# 機械学習・要点まとめ

## 第１章 線形回帰モデル

### ▣ 線形回帰モデル

以下の線形結合において、最小二乗法によりパラメータ $w_i$ を推定

$$\hat{y} = w^T x + w_0 = \sum_{j=1}^{m} w_j x_j + w_0 \quad (m：パラメータ数、説明変数の数)$$

回帰係数 $\hat{w} = (X^{(train)T} X^{(train)})^{-1} X^{(train)T} y^{(train)}$

予測値 $\hat{y} = X(X^{(train)T} X^{(train)})^{-1} X^{(train)T} y^{(train)}$

$n \times (m+1)$ (n：データ数)

### ▣ 線形回帰モデルのハンズオン

- sklearn の場合： linear_model の LinearRegression を用いる。fit 関数でパラメータを推定、predict 関数で予測。 ⇒ サマリー・考察は、skl_regression.ipynb に記載
- numpy の場合： $\hat{w_1} = Cov[x,y]/Var[x]$, $\hat{w_0} = \mu_y - \hat{w_1}\mu_x$ (Cov：共分散、Var：分散、μ：平均) により、パラメータを推定する実装を行う。

## 第２章 非線形回帰モデル

### ▣ 非線形回帰モデル

- **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ーメータベクトルの線形結合で表現。未知パラメータは、線形回帰モデルと同様に最小二乗法や最尤法により推定。

$$\hat{y}_i = w^T \phi(x_i) + w_0 = \sum_{j=1}^{m} w_j \varphi_j(x_i) + w_0$$

説明変数 $\boldsymbol{x_i} = (x_{i1}, x_{i2}, \cdots, x_{im}) \in R^m$

非線形関数ベクトル $\phi(x_i) = (\varphi_1(x_i), \varphi_2(x_i), \cdots, \varphi_k(x_i)) \in R^k$

非線形関数の計画行列 $\Phi^{(train)} = (\phi(x_1), \phi(x_2), \cdots, \phi(x_n)) \in R^{n \times k}$

最尤法による予測 $\hat{y} = \Phi (\Phi^{(train)T} \Phi^{(train)})^{-1} \Phi^{(train)T} y^{(train)}$

- よく使われる基底関数：

多項式関数 $\varphi_j = x^j$ ガウス型基底関数 $\varphi_j(x) = \exp\left\{(x - \mu_j)^2 / 2h_j\right\}$

- **不要な基底関数を削除**： 基底関数の数、位置やバンド幅によりモデルの複雑さが変化。多くの基底関数を用意してしまうと過学習が起こるため、適切な基底関数を用意 (交差検証 CV 等で選択)。
- **正則化法 (罰則化法)**：「モデルの複雑さに伴って、その値が大きくなる正則化項 (罰則項) を課した関数」を最小化。

$$S_\gamma = \underset{n \times k}{(y - \Phi w)^T (y - \Phi w)} + \gamma R(w) \quad (\gamma > 0)$$

基底関数の数 (k) が増加するとパラメータが増加し、残差は減少 (モデルが複雑化)

モデルの複雑さに伴う罰則

### ▣ 非線形回帰モデルのハンズオン

- サマリー・考察は、skl_nonlinear_regression.ipynb に記載

# 機械学習・要点まとめ

## 第３章 ロジスティック回帰モデル

### ▣ ロジスティック回帰モデル

- **分類問題（クラス分類）**： **対数オッズ** $\log(p/1-p)$ を線形回帰で予測し、それを**正規化**して確率として出力を得ることで、結果としてクラス分類を実現。対数オッズの逆関数が**シグモイド関数**（活性化関数） ⇒ 活性化関数（＝**ロジット変換**）により、クラス i に属する確率 p が求まる。

- 目的関数は、**対数尤度**（回帰モデル構築時に最大化したい**尤度関数**の**対数**をとったもの、その負値が**交差エントロピー**関数）。**交差エントロピー**関数の最小化は、**対数尤度**の最大化と同値。

  説明変数　$x = (x_1, x_2, \cdots, x_m)^T \in R^m$

  目的変数　$y \in \{0, 1\}$

  パラメータ　$w = (w_1, w_2, \cdots, w_m)^T \in R^m$

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

  出力　$\hat{y} = P(Y=1 \mid x) = \sigma(z = \sum_{j=1}^{m} w_j x_j + w_0) = \frac{1}{1+exp(-az)}$　σ：シグモイド関数

