合で表現できる

$$g(\underline{\mu}) = \mathbf{a}^T \mathbf{x} + b$$

目的変数の分布の平均 (期待値)  $g()$  をリンク関数と呼ぶ

- Rのstatsパッケージの関数glm  
> glm(formula, family, data)

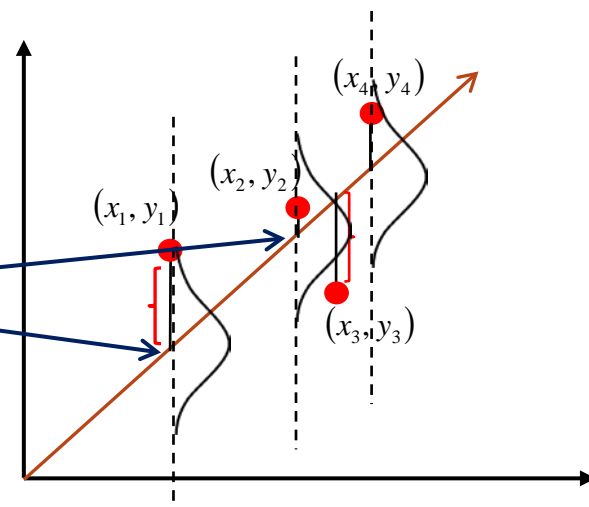
回帰の種類	分布族 (family)	リンク関数	
線形回帰	正規分布 (gaussian)	$\mu$	link="identity"
ポアソン回帰	ポアソン分布 (poisson)	$\log(\mu)$	link="log"
ロジスティック回帰	二項分布 (binomial)	$\log(\mu/(1-\mu))$	link="logit"

# 線形回帰（基本）

- 目的変数は等分散の正規分布に従う
  - i 番目の分布の平均  $\mu_i$ 、標準偏差  $\sigma$  とする

$$g_{LR}(\mu_i) = \mu_i = \mathbf{a}^T \mathbf{x}_i + b$$

リンク関数は恒等変換



# ポアソン回帰

- 目的変数がポアソン分布に従う場合
  - ある一定の時間内に発生する離散的な事象を数える  
特定の確率変数を持つ離散確率分布

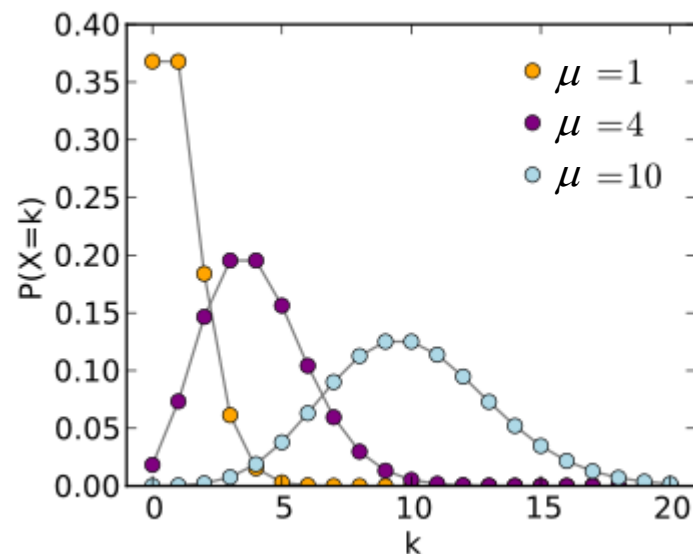
- 計数（カウント）データに使う

$$P(y_i = k) = \frac{\mu_i^k e^{-\mu_i}}{k!}$$

$$\mu_i = \exp(\mathbf{a}^T \mathbf{x}_i + b)$$

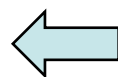
$$g_{Po}(\mu_i) = \log(\mu_i) = \mathbf{a}^T \mathbf{x}_i + b$$

対数線形モデル



例) Large-scale behavioral targeting [Chen et al., KDD'09]

