

機械学習

秋葉洋哉

2024 年 6 月 19 日

1 導入

1.1 機械学習とは

トム・ミッチェル (1997) によれば、コンピュータプログラムにおいて、タスク T を性能指標 P で測定し、経験 E から学習させ、このとき性能指標 P が最大となるように学習を繰り返すことが機械学習である。

1.2 機械学習のモデリングプロセス

機械学習のモデリングプロセスは以下のようになる。

1. 問題設定
2. データ選定
3. データ前処理
4. 機械学習・モデル選択
5. モデル学習
6. モデル評価・改良

ビジネスにおいて、最も大切なプロセスが問題設定の部分であり、機械学習を用いるかどうか、という点をまず第一に考える必要がある。

1.3 機械学習の分類

機械学習は、学習種類・タスク・アルゴリズム (パラメータの推定問題) の 3 つの観点から分類される。

1. 学習種類
 - 教師あり学習
 - 教師なし学習
2. タスク
 - 予測 (回帰)
 - 分類
 - クラスタリング
 - 次元削減

3. アルゴリズム

- 線形・非線形回帰 (最小二乗法・尤度最大化)
- ロジスティック回帰 (尤度最大化)
- サポートベクターマシン (マージン最大化)
- K-means
- 主成分分析 (分散最大化)

2 線形回帰モデル

2.1 概要

線形とは、比例の関係である状態のことであり、線形回帰モデルは、入力変数と出力変数の関係が線形であると仮定した時の回帰モデルである。n 次元の線形回帰モデルは、 a_k, x_k を用いて以下のように表される。

$$y = a_0 + a_1x_1 + a_2x_2 + \cdots + a_{n-1}x_{n-1} \quad (1)$$

$$= a_0 + \sum_{k=1}^{n-1} a_k x_k \quad (2)$$

$$= \sum_{k=1}^{n-1} a_k x_k, \text{ where } x_0 = 1 \quad (3)$$

$$= \mathbf{a}^T \mathbf{x} \quad (4)$$

ただし、 $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})^T$, $\mathbf{x} = (1, x_1, x_2, \dots, x_{n-1})^T$ である。

回帰問題とは、ある入力から連続値の出力を予測する問題のことで、直線で予測されるなら線形回帰モデル、曲線で予測されるなら非線形回帰モデルが用いられる。

本来解きたい問題よりも、難しい問題にして解決しようとせずに、解きたい問題よりも簡単な問題にして解決するほうがよい、というエンジニアリング上の教訓がある。これは、Vapnic の原理と呼ばれている。回帰分析を用いることで、例えば何かの順位を予測することは可能である。ただし、解きたい問題が順位を決める、という問題である場合、回析分析を用いることは適切ではない。なぜなら、回帰分析では、順位だけではなく、順位の間隔も予測することになるため、解きたい問題よりも難しい問題になってしまうからである。つまり、回帰分析を用いる場合は、解きたい問題が何かを明確にし、連続値として予測される問題であれば使用することが適切である。

回帰（さしずめ機械学習）においては、入力値を説明変数、出力値を目的変数と呼ぶ。説明変数は特徴量とも呼ばれる。回帰分析では、説明変数と目的変数の関係を表すモデルを構築し、そのモデルを用いて目的変数を予測する。回帰分析は、教師あり学習の一つで、入力と m 次元パラメータの線形結合を出力するモデルである。式で表すと、

$$y = \mathbf{w}^T \mathbf{x} + w_0 + \epsilon_i \quad (5)$$

$$= \sum_{k=1}^m w_k x_k + w_0 + \epsilon_i \quad (6)$$

ただし、 ϵ_i は誤差項である。で表すことができる。これを図で表すと、図 1 のようになる。

数式のそれぞれを説明した図が、図 2 である。図 2 における誤差は、 x_1 以外の要素があるとき、本来は x_2, x_3 を用いて多次元にすべき部分が定数として収められた部分になる。

また、誤差項 ϵ_i は、平均 0、分散 σ^2 の正規分布に従うと仮定して、モデルの不完全的要素を表す。この仮定に基づいて、最尤推定法を用いてパラメータを推定する。

