

機械学習：レポート

回帰問題：

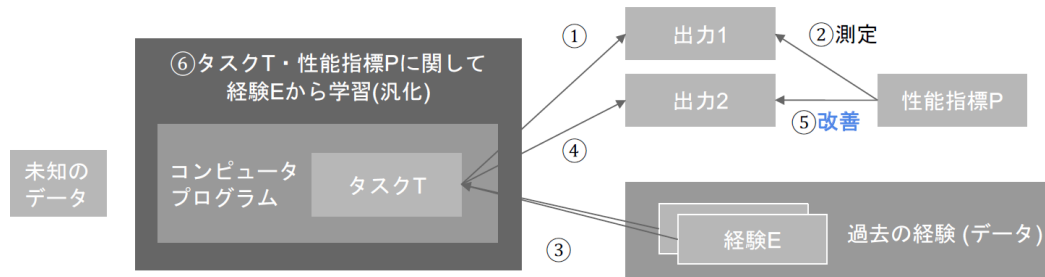
ある入力(離散あるいは連続値)から出力(連続値)を予測する問題

- ・直線で予測する場合が線形回帰
- ・非線形（曲線など）で予測する場合が非線形回帰

機械学習：

トム・ミッチェルは著書「Machine Learning」にて下記のように記載している。

コンピュータプログラムは、タスク T(アプリケーションにさせたいこと)を性能指標 P で測定し、その性能が経験 E(データ)により改善される場合、タスク T および性能指標 P に関して経験 E から学習する。



また、機械学習の父とされるアーサー・サミュエルによる定義は下記となる。

明示的にプログラムしなくても学習する能力をコンピュータに与える研究分野

機械学習には主に下記2つの学習分類がある。

- ・教師あり学習
- ・教師なし学習

その他、教科学習、半教師あり学習なども聴くことがあるが、本章では上記2分類となる。

線形回帰モデル

- ・ざっくり比例
- ・回帰問題を解くための機械学習モデルの一つとなる。
- ・教師あり学習に属する
- ・ランキング問題は、回帰問題で解くべきではない。[\(密度比推定\)](#)

説明変数 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T \in \mathbb{R}^m$

目的変数 $y \in \mathbb{R}^1$

- ・ 入力と m 次元パラメータの線形結合を出力するモデルとなる。(下記式)
- ・ 線形結合 (入力とパラメータの内積)

パラメータ $\mathbf{w} = (w_1, w_2, \dots, w_m)^T \in \mathbb{R}^m$

線形結合 $\hat{y} = \mathbf{w}^T \mathbf{x} + w_0 = \sum_{j=1}^m w_j x_j + w_0$

家賃予測モデルなどが線形回帰モデルとなる。

- ・ データへの仮定
データは回帰直線に誤差が加わり観測されている。
(モデルで仮定した説明変数以外の要素も含めて誤差に表れる)

モデル数式

$$y = w_0 + w_1 x_1 + \varepsilon$$

目的変数 切片 回帰係数 説明変数 誤差

- ・ データの分割
モデルの汎化性能を測定するために必要
学習データと検証データが同一の場合未知のデータに対する精度がわからない。
 - 学習用データ：機械学習モデルの学習に利用するデータ
 - 検証用データ：学習済みモデルの汎化性能を測るための検証データ
- ・ 線形回帰モデルのパラメータは最小二乗法で推定
 - 平均二乗誤差 (残差平方和) . . . 外れ値に弱い (データの確認要)
データとモデル出力の二乗誤差の和

$$\sum_i (\hat{y}_i - y_i)^2 = \sum_i \varepsilon_i^2$$

↑
二乗誤差
総和

- 最小二乗法
学習データの平均二乗誤差を最小とするパラメータを探索

$$\underset{\mathbf{J}(\mathbf{w})}{\text{MSE}}_{\text{train}} = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} \left(\underbrace{\hat{y}_i^{(\text{train})}}_{\mathbf{w}^T \mathbf{x}_i^{(\text{train})}} - y_i^{(\text{train})} \right)^2$$

※ 線形回帰モデル (※ arg ; 条件に合う入力値)

$$\hat{w} = \arg \min_{w \in \mathbb{R}^{m+1}} \text{MSE}_{\text{train}} \quad \frac{\partial}{\partial w} \text{MSE}_{\text{train}} = 0$$

MSEを最小にするようなw(m次元)