  シグモイド関数の出力を Y=1 になる確率に対応させる

  （Y は、$\hat{y} \geqq 0.5$ なら 1、$\hat{y} < 0.5$ なら 0 と予測）

- **ロジスティック回帰モデルの最尤推定**： 対数尤度関数 E を最大とするパラメータを探索

  $y_1 \sim y_n$ のデータが得られた際の**尤度関数** $L(\mathbf{w})$（確率 P はベルヌーイ分布に従うものとする）：

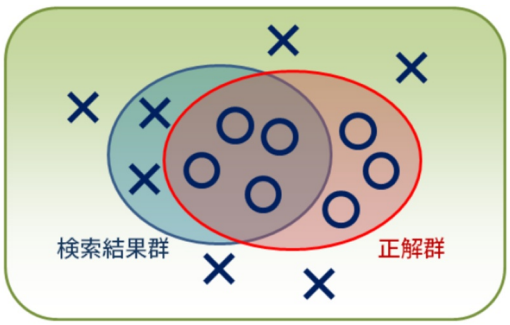
  $L(\mathbf{w}) = P(y_1, y_2, \cdots, y_n \mid w_0, w_1, \cdots, w_m) = \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} = \prod_{i=1}^{n} \sigma(w^T x_i)^{y_i}(1-\sigma(w^T x_i))^{1-y_i}$

  交差エントロピー　$E(w) = -\log L(w) = \sum_{i=1}^{n}\{y_i \log p_i + (1-y_i)\log(1-p_i)\}$

  $\frac{\partial E(w)}{\partial w} = -\sum_{i=1}^{n}(y_i - p_i)\, x_i$　より、

  勾配降下法　$w^{(k+1)} = w^{(k)} - \eta \frac{\partial E(w)}{\partial w} = w^{(k)} + \eta \sum_{i=1}^{n}(y_i - p_i)\, x_i$

### ▣ 適合率と再現率

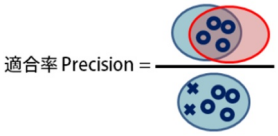


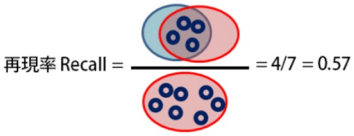

混同行列：真陽性（TP）、偽陽性（FP）、偽陰性（FN）、真陰性（TN）

正解率 accuracy＝（TP＋TN）／全体＝9／14＝0.64

適合率 precision＝TP／（TP＋FP）　⇔　偽陽性（FP）

再現率 recall＝TP／（TP＋FN）　⇔　偽陰性（FN）




### ▣ ロジスティック回帰モデルのハンズオン

- サマリー・考察は、skl_logistic_regression.ipynb ，np_logistic_regression.ipynb に記載

# 機械学習・要点まとめ

## 第４章 主成分分析

### ▣ 主成分分析

- **主成分分析**（PCA；Principal Component Analysis）： 教師なし学習で、多変量データの持つ構造をより少数個の指標に圧縮しつつそれに伴う情報の損失はなるべく小さくするため、変換後の分散を最大にするように探索する。

  学習データ　$\boldsymbol{x_i} = (x_{i1},\ x_{i2}, \cdots,\ x_{im}) \in \mathrm{R}^m$

  平均（ベクトル）　$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$

  データ行列　$\bar{X} = (x_1 - \bar{x},\ \cdots,\ x_n - \bar{x})^{\mathrm{T}}$

  分散共分散行列　$\Sigma = \mathrm{Var}(\bar{X}) = \frac{1}{n}\bar{X}^{\mathrm{T}}\bar{X}$

  線形変換後のベクトル　$s_j = (s_{1j},\ s_{2j}, \cdots, s_{nj})^{\mathrm{T}} = \bar{X}\boldsymbol{a_j}$　（$\bar{X}$: n×m, $\boldsymbol{a_j}$ :m×1）　j は射影後のインデックス

  $\mathrm{Var}(s_j) = \frac{1}{n}s_j{}^{\mathrm{T}}s_j = \frac{1}{n}(\bar{X}\boldsymbol{a_j})^{\mathrm{T}}(\bar{X}\boldsymbol{a_j}) = \frac{1}{n}\boldsymbol{a_j}{}^{\mathrm{T}}\bar{X}^{\mathrm{T}}\bar{X}\boldsymbol{a_j} = \boldsymbol{a_j}{}^{\mathrm{T}}\mathrm{Var}(\bar{X})\boldsymbol{a_j}$

  線形変換後の変数の分散が最大となる射影軸は、以下のラグランジュ関数を最大にする：

  $E(\boldsymbol{a_j}) = \boldsymbol{a_j}{}^{\mathrm{T}}\,\mathrm{Var}(\bar{X})\,\boldsymbol{a_j} - \lambda(\boldsymbol{a_j}{}^{\mathrm{T}}\boldsymbol{a_j} - 1)$