- 広告CTRをユーザの行動データから予測



$\mu_i = \mathbf{a}^T \mathbf{x}_i$  で解いている

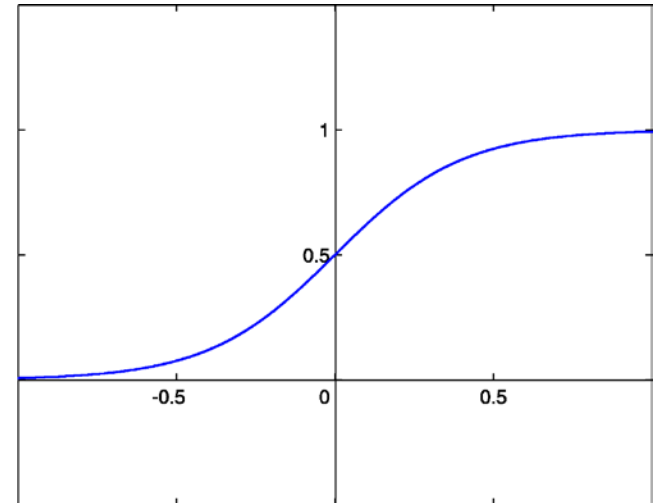
# ロジスティック回帰

- 目的変数が二項分布に従う場合
- 二値データ（質的データ）の回帰
  - 実用上はクラス識別の手法として解釈される場合が多い

$$y_i = P(C_1 | \mathbf{x}_i) = \frac{\exp(\mathbf{a}^T \mathbf{x}_i + b)}{1 + \exp(\mathbf{a}^T \mathbf{x}_i + b)}$$

$$\mu_i = \frac{\exp(\mathbf{a}^T \mathbf{x}_i + b)}{1 + \exp(\mathbf{a}^T \mathbf{x}_i + b)}$$

$$g(\mu_i) = \log\left(\frac{\mu_i}{1 - \mu_i}\right) = \mathbf{a}^T \mathbf{x}_i + b$$



リンク関数はロジット関数

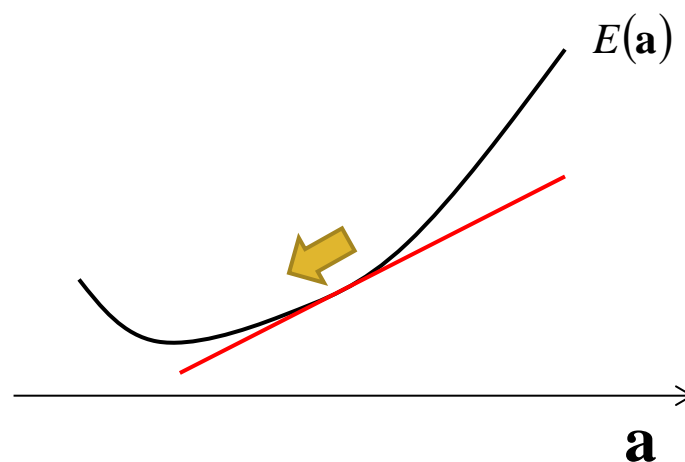
# 最急降下法

- 収束するまで以下のように更新
  - $\alpha$  は微小な正のパラメータ

$$\mathbf{a} \leftarrow \mathbf{a} - \alpha \sum_{i=1}^N (y_i - t_i) \mathbf{x}_i$$


---

$\frac{E(\mathbf{a})}{\partial \mathbf{a}}$



# 解き方

- 最尤推定

訓練データ集合  $\{\mathbf{x}_i, t_i\}, t_i \in \{0, 1\}$

尤度関数は  $L = \prod_{i=1}^N y_i^{t_i} \{1 - y_i\}^{1-t_i} \quad y_i = P(C_1 | \mathbf{x}_i)$

負の対数尤度は  $E(\mathbf{a}) = -\ln L = \sum_{i=1}^N \{t_i \ln y_i + (1 - t_i) \ln(1 - y_i)\}$



$$\frac{\partial E(\mathbf{a})}{\partial \mathbf{a}} = \sum_{i=1}^N \underline{(y_i - t_i) \mathbf{x}_i}$$