MSEをwに関して微分したものが0となるwの点を求める

上記式より、下記展開が行われる。

$$\begin{aligned} &\Rightarrow \frac{\partial}{\partial w} \left\{ \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} (\hat{y}_i^{(\text{train})} - y_i^{(\text{train})})^2 \right\} = 0 \\ &\Rightarrow \frac{\partial}{\partial w} \left\{ \frac{1}{n_{\text{tr}}} \sum_{i=1}^{n_{\text{tr}}} (\underline{x_i^T w} - y_i)^2 \right\} = 0 \\ &\Rightarrow \frac{\partial}{\partial w} \left\{ \frac{1}{n_{\text{tr}}} (Xw - y)^T (Xw - y) \right\} = 0 \\ &\Rightarrow \frac{\partial}{\partial w} \left\{ \frac{1}{n_{\text{tr}}} \underbrace{(Xw - y)^T (Xw - y)} \right\} = 0 \\ &\quad \begin{array}{ccc} X & w & y \\ \begin{pmatrix} 1 & x_{11} & \dots & x_{1m} \\ 1 & x_{21} & \dots & x_{2m} \\ \vdots & \vdots & & \vdots \\ 1 & x_{n1} & \dots & x_{nm} \end{pmatrix} & \begin{pmatrix} w_0 \\ w_1 \\ \vdots \\ w_m \end{pmatrix} & \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \\ n \times (m+1) & (m+1) \times 1 & n \times 1 \end{array} \\ &\Rightarrow \frac{1}{n_{\text{tr}}} \cdot \frac{\partial}{\partial w} \left\{ (w^T X^T - y^T) (Xw - y) \right\} = 0 \\ &\Rightarrow \frac{1}{n_{\text{tr}}} \cdot \frac{\partial}{\partial w} \left\{ \underbrace{w^T X^T X w}_{\text{wの2次項}} - \underbrace{w^T X^T y}_{\text{wの1次項}} - \underbrace{y^T X w}_{\text{wの1次項}} + \underbrace{y^T y}_{\text{定数項}} \right\} = 0 \\ &\Rightarrow \frac{1}{n_{\text{tr}}} \cdot \frac{\partial}{\partial w} \left\{ \underbrace{w^T X^T X w}_{\text{wの2次項}} - 2 \underbrace{w^T X^T y}_{\text{wの1次項}} + y^T y \right\} = 0 \\ &\Rightarrow \frac{1}{n_{\text{tr}}} \left\{ 2X^T X w - 2X^T y \right\} = 0 \\ &\Rightarrow 2X^T X w = 2X^T y \\ &\Rightarrow \underbrace{(X^T X)^{-1}}_{A^{-1}A=I(\text{単位行列})} (X^T X) w = (X^T X)^{-1} X^T y \\ &\Rightarrow w = (X^T X)^{-1} X^T y // \end{aligned}$$

行列の微分で押さえておくべき式

$$\begin{aligned} \textcircled{1} \quad &\frac{\partial (w^T x)}{\partial w} = x \\ &\frac{\partial (w^T A w)}{\partial w} = (A + A^T) \cdot x \\ &\quad = 2Ax \quad (A: \text{対称行列}) \end{aligned}$$

線形回帰モデルのまとめ

回帰係数

$$\hat{\mathbf{w}} = (X^{(\text{train})T} X^{(\text{train})})^{-1} X^{(\text{train})T} \mathbf{y}^{(\text{train})}$$

予測値

$$\hat{\mathbf{y}} = X(X^{(\text{train})T} X^{(\text{train})})^{-1} X^{(\text{train})T} \mathbf{y}^{(\text{train})}$$

$n_{\text{new}} \times m+1$

$\hat{\mathbf{y}} = X_* \hat{\mathbf{w}} = X_* (X_*^T X_*)^{-1} X_*^T \mathbf{y}$

予測したい
新たな点
(n_* 個)

$n_* \times (m+1)$

$(m+1) \times 1$

$X_* = \begin{pmatrix} 1 & x_{11}^* & \dots & x_{1m}^* \\ 1 & x_{21}^* & \dots & x_{2m}^* \\ \vdots & \vdots & \ddots & \vdots \end{pmatrix}$

※上記式の \mathbf{y} (\mathbf{y} ハットではなく) の左は射影行列

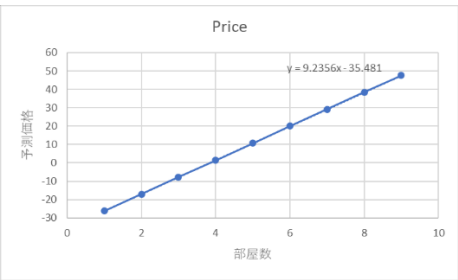
コード演習 (線形回帰モデル: skl_regression.ipynb)