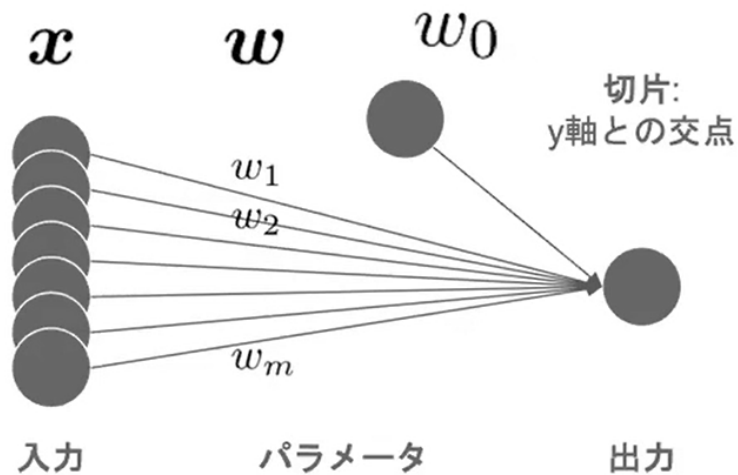


図1 線形回帰モデルの概念図

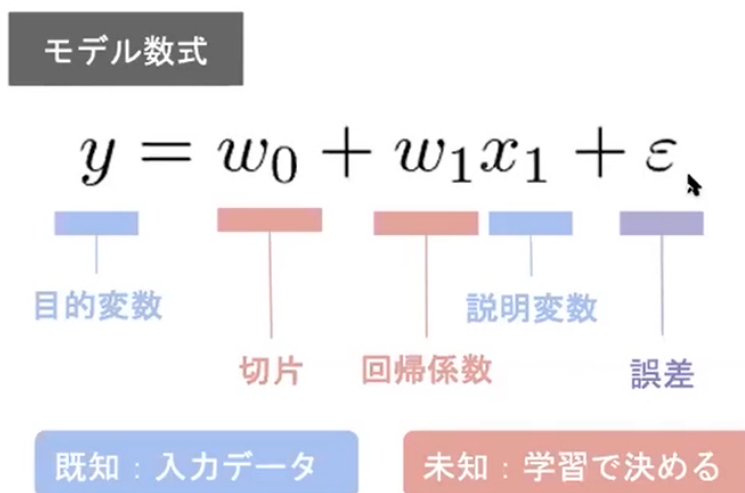


図2 線形回帰モデル式のそれぞれの意味

2.2 汎化

機械学習において、学習データに対して過度に適合してしまうことを過学習と呼び、未知のデータに対しても適切に予測できることを汎化と呼ぶ。過学習を防ぐためには、学習データを訓練データとテストデータに分割し、訓練データで学習を行い、テストデータで評価を行うことが一般的である。

2.3 最小二乗法

最小二乗法は、誤差の二乗和を最小にするようなパラメータを求める方法である。学習データを用いて、誤差の二乗和を最小にするようなパラメータを求めることで、モデルのパラメータを最適化することができる。式で表すと、

$$\text{MSE}_{\text{train}}(\mathbf{w}) = \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} (\hat{y}_i - y_i)^2 \quad (7)$$

$$= \frac{1}{n_{\text{train}}} \sum_{i=1}^{n_{\text{train}}} (\mathbf{x}_i^T \mathbf{w} - y_i)^2 \quad (8)$$

で表すことができる。ただし、 \hat{y}_i は予測値、 y_i は実測値、 n_{train} は学習データの数である。この式を最小にするような m 次元の \mathbf{w} の組み合わせを求めることが、線形回帰モデルを作成する上での目標である。最小二乗法は、一般的に外れ値の影響を受けやすいという欠点があるため、外れ値を多く含むデータに対しては、前処理を工夫したり、Huber 損失、Tukey 損失 (外れ値無視) などのロバスト回帰を用いることがある。

MSE を最小にするような m 次元の w を求める、という部分は数式では

$$\hat{w} = \arg \min_{w \in \mathbb{R}^{m+1}} \text{MSE}_{\text{train}}(w) \quad (9)$$