エラーに説明変数をかけたもの

# 例)

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

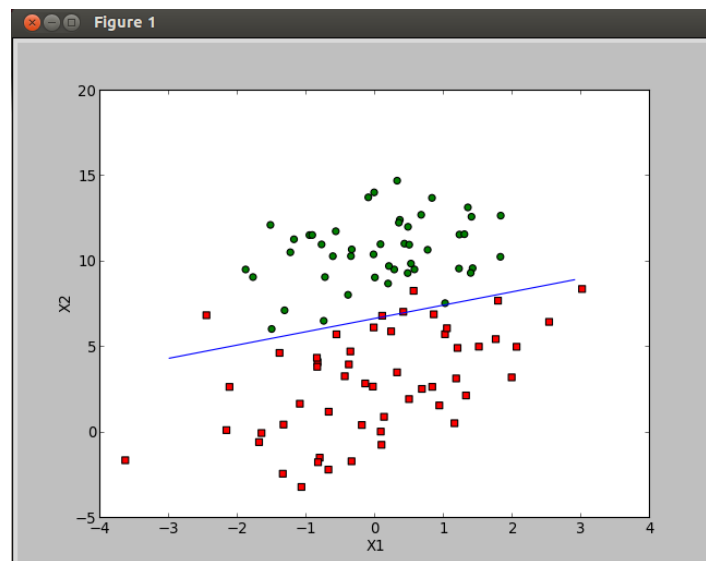
```
>>> import logRegres
```

```
>>> dataArr,labelMat=logRegres.loadDataSet()
```

```
>>> weights=logRegres.gradAscent(np.array(dataArr),labelMat)
```

```
>>> logRegres.plotBestFit(weights.getA())
```

-0.017612	14.053064	0
-1.395634	4.662541	1
-0.752157	6.538620	0
-1.322371	7.152853	0
0.423363	11.054677	0
0.406704	7.067335	1
...	...	...

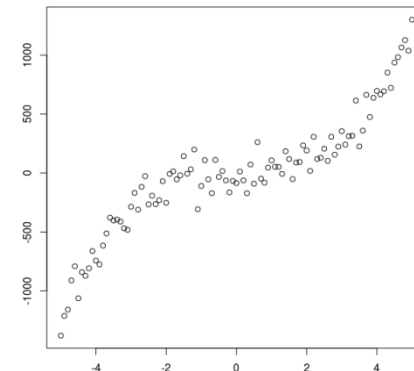


# コード

```
def sigmoid(inX):  
    return 1.0/(1+exp(-inX))  
  
def gradAscent(dataMatIn, classLabels):  
    dataMatrix = mat(dataMatIn)          #convert to NumPy matrix  
    labelMat = mat(classLabels).transpose() #convert to NumPy matrix  
    m,n = shape(dataMatrix)  
    alpha = 0.001  
    maxCycles = 500  
    weights = ones((n,1))  
    for k in range(maxCycles):            #heavy on matrix operations  
        h = sigmoid(dataMatrix*weights)   #matrix mult  
        error = (labelMat - h)            #vector subtraction  
        weights = weights + alpha * dataMatrix.transpose()* error #matrix mult  
    return weights
```

# その他

- 一般の関数を用いた回帰
  - Rの関数 nls など  
`nls(formula, data, start, trace)`
- 多項式回帰の例
  - > `set.seed(30)`
  - > `x<-seq(-5,5,0.1)`
  - > `y<-10*x^3+100*rnrm(x,0,1)` #3次式に従う人口データを生成
  - > `plot(x,y)`



# つづき

```
> fm3<-nls(y~a+b*x+c*x^2+d*x^3,start=c(a=1,b=1,c=1,d=1),trace=T)
21031980 : 1 1 1 1
1073246 : 1.3077234 13.8639457 -0.9720568 9.4123383
> summary(fm3)
```

Formula:  $y \sim a + b * x + c * x^2 + d * x^3$

任意に指定可能(解ければ)

Parameters:

	Estimate	Std. Error	t value	Pr(> t )
a	1.3077	15.7011	0.083	0.934
b	13.8639	8.9781	1.544	0.126
c	-0.9721	1.3769	-0.706	0.482
d	9.4123	0.5379	17.498	<2e-16 ***