線形回帰モデル-Boston Housing Data-

```
1 # 単回帰の回帰係数と切片を出力
2 print('推定された回帰係数: %.3f, 推定された切片: %.3f' % (model.coef_, model.intercept_))
3
4 for i in range(1, 10):
5     price=model.predict([[i]])
6     print("Room = ", i, "予測値 = ", model.predict([[i]]))
```

推定された回帰係数: 9.236, 推定された切片: -35.481

Room = 1 予測値 = [-26.24530476]
Room = 2 予測値 = [-17.00970322]
Room = 3 予測値 = [-7.77410166]
Room = 4 予測値 = [1.4614939]
Room = 5 予測値 = [10.63710146]
Room = 6 予測値 = [19.93270302]
Room = 7 予測値 = [29.16830459]
Room = 8 予測値 = [38.40390615]
Room = 9 予測値 = [47.63950771]



予測結果は上記となり、4 部屋以下ではマイナス予測となる結果。

これは、Boston Housing Data で RM の最小が 3.561 部屋となるため。

[16] 1 df.describe()

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	PRICE
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	00.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

非線形回帰モデル

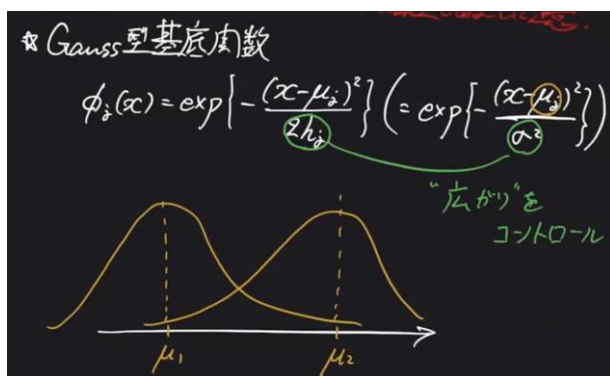
複雑な非線形構造を内在する現象に対して、非線形回帰モデリングを実施

・基底展開法

- 回帰関数として、基底関数と呼ばれる既知の非線形関数とパラメータベクトルの線型結合を使用
- 未知パラメータは線形回帰モデルと同様に最小2乗法や最尤法により推定

・よく使われる基底関数

- 多項式関数
- ガウス型基底関数



非線形回帰モデル

説明変数	$\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{im}) \in \mathbb{R}^m$	説明変数の数
非線形関数ベクトル	$\phi(\mathbf{x}_i) = (\phi_1(\mathbf{x}_i), \phi_2(\mathbf{x}_i), \dots, \phi_k(\mathbf{x}_i))^T \in \mathbb{R}^k$	基底関数の数
非線形関数の計画行列	$\Phi^{(train)} = (\phi(\mathbf{x}_1), \phi(\mathbf{x}_2), \dots, \phi(\mathbf{x}_n))^T \in \mathbb{R}^{n \times k}$	
最尤法による予測値	$\hat{\mathbf{y}} = \Phi(\Phi^{(train)T}\Phi^{(train)})^{-1}\Phi^{(train)T}\mathbf{y}^{(train)}$	

基底展開法も線形回帰と同じ枠組みで推定可能

$$\text{例: } \mathbf{y} = \mathbf{X}\mathbf{w} \rightarrow \text{例: } \mathbf{y} = \Phi\mathbf{w}$$

$$\begin{matrix} n \times 1 & n \times (m+1) & n \times 1 & n \times (m+1) \\ & (m+1) \times 1 & & (m+1) \times 1 \end{matrix}$$

結局, MSEを最小化するwは先と同じ様に,

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

$$\rightarrow \hat{\mathbf{w}} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{y}$$

$$\therefore \hat{\mathbf{y}} = \Phi_*\hat{\mathbf{w}} = \Phi_*(\Phi^T\Phi)^{-1}\Phi^T\mathbf{y}$$

・ 未学習(underfitting)と過学習(overfitting)

- 学習データに対して、十分小さな誤差が得られないモデル→未学習
(対策)モデルの表現力が低いため、表現力の高いモデルを利用する
- 小さな誤差は得られたけど、テスト集合誤差との差が大きいモデル→過学習
(対策 1) 学習データの数を増やす
(対策 2) 不要な基底関数(変数)を削除して表現力を抑止
(対策 3) 正則化法を利用して表現力を抑止

— 正則化法 —

予測: $\hat{y} = X_* (X^T X)^{-1} X^T y$

$X = \begin{pmatrix} 1 & 2 & 4 \\ 1 & 3 & 5.9 \\ 1 & 4 & 8.1 \end{pmatrix} \Rightarrow (X^T X)^{-1}$ の要素は
 \nearrow \times が \times 多くなる...
 \nwarrow ほぼ平行...