と表すことができる。この \hat{w} を求めることが最小二乗法の目標である。そのためには、MSE を w で微分し、その微分が 0 になるような w を求めることが必要である。つまり、

$$\frac{\partial}{\partial w} \text{MSE}_{\text{train}}(w) = 0 \quad (10)$$

を解くことで、最小二乗法によるパラメータの最適化を行うことができる。

```

import pandas as pd
from sklearn.datasets import fetch_california_housing
import numpy as np
housing = fetch_california_housing()
print(housing["target"])
df = pd.DataFrame(data=housing.data, columns=housing.feature_names)
df.head(5)

```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25

Next steps: [Generate code with df](#) [View recommended plots](#)

```

data = df.loc[:, ["AveRooms"]].values
data[0:5]
target_df = pd.DataFrame(housing.target, columns=["Price"])
target_df.head()

```

	Price
0	4.526
1	3.585
2	3.521
3	3.413
4	3.422

```

[32] target = target_df.loc[:, ["Price"]].values
print(target)

```

```

[33] from sklearn.linear_model import LinearRegression

```

```

[34] model = LinearRegression()
model.get_params()

```

```

[36] model.fit(data, target)

```

```

model.predict([[1]])

```

```

[42] df[["HouseAge", "AveRooms"]].head()
data2 = df.loc[:, ["HouseAge", "AveRooms"]].values
data2[0:5]

```

```

[44] model2 = LinearRegression()
model2.get_params()

```

```

[45] model2.fit(data2, target)

```

```

[54] model2.predict([[20, 5]])

```

```

[49] print(model2.coef_)
print(model2.intercept_)

```

図3 実装演習キャプチャ (カリフォルニアデータセットで代用)

3 非線形回帰モデル

3.1 概要

非線形回帰とは、例えば、以下のような関数を用いて回帰を行う。

$$y = w_0 + w_1x + w_2x^2 + w_3x - 3 \quad (11)$$

$$y = w_0 + w_1 \sin(x) + w_2 \cos(x) + w_3 \log(x) \quad (12)$$

非線形回帰モデルは、線形回帰における x という変数部分を $\phi(x)$ で置き換えたものと考えることができる。従って、線形回帰における重み \mathbf{w} の導出を、非線形回帰においても適用することができる。

$$\hat{y} = w_0 + w_1\phi(x_1) + w_2\phi(x_2) + \cdots + w_m\phi(x_m) \quad (13)$$

このようにして展開する方法を、基底展開法と呼ぶ。このことは、重み \mathbf{w} について線形であるということができる。ただし、 $\phi(x)$ は基底関数と呼ばれる関数で、多項式関数、ガウス基底関数、シグモイド基底関数などがある。多項式関数は、

$$\phi(x) = x^j \quad (14)$$

で表される。ガウス基底関数は、

$$\phi(x) = \exp\left(-\frac{(x - \mu_j)^2}{2\sigma^2}\right) \quad (15)$$

で表される。ガウス基底関数は、2つの山を考え、それぞれの山の中心を μ_1, μ_2 とし、それぞれの山の広がりを σ_1, σ_2 とすることで、ある入力を与えられたとき、それがどちらの山の影響を強く受けるかを表すことができる。シグモイド基底関数は、