  $\frac{\partial E(\boldsymbol{a_j})}{\partial \boldsymbol{a_j}} = 2\mathrm{Var}(\bar{X})\boldsymbol{a_j} - 2\lambda\boldsymbol{a_j} = 0 \ \Rightarrow\ \mathrm{Var}(\bar{X})\boldsymbol{a_j} = \lambda\boldsymbol{a_j}$

  $\mathrm{Var}(s_j) = \boldsymbol{a_j}{}^{\mathrm{T}}\mathrm{Var}(\bar{X})\boldsymbol{a_j} = \lambda_j\boldsymbol{a_j}{}^{\mathrm{T}}\boldsymbol{a_j} = \lambda_j$　射影先の分散は固有値と一致

- **寄与率**： 各成分の重要度が測れる

  第1～元次元分の主成分の分散は固有値と等しく、元のデータの分散と一致

  元データの総分散　$\mathrm{V_{total}} = \sum_{i=1}^{m}\lambda_i$

  **寄与率**　$c_k$：　第 k 主成分の分散の全分散に対する割合（第 k 主成分が持つ情報量の割合）

  **累積寄与率**　$r_k$：　第1～k 主成分まで圧縮した際の情報損失量の割合

  $c_k = \frac{\lambda k}{\sum_{i=1}^{m}\lambda i}$　　$r_k = \frac{\sum_{j=1}^{k}\lambda j}{\sum_{i=1}^{m}\lambda i}$

- オートエンコーダと主成分分析の関係･･･オートエンコーダの中間層の活性化関数を**恒等関数**にすると、オートエンコーダと主成分分析は等価な数式で表すことができ同じ結果を返す。

### ▣ 主成分分析のハンズオン

- サマリー･考察は、skl_pca.ipynb ，np_pca.ipynb　に記載

## 第５章 k 近傍法

### ▣ k 近傍法

- 分類問題で適用：　最近傍のデータを k 個取ってきて、それらが最も多く所属するクラスに識別

  ⇒ k（ハイパーパラメータ）により結果が変わる、k を大きくすると決定境界は滑らかになる

### ▣ k 近傍法のハンズオン

# 機械学習・要点まとめ

- サマリー・考察は、np_knn.ipynb に記載

## 第６章 k-means（k 平均クラスタリング）

### ▣ k-means

- 教師なし学習、分類問題で適用： 最近傍のデータを k 個取ってきて、それらが最も多く所属するクラスに識別 ⇒ k（ハイパーパラメータ）により結果が変わる、k を大きくすると決定境界は滑らかになる

**【学習】**

①各クラスタ中心の初期値を設定する。
②各データ点に対して、各クラスタ中心との距離を計算し、最も距離が近いクラスタを割り当てる
③各クラスタの平均ベクトル（中心）を計算する
④収束するまで②・③の処理を繰り返す

### ▣ k-means のハンズオン

- サマリー・考察は、skl_kmeans.ipynb ，np_ kmeans.ipynb に記載

## 第７章 サポートベクターマシン

### ▣ サポートベクターマシン（線形・非線形共通）

**【学習】**

特徴空間上で線形なモデル$y(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x})+b$ を用い、その正負によって 2 値分類を行うことを考える。
サポートベクトターマシンではマージンの最大化を行うが、それは以下の最適化問題を解くことと同じ。
ただし、訓練データを$X=[\boldsymbol{x}_1,\boldsymbol{x}_2,...,\boldsymbol{x}_n]^{\mathrm{T}}$，$\boldsymbol{t}=[t_1,t_2,...,t_n]^{\mathrm{T}}$ $(t_i=\{-1,+1\})$ とする。

$$\min_{\boldsymbol{w},b} \ \frac{1}{2}\|\boldsymbol{w}\|^2 \ ， \ \text{subject to} \quad t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_i)+b)\geq 1 \ \ (i=1,2,\cdots,n)$$

＜線形の場合は、$\boldsymbol{\phi}(\boldsymbol{x}_i)=\boldsymbol{x}_i$ で、以降、同様に読み替えること＞
ラグランジュ乗数法を使うと、上の最適化問題はラグランジュ乗数$\boldsymbol{a}$ (≥0)を用いて、以下の目的関数を最小化する問題となる。

$$L(\boldsymbol{w},b,\boldsymbol{a})=\frac{1}{2}\|\boldsymbol{w}\|^2-\sum_{i=1}^{n}a_i\,t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_i)+b-1) \quad \cdots \ (1)$$