罰則項
 $E(w) = J(w) + \lambda \cdot w^T w$ ($w \rightarrow \infty$ で 罰則 $\rightarrow \infty$)
 \hookrightarrow MSE (コレが小さくなるよ! w も考えろ!)

正則化項 (罰則項) の役割は

- リッジ推定 (L2 ノルム) . . . パラメータを 0 に近づけるように推定
- ラッソ推定 (L1 ノルム) . . . いくつかのパラメータを正確に 0 に推定

解きたいのは, $\min MSE \quad s.t. \quad R(w) \leq t$

\downarrow 条件を満たしつつ...

(最適化): KKT条件より,
 $\min MSE + \lambda \cdot R(w)$ "オメガ"を付ける
 不等式条件を回避

正則化の効かせ方はハイパーパラメータとなる。

・ ホールドアウト法 . . . 1 回のみ精度検証

有限のデータを学習用とテスト用の 2 つに分割し、「予測精度」や「誤り率」を推定する為に使用

※欠点: データが少ない場合に、外れ値にモデルがフィッティングするケース。

・ クロスバリデーション (交差検証) . . . 分割分精度検証がある

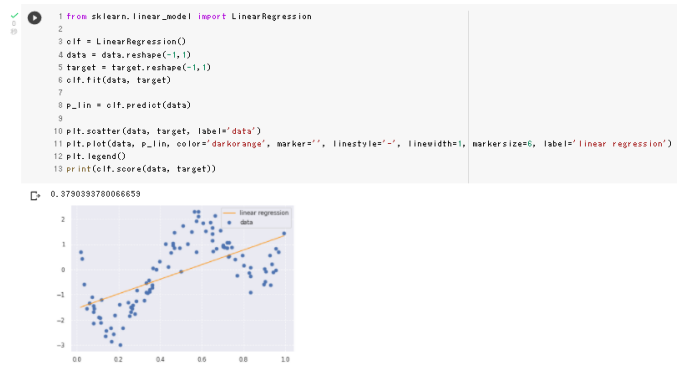


・グリッドサーチ

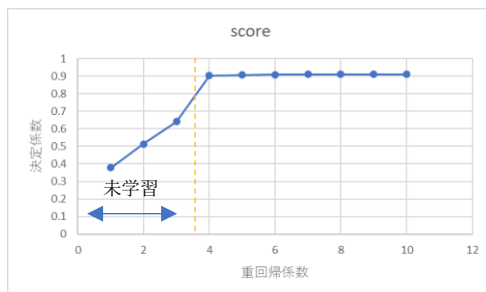
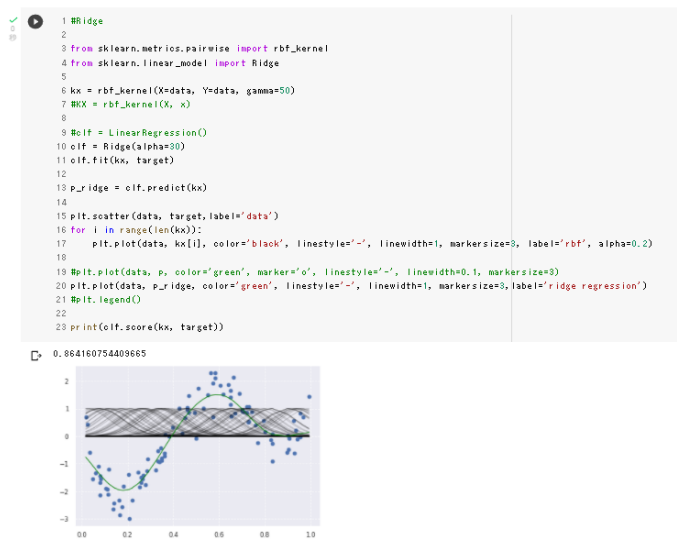
ハイパーパラメータの候補を用意し、全ての組み合わせで評価値を算出