$$\phi(x) = \frac{1}{1 + \exp(-\beta(x - \mu))} \quad (16)$$

で表される。

線形回帰における行列表記を、非線形回帰に適応させると、

$$\mathbf{y} = \mathbf{X}\mathbf{w} \quad (17)$$

が、

$$\mathbf{y} = \Phi\mathbf{w} \quad (18)$$

となる。ただし、 Φ は、

$$\Phi = (\phi(x_1) \quad \phi(x_2) \quad \cdots \quad \phi(x_m)) \quad (19)$$

である。結局、MSE を最小化する \mathbf{w} は、 \mathbf{X} を Φ に置き換えるだけで、

$$\hat{\mathbf{w}} = \Phi^T(\Phi\Phi^T)^{-1}\mathbf{y} \quad (20)$$

となる。

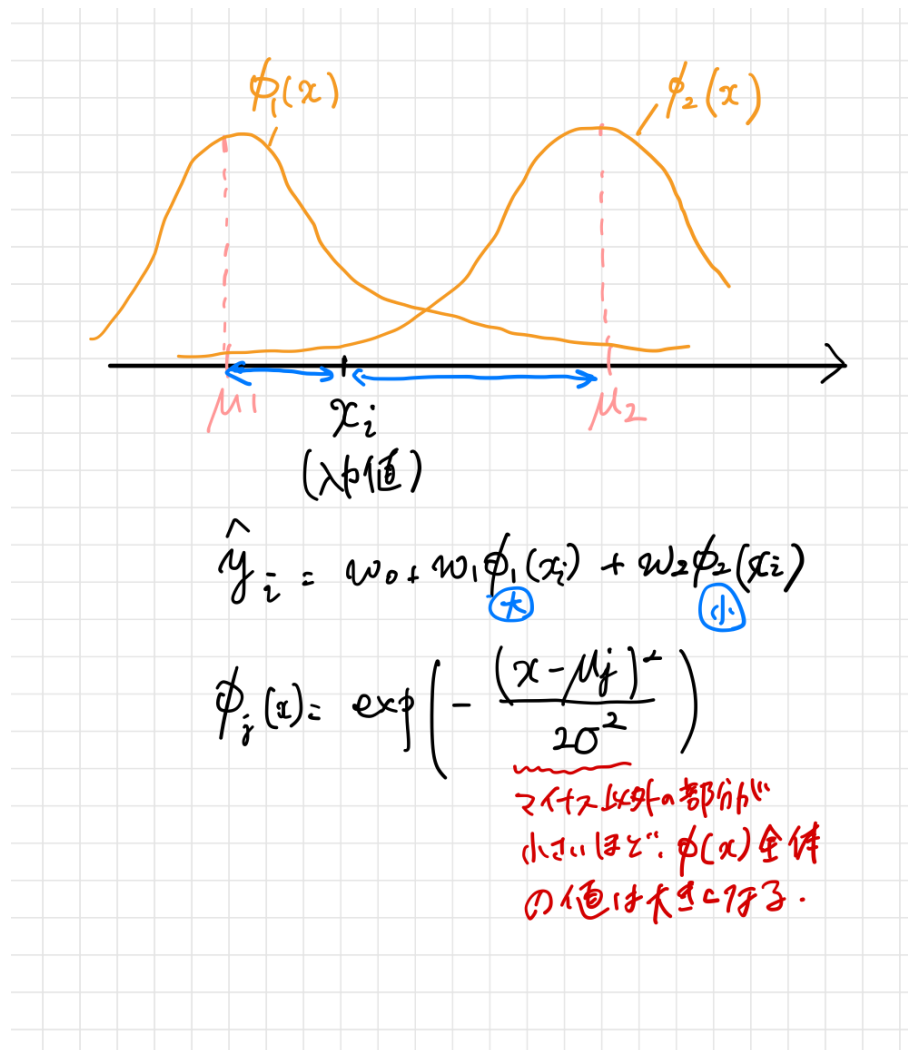


図4 ガウス基底関数のイメージ

3.2 未学習と過学習

未学習とは、学習データに対しても適合が悪い状態のことであり、過学習とは、学習データに対して過度に適合してしまう状態のことである。過学習を防止する策として、以下が挙げられる。

1. 学習データを増やす
2. 使用する説明変数を減らす
3. 正則化を行う

正則化については、以下のように考える。線形回帰における、予測 \hat{y} は、

$$\hat{y} = X_* \cdot (X^T X)^{-1} X^T y \quad (21)$$

であった。ただし、 $(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}$ は、 \mathbf{w} である。過学習を防ぐということは、あまりにも \mathbf{w} が大きくなりすぎないようにすることである。つまり、 $(\mathbf{X}^T \mathbf{X})^{-1}$ が大きくなりすぎないようにすることが必要である。例えば、ある入力値 \mathbf{X} が

$$\mathbf{X} = \begin{pmatrix} 1 & 2 & 4 \\ 1 & 3 & 5.9 \\ 1 & 4 & 8.1 \end{pmatrix} \quad (22)$$