目的関数が最小となるのは、$\boldsymbol{w}$,$b$に関して偏微分した値が 0 となるときなので、

$$\frac{\partial L}{\partial \boldsymbol{w}}=\boldsymbol{w}-\sum_{i=1}^{n}a_i\,t_i\boldsymbol{\phi}(\boldsymbol{x}_i)=0 \ \ ， \ \ \frac{\partial L}{\partial b}=\sum_{i=1}^{n}a_i\,t_i=0$$

これを式(1) に代入することで、最適化問題は結局以下の目的関数の最大化となる。

$$\tilde{L}(\boldsymbol{a})=\sum_{i=1}^{n}a_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}a_ia_jt_it_j\boldsymbol{\phi}(\boldsymbol{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}_j)=\boldsymbol{a}^{\mathrm{T}}\mathbf{1}-\frac{1}{2}\boldsymbol{a}^{\mathrm{T}}H\boldsymbol{a} \quad （\mathbf{1}：全成分が 1 のベクトル）$$

ただし、行列$H$の$i$行$j$列成分は$H_{ij}=t_it_j\boldsymbol{\phi}(\boldsymbol{x}i)^{\mathrm{T}}\boldsymbol{\phi}(\boldsymbol{x}j)=t_it_jk(\boldsymbol{x}_i,\boldsymbol{x}_j)$である。

また制約条件は、$\boldsymbol{a}^{\mathrm{T}}\boldsymbol{t}=0$ $(\frac{1}{2}\|\boldsymbol{a}^T\boldsymbol{t}\|^2=0)$である。

# 機械学習・要点まとめ

この最適化問題を最急降下法で解く。目的関数と制約条件を$\boldsymbol{a}$で微分すると、

$\frac{d\tilde{L}}{d\boldsymbol{a}}=1-H\boldsymbol{a}$ ， $\frac{d}{d\boldsymbol{a}}(\frac{1}{2}\|\boldsymbol{a}^T\boldsymbol{t}\|^2)=(\boldsymbol{a}^T\boldsymbol{t})\boldsymbol{t}$

従って、$\boldsymbol{a}$を以下の二式で更新する。

$\boldsymbol{a}\leftarrow\boldsymbol{a}+\eta_1(1-H\boldsymbol{a})$ ， $\boldsymbol{a}\leftarrow\boldsymbol{a}-\eta_2(\boldsymbol{a}^{\mathrm{T}}\boldsymbol{t})\boldsymbol{t}$

**【予測】**

新しいデータ点$\boldsymbol{x}$に対しては、$y(\boldsymbol{x})=\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x})+b=\sum_{i=1}^{n}a_i\,t_i k(\boldsymbol{x},\boldsymbol{x}_i)+b$ の正負によって分類する。

ここで、最適化の結果得られた$a_i(i=1,2,...,n)$の中で$a_i=0$ に対応するデータ点は予測に影響を与えないので、$a_i>0$ に対応するデータ点（**サポートベクトル**）のみ保持しておく。$b$はサポートベクトルのインデックスの集合を$S$とすると、$b=\frac{1}{S}\sum_{s\in S}(t_s-\sum_{i=1}^{n}a_i\,t_i k(\boldsymbol{x},\boldsymbol{x}_i))$によって求める。

## ▣ ソフトマージン SVM

**【学習】**

分離不可能な場合は学習できないが、データ点がマージン内部に入ることや誤分類を許容することでその問題を回避する。

スラック変数$\xi_i\geq 0$ を導入し、マージン内部に入った点や誤分類された点に対しては、$\xi_i=|1-t_i y(\boldsymbol{x}_i)|$とし、これらを許容する代わり、ペナルティを与えるように、最適化問題を以下のように修正する。

$\min_{\boldsymbol{w},b}\ \frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{n}\xi_i$

subject to $ti(\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x}_i)+b)\geq 1-\xi i\ \ (i=1,2,\cdots,n)$

ただし、パラメータ$C$はマージンの大きさと誤差の許容度のトレードオフを決めるパラメータである。

この最適化問題にラグランジュ乗数法などを用いると、結局最大化する目的関数はハードマージン SVM と同じになる。

$\tilde{L}(\boldsymbol{a})=\sum_{i=1}^{n}a_i-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}a_i a_j t_i t_j\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)$

ただし、制約条件が$a_i\geq 0$ の代わりに $0\leq a_i\leq C\ (i=1,2,\cdots,n)$ となる。（ハードマージン SVM と同じ $\sum_{i=1}^{n}a_i\,t_i=0$ も制約条件）

- $C$について： $C$が大きいと完全に分離することに過度になり、汎化性能が小さくなるので、一般に過学習を防ぐためには$C$を小さくする方がよい。

## ▣ サポートベクターマシンのハンズオン

- サマリー・考察は、np_svm.ipynb に記載

# 機械学習・要点まとめ

http://study-ai.com/jdla/