コード演習（非線形回帰モデル：skl_nonlinear regression.ipynb）

線形回帰の場合、正答率（決定係数）は 0.379 と低い



非線形回帰（リッジ推定 $\alpha=30$ あり）の場合、正答率は 0.864



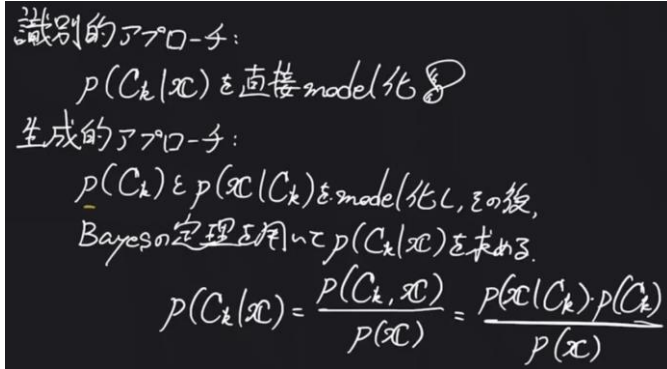
線形回帰では表現できない回帰が非線形回帰では可能となっている。

※モデル表現は適切な設定が必要となる。

ロジスティック回帰モデル

・ 分類問題(クラス分類)

ある入力(数値)からクラスに分類する問題



参考文献：



・ 分類で扱うデータ

入力 (m 次元のベクトル)

出力 (0 or 1)

説明変数 $\mathbf{x} = (x_1, x_2, \dots, x_m)^T \in \mathbb{R}^m$

目的変数 $y \in \{0, 1\}$ ← 0か1

目的変数は 0/1 に対し、入力は実数のためシグモイド関数などを用い、0~1 へ圧縮する。

・ シグモイド関数

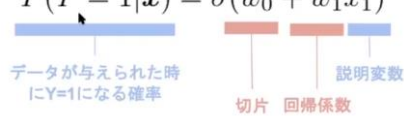
入力の実数・出力は必ず 0~1 の値 (確率を表現できる)

シグモイド関数の微分は、シグモイド関数自身で表現することが可能

$$\sigma(x) = \frac{1}{1 + \exp(-ax)}$$

- ・シグモイド関数の出力を $Y=1$ になる確率に対応させる

数式

$$P(Y=1|\mathbf{x}) = \sigma(w_0 + w_1 x_1)$$


データが与えられた時に $Y=1$ になる確率

切片 回帰係数

説明変数

データ Y は確率が 0.5 以上なら 1, 未満なら 0 と予測するのが一般的
求める精度により 0.5 は調整する。

- ・最尤推定

- ベルヌーイ分布

$$Y \sim \text{Ber}(p) \quad P(y) = p^y (1-p)^{1-y}$$

- ・最尤推定

尤度関数を最大化するようなパラメータを選ぶ推定方法

1回の試行で $y=y_1$ になる確率 $P(y) = p^y (1-p)^{1-y}$: 既知 : 未知

n回の試行で $y_1 \sim y_n$ が同時に起こる確率(p 固定) $P(y_1, y_2, \dots, y_n; p) = \prod_{i=1}^n p^{y_i} (1-p)^{1-y_i}$

$y_1 \sim y_n$ のデータが得られた際の尤度関数 $P(y_1, y_2, \dots, y_n; p) = \prod_{i=1}^n p^{y_i} (1-p)^{1-y_i}$

- ・尤度関数を最大とするパラメータを探す(推定)

確率は 0~1 と小さいため、複数をかけると 0 に近くなる。

そこを解決するために log をとっている。また、対数をとると微分の計算が簡単

$$\begin{aligned} E(w_0, w_1, \dots, w_m) &= -\log L(w_0, w_1, \dots, w_m) \\ &= -\sum_{i=1}^n \{y_i \log p_i + (1-y_i) \log(1-p_i)\} \end{aligned}$$

- ・ロジスティック回帰モデルの学習

対数尤度関数をパラメータで微分して 0 になる値を求める (解析的に求めることは困難)

- 勾配降下法 (Gradient descent)

学習データをまとめて反復学習により逐次更新するアプローチ

η は学習率と呼ばれるハイパーパラメータ (収束しやすさを調整)

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta \sum_{i=1}^n (y_i - p_i) \mathbf{x}_i$$

- 確率的勾配降下法(SGD)

データを一つずつランダムに(「確率的」に)選んでパラメータを更新

$$\mathbf{w}^{(k+1)} = \mathbf{w}^{(k)} + \eta (y_i - p_i) \mathbf{x}_i$$

コード演習 (ロジスティック回帰モデル: skl_logistic_regression.ipynb)