であるとする。この 3 番目のベクトルが、2 番目のベクトルとほぼ同じであると考えられる。これは、2 本のベクトルが平行に近いことを意味している。このような場合、転置行列を前から掛けた時の行列は、

$$\mathbf{X}^T \mathbf{X} = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 3 & 4 \\ 4 & 5.9 & 8.1 \end{pmatrix} \begin{pmatrix} 1 & 2 & 4 \\ 1 & 3 & 5.9 \\ 1 & 4 & 8.1 \end{pmatrix} = \begin{pmatrix} 3 & 9 & 18 \\ 9 & 29 & 58.1 \\ 18 & 58.1 & 116.42 \end{pmatrix} \quad (23)$$

であるから、逆行列は、

$$(\mathbf{X}^T \mathbf{X})^{-1} = \begin{pmatrix} 3 & 9 & 18 \\ 9 & 29 & 58.1 \\ 18 & 58.1 & 116.42 \end{pmatrix}^{-1} \approx \begin{pmatrix} 6.33 & -22 & 10 \\ -22 & 281 & -137 \\ 10 & -137 & 66.7 \end{pmatrix} \quad (24)$$

となる。本来であれば、大きくても 2 桁程度の値に収まるべきであるが、この逆行列は、数値が大きくなりすぎている。よって、 \mathbf{w} が大きな値を取ることとなり、過学習を引き起こすこととなる。これを防ぐために、MSE 最小化に対して、罰則化項を導入することで、 \mathbf{w} が大きな値を取らないようにすることができる。

$$E(\mathbf{w}) = \text{MSE}_{\text{train}}(\mathbf{w}) + \lambda \mathbf{w}^T \mathbf{w} \quad (25)$$

ただし、 $E(\mathbf{w})$ は、正則化項を導入した誤差関数で、 λ は正則化項の重みである。この 2 項目で、 \mathbf{w} が大きな値を取った時に、誤差が大きくなるようになり、MSE を最小化するという目的に対して、 \mathbf{w} が大きな値を取らないようにするという仕組みである。

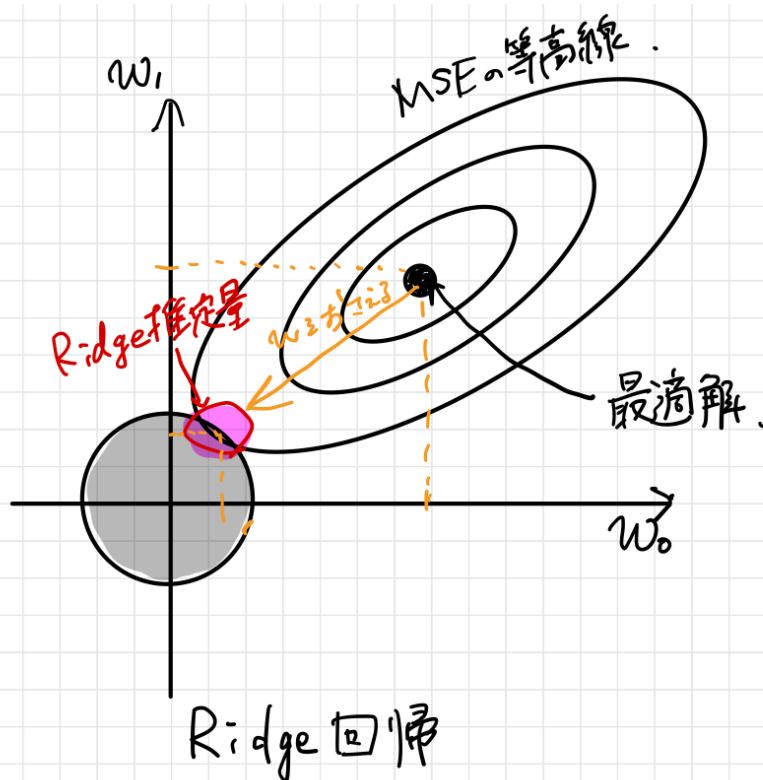
このような最適化の流れを一般化すると、以下で表される。解きたいものは、 $R(\mathbf{w}) \leq r$ という条件を満たしつつ、 $\min \text{MSE}$ を求めることである。そこで、KKT 条件を用いて、以下のように最適化問題を解くことができる。