---

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

Residual standard error: 105.2 on 97 degrees of freedom

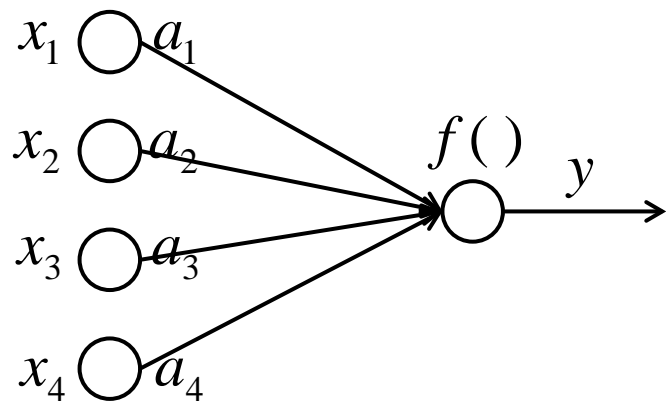
Number of iterations to convergence: 1

Achieved convergence tolerance: 1.103e-07

```
> AIC(fm3)
[1] 1233.004
```

# ニューラルネットワークによる回帰

- 単純パーセプトロン
  - 一般化線形モデルと密接に関係



$$y = f(\eta)$$

$$\eta = \mathbf{a}^T \mathbf{x}$$

$$f(\eta) = \eta \quad \rightarrow \text{線形回帰と等価}$$

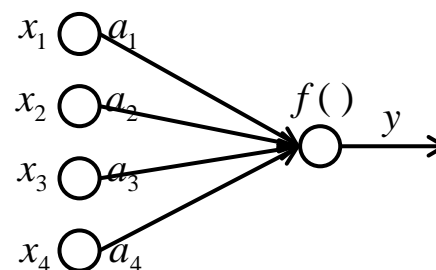
$$f(\eta) = \frac{1}{1 + \exp(-\eta)} \quad \rightarrow \text{ロジスティック回帰と等価}$$

# 単層パーセプトロンの学習

- 二乗誤差最小化

$$\varepsilon^2 = \sum_{i=1}^N (y_i - \hat{y}_i)^2 = \sum_{i=1}^N (y_i - f(\mathbf{a}^T \mathbf{x}_i))^2$$

$$\frac{\partial \varepsilon^2}{\partial a_k} = -2 \sum_{i=1}^N (y_i - \hat{y}_i) x_{ik}$$



なので、最急降下法による更新式は、

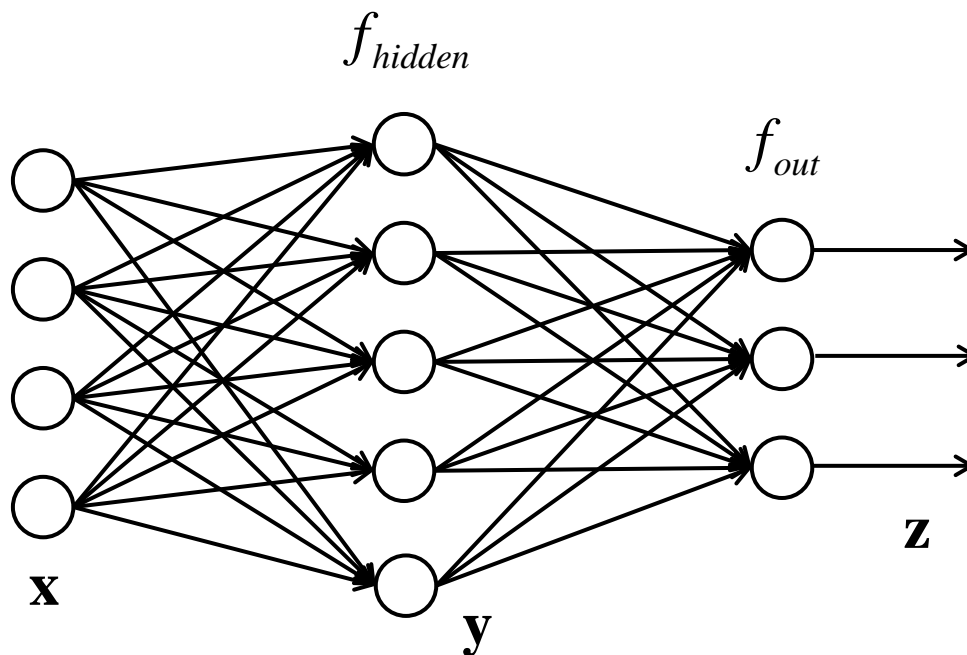
$$a_k \leftarrow a_k + \alpha \left( \sum_{i=1}^N (y_i - \hat{y}_i) x_{ik} \right)$$

Widrow-Hoffの学習規則

※  $f(\eta) = \eta$  の時は、線形回帰と同じ解析解が求まる

# 多層パーセプトロン

- 十分な数の素子があれば、任意の連続関数は3層のニューラルネットワークで近似できる
  - 誤差逆伝播法と呼ばれる方法でパラメータを最適化
  - 最近はもっと多層にするのがはやり (deep learning)
  - 本講義後半で詳しく取り上げる予定



# まとめ

- 一般化線形モデル
  - 出力変数（の誤差）の確率密度分布によって分類される
  - 使いどころ（代表的なタスク）は覚えよう
- ニューラルネットワーク
  - 単層パーセプトロンは一般化線形モデルと密接に関係
- 次回
  - 時系列分析
  - 課題発表