1. チケット価格から生死を判別

predict_proba(X): [データ数]行 × [次元数]列の特徴量行列 X を引数にして、
各データがそれぞれのクラスに所属する確率を返す

```
[63]: 1 for i in range(0,10):
      2     predic_val1 = model.predict([[i]])
      3     predic_val2 = model.predict_proba([[i]])
      4     print(f'{predic_val1}')
```

[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]
[0]

```
1 for i in range(0,10):
2     predic_val1 = model.predict([[i]])
3     predic_val2 = model.predict_proba([[i]])
4     print(f'{predic_val2}')
```

[[0.7193658 0.2806342]]
[[0.71628772 0.28371228]]
[[0.71318934 0.28681066]]
[[0.71007081 0.28992919]]
[[0.70693231 0.29306769]]
[[0.70377401 0.29622599]]
[[0.70059609 0.29940391]]
[[0.69739873 0.30260127]]
[[0.69418214 0.30581786]]
[[0.6909465 0.3090535]]

predict では 0/1 分類結果となるが、predict_proba では 0/1 分類の確率が返り値となる。



確率 0.5 にて予測が 0/1 切り替わる。

主成分分析

変数間に相関があるデータに対して有効な、変数削減手法となる。

多変量データの持つ構造をより少数個の指標に圧縮

- 変数の個数を減らすことに伴う、情報の損失はなるべく小さくしたい
- 少数変数を利用した分析や可視化(2・3次元の場合)が実現可能

具体的な対応は、情報の量を分散の大きさと捉え、分散最大方向を射影軸として探索する。

条件付き最適化問題を解く

- ノルムが1となる制約を入れる(制約を入れないと無限に解がある)

目的関数

$$\arg \max_{\mathbf{a} \in \mathbb{R}^m} \mathbf{a}_j^T \text{Var}(\bar{X}) \mathbf{a}_j$$

制約条件

$$\mathbf{a}_j^T \mathbf{a}_j = 1$$

- ラグランジュ関数を最大にする係数ベクトルを探索(微分して0になる点)
分散共分散行列の固有値と固有ベクトルが、制約付き最適化問題の解となる

ラグランジュ乗数

ラグランジュ関数

$$E(\mathbf{a}_j) = \mathbf{a}_j^T \text{Var}(\bar{X}) \mathbf{a}_j - \lambda(\mathbf{a}_j^T \mathbf{a}_j - 1)$$

目的関数

制約条件

微分

$$\frac{\partial E(\mathbf{a}_j)}{\partial \mathbf{a}_j} = 2\text{Var}(\bar{X}) \mathbf{a}_j - 2\lambda \mathbf{a}_j = 0$$



解

$$\text{Var}(\bar{X}) \mathbf{a}_j = \lambda \mathbf{a}_j$$

- 射影先の分散は固有値と一致

$$\text{Var}(\mathbf{s}_1) = \mathbf{a}_1^T \text{Var}(\bar{X}) \mathbf{a}_1 = \lambda_1 \mathbf{a}_1^T \mathbf{a}_1 = \lambda_1$$

コード演習 (PCA モデル: skl_pca.ipynb)

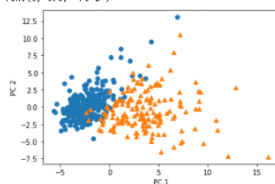
○ 乳がん検査データを利用しロジスティック回帰モデルを作成

```
Train score: 0.988
Test score: 0.972
Confusion matrix:
[[89  1]
 [ 3 50]]
```