$$\min \text{MSE} + \lambda R(\mathbf{w}) \quad (26)$$

$$(27)$$

この第 2 項目部分によって不等式条件を満たすことができる。

回帰分析における Ridge 回帰と、Lasso 回帰は、どちらも正則化項を導入することで、過学習を防ぐ方法であるが、Ridge 回帰は、L2 正則化を行い、Lasso 回帰は、L1 正則化を行うという説明がなされることが多い。そのうち、L2 正則化というものが $\lambda \mathbf{w}^T \mathbf{w}$ を使用する方法になる。



- ・ 正則化項が無い時は、 (w_0, w_1) の組み合わせは、 w_0 と w_1 の座標の中で存在しているが、正則化項が付与された時には、図のグレーの円の中で最も最適化されるのはどこか? という問題になる。

図5 Ridge 回帰のイメージ

3.3 学習と検証

学習と検証は、モデルの汎化性能を評価するための方法である。

ホールドアウト法は、データを訓練データとテストデータに分割し、訓練データで学習を行い、テストデータで評価を行う方法である。一括でデータを分割するため、データの偏りが生じる可能性がある。例えば、検証データ内に外れ値が含まれてしまうと、モデルの評価が不正確になる可能性がある。

クロスバリデーション (交差検証) は、データを k 個に分割し、 $k-1$ 個のデータで学習を行い、残りの 1 個のデータで評価を行う方法である。この操作を k 回繰り返し、 k 個の評価結果の平均を取ることで、モデルの評価を行うことができる。交差検証は、ホールドアウト法に比べて、データの偏りが生じにくいという利点がある。

グリッドサーチは、ハイパーパラメータの最適な値を探索するための方法である。ハイパーパラメータとは、モデルの学習時に設定するパラメータのことで、例えば、正則化項の重み λ や、基底関数の次数などがある。

4 ロジスティクス回帰

4.1 概要

以降では、分類における問題を考える。分類においては、識別的アプローチと生成的アプローチの2つのアプローチがある。

- 識別的アプローチ: ある入力に対して、その入力がどのクラスに属するかを直接予測する方法。例えば、ロジスティック回帰、SVM、決定木などがある。 $P(C_k)$ が求めたいクラスの確率とすると、 $P(C_k|\mathbf{x})$ を直接モデル化して求める。
- 生成的アプローチ: ある入力を与えられた時に、その入力が生成される確率を求め、その確率が最大となるクラスを予測する方法。例えば、ナイーブベイズ、GMM、HMM などがある。つまり、 $P(C_k)$, $P(\mathbf{x}|C_k)$ をモデル化し、その後、ベイズの定理を用いて $P(C_k|\mathbf{x})$ を求める。

識別的アプローチにおいては、分類される確率の最も高いクラス \hat{y} を求めることが目的であり、

$$\hat{y} = \arg \max_k P(C_k|\mathbf{x}) \quad (28)$$

を解くことで求める。一方で、生成的アプローチにおいては、 $P(\mathbf{x})$ の確率は一定であるため、

$$\hat{y} = \arg \max_k P(C_k)P(\mathbf{x}|C_k) \quad (29)$$

を解くことで求める。ここで、 $P(C_k)$ は各クラスの割合である (例えば、クラス 1 の確率が 60% なら $P(C_1) = 0.6$)。 $P(\mathbf{x}|C_k)$ は、「各クラスが観測データ x を生成する確率 (尤度) がどのくらいであるか」を表したものである。このようにすることで、あるクラスに属する疑似的なデータを生成することができるという利点がある。