○ 主成分を利用し2次元空間上に次元圧縮

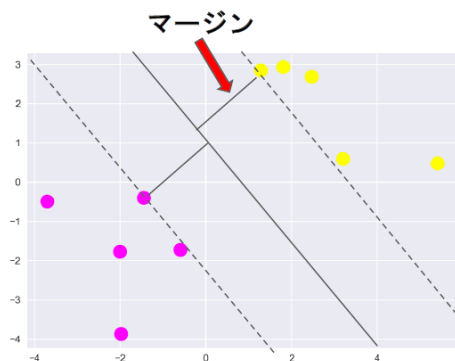
```
1 # PCA
2 # 次元数2まで圧縮
3 pca = PCA(n_components=2)
4 X_train_pca = pca.fit_transform(X_train_scaled)
5 print('X_train_pca shape: {}'.format(X_train_pca.shape))
6 # X_train_pca shape: (428, 2)
7
8 # 寄与率
9 print('explained variance ratio: {}'.format(pca.explained_variance_ratio_))
10 # explained variance ratio: [ 0.43315126  0.19585506]
11
12 # 散布図にプロット
13 temp = pd.DataFrame(X_train_pca)
14 temp['Outcome'] = y_train.values
15 b = temp[temp['Outcome'] == 0]
16 a = temp[temp['Outcome'] == 1]
17 plt.scatter(x=b[0], y=b[1], markers='o') # 最良は○でマーク
18 plt.scatter(x=a[0], y=a[1], markers='^') # 悪化は△でマーク
19 plt.xlabel('PC 1') # 第1主成分をx軸
20 plt.ylabel('PC 2') # 第2主成分をy軸
```

```
X_train_pca shape: (428, 2)
explained variance ratio: [0.43315126 0.19585506]
Text(0, 0.5, 'PC 2')
```



サポートベクターマシン

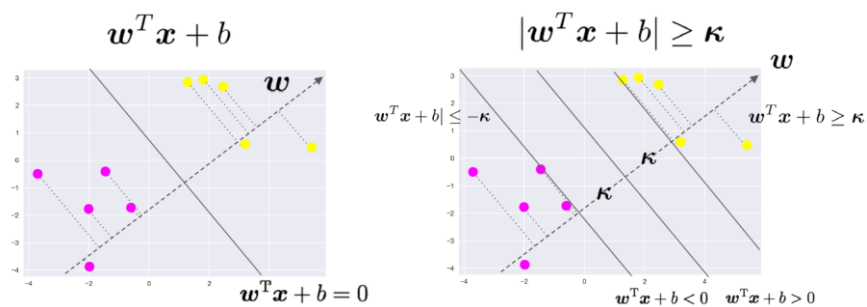
サポートベクターマシンはデータからできるだけ離れた線形判別関数（決定境界）を求める
 → マージン最大化（マージンをパラメータに依存する関数として表現している。）



・ 目的関数の導出(準備)

各クラスのデータを w の方向 k へ射影した点 w 軸に座標を変換

線形判別関数のマージンを κ とした時に全てのデータ点でなりたつ条件



$$\begin{aligned} \rho(w, b) &= \min_{x \in C_{t_i=+1}} \frac{w^T x_i}{\|w\|} - \max_{x \in C_{t_i=-1}} \frac{w^T x_i}{\|w\|} \\ &= \frac{1 - b}{\|w\|} - \frac{-1 - b}{\|w\|} \\ &= \frac{2}{\|w\|} \end{aligned}$$

W が決まればマージンが決まる。

マージンとは決定境界と最も距離の近い点との距離なので、

$$\min_i \frac{t_i(w^T x_i + b)}{\|w\|}$$

SVM はマージンを最大化することを目標なので目的関数は

$$\max_{w, b} \left[\min_i \frac{t_i(w^T x_i + b)}{\|w\|} \right]$$

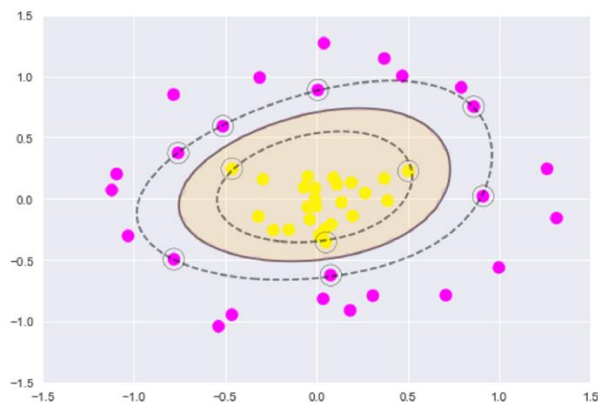
マージンが最大となる直線(パラメータ)を探す

- ・ ハードマージン SVM
 - マージンの内側にデータが存在することを許容しない
- ・ ソフトマージン SVM
 - サンプルを線形分離できないときに使用
 - 誤差を許容し、誤差に対してペナルティを与える
- ・ 非線形分離
 - 線形分離できないとき
 - 特徴空間に写像し、その空間で線形に分離する
- ・ カーネルトリック
 - カーネル関数を使用し、高次元ベクトルの内積をスカラー関数で表現

$$k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$$

- 特徴空間が高次元でも計算コストを抑えられる
- ・ 非線形カーネルを用いた分離
 - 放射基底関数カーネル（RBF カーネル・ガウシアンカーネル）

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right)$$



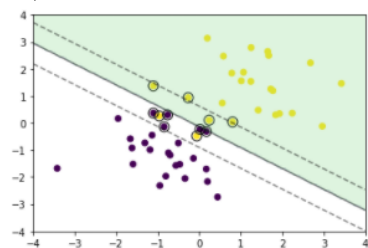
元データを高次元からの射影だと考え高次元な空間で決定境界を得る。

コード演習 (SVM モデル: np_svm.ipynb)

ソフトマージン (マージン内に重なりあり)

```
[19] 1 # 訓練データを可視化
2 plt.scatter(x_train[:, 0], x_train[:, 1], c=y_train)
3 # サポートベクトルを可視化
4 plt.scatter(support_vectors[:, 0], support_vectors[:, 1],
5             s=100, facecolors='none', edgecolors='k')
6 # 領域を可視化
7 plt.contourf(xx0, xx1, y_pred.reshape(100, 100), alpha=0.2, levels=np.linspace(0, 1, 3))
8 # マージンと決定境界を可視化
9 plt.contour(xx0, xx1, y_project.reshape(100, 100), colors='k',
10            levels=[-1, 0, 1], alpha=0.5, linestyle=['--', '-', '--'])
```

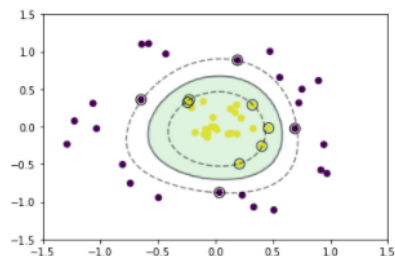
<matplotlib.contour.QuadContourSet at 0x7f48a0b70c50>



RFB カーネル (ガウシアンカーネル) を利用した SVM

```
[13] 1 # 訓練データを可視化
2 plt.scatter(x_train[:, 0], x_train[:, 1], c=y_train)
3 # サポートベクトルを可視化
4 plt.scatter(support_vectors[:, 0], support_vectors[:, 1],
5             s=100, facecolors='none', edgecolors='k')
6 # 領域を可視化
7 plt.contourf(xx0, xx1, y_pred.reshape(100, 100), alpha=0.2, levels=np.linspace(0, 1, 3))
8 # マージンと決定境界を可視化
9 plt.contour(xx0, xx1, y_project.reshape(100, 100), colors='k',
10            levels=[-1, 0, 1], alpha=0.5, linestyle=['--', '-', '--'])
```

<matplotlib.contour.QuadContourSet at 0x7f48a0c481d0>



加筆（参考文献より）

1. 線形回帰

単回帰：独立な説明変数が 1 つの場合

重回帰：独立な説明変数が 2 つ以上の場合

多項式回帰：説明変数のべき乗で表される線形回帰

多項式回帰は、説明変数 x_1 に対して線形ではないが線形回帰となる。

2. 非線形回帰

$y = \exp\{W_1 X_1\}$ 、 $y = 1/(W_1 X_1 + 1)$ のような学習パラメータ W と目的関数 y との関係が線形になっていないものを非線形回帰と呼ぶ。

3. ロジスティック回帰

ロジスティック回帰では分類結果が切り替わる境界を決定境界と呼び、学習する。

各特徴量の係数を見ることができ、解釈性がある回帰となる。

4. 主成分分析

主成分の選び方：元データの変数を新たな軸となる主成分で表現する。

この主成分は、第一主成分、第二主成分、・・・と寄与率の大きい順に並ぶ。

データと相関のないデータを主成分とした場合寄与率がほぼ変わらず次元削減が進まない。

5. SVM

カーネル補足

(a) 線形カーネル・・・線形サポートベクターマシンと等価

(b) シグモイドカーネル・・・カーネル関数にシグモイド関数を使用
曲がった境界を学習できる。

(c) 多項カーネル（2 次）では円形の境界を学習できる。

(d) RBF カーネル・・・上記に対しより複雑な境界を学習できる。

カーネル法を利用すると、特徴量が何であることを明示的にすることはできなくなる。

そのため特徴量の解釈より精度の要求がある場合に利用することが望ましい。

参考文献

機械学習のエッセンス（加藤 公一） ※E 資格の範囲と一致しているとのこと。

見て試してわかる機械学習アルゴリズムの仕組み 機械学習図鑑（秋庭 伸也ほか）