ロジスティック回帰は、識別的アプローチに属する手法で、入力を与えられた時に、その入力がどのクラスに属するかを直接予測する方法である。

回帰分析では、実数全体から実数全体への関数を求めるため、

$$\mathbf{w}^T \mathbf{x} \in \mathbb{R} \quad (30)$$

であったが、ロジスティック回帰では、実数全体から 0, 1 の 2 値への関数を求めるため、

$$\sigma(\mathbf{w}^T \mathbf{x}) \in [0, 1] \quad (31)$$

を考える。ただし、 σ はシグモイド関数で、

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \quad (32)$$

で表される。このシグモイド関数は、 z が大きいときに 1 に近づき、 z が小さいときに 0 に近づく性質を持つ。この性質を用いて、入力を与えられた時に、その入力がどのクラスに属するかを予測することができる。

シグモイド関数の微分

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

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

$$= -(1 + \exp(-ax))^{-2} \cdot (1 + \exp(-ax))'$$

$$= -(1 + \exp(-ax))^{-2} (-a \exp(-ax))$$

$$= - \frac{-a \cdot \exp(-ax)}{(1 + \exp(-ax))^2}$$

$$= \frac{a}{1 + \exp(-ax)} \cdot \frac{(1 + \exp(-ax)) - 1}{(1 + \exp(-ax))^2}$$

$$= a \cdot \sigma(x) \cdot (1 - \sigma(x))$$

図6 シグモイド関数の微分

4.2 最尤推定

ある分布を仮定した時に、その分布が観測データを生成する確率が最大となるようなパラメータを求める方法を最尤推定と呼ぶ。データは固定されたものとして、パラメータを変化させることで、尤度を最大化するよ

うなパラメータを求めるアプローチである。ロジスティック回帰においては、尤度関数は、

$$L(\mathbf{w}) = \prod_{i=1}^n P(C_k | \mathbf{x}_i)^{y_i} (1 - P(C_k | \mathbf{x}_i))^{1-y_i} \quad (33)$$

$$= \prod_{i=1}^n \sigma(\mathbf{w}^T \mathbf{x}_i)^{y_i} (1 - \sigma(\mathbf{w}^T \mathbf{x}_i))^{1-y_i} \quad (34)$$

で表される。ただし、 y_i は、 i 番目のデータがクラス C_k に属するかどうかを表すラベルである。尤度関数では、確率変数が独立かつ同一の分布に従っていることを仮定しているため、各データの確率の積として表されている。この尤度関数を最大化するようなパラメータ \mathbf{w} を求めることが、ロジスティック回帰における最尤推定の目標である。ただし、尤度関数をそのまま最大化することは難しいため、尤度関数の対数を取り、対数尤度関数を最大化することが一般的である。対数尤度関数は、

$$\log L(\mathbf{w}) = \sum_{i=1}^n \{y_i \log \sigma(\mathbf{w}^T \mathbf{x}_i) + (1 - y_i) \log(1 - \sigma(\mathbf{w}^T \mathbf{x}_i))\} \quad (35)$$

で表される。

なぜ対数尤度関数を用いるのか？

尤度関数は確率の掛け算で表されている。1 よりも小さい値の確率の掛け算を繰り返すことで、とても小さな値となって計算機でオーバーフローが発生する可能性があるため、対数を用いて積を和に、指数を積に変換することで、計算を安定化させることができる。

対数尤度関数の微分.

$$\begin{aligned}
 E(\vec{w}) &= -\ln L(\vec{w}) \\
 &= -\sum_{i=1}^n \left\{ \underbrace{y_i \ln p_i}_{\text{red dashed}} + \underbrace{(1-y_i) \ln(1-p_i)}_{\text{blue dashed}} \right\} \\
 \frac{\partial E(\vec{w})}{\partial \vec{w}} &= \sum_{i=1}^n \frac{\partial E_i(\vec{w})}{\partial p_i} \frac{\partial p_i}{\partial \vec{w}} = \sigma(\vec{w} \vec{x}_i) \left(\frac{\partial \sigma}{\partial \vec{w}} = \sigma \cdot (1-\sigma) \right)_{\text{E使用}} \\
 &= -\sum_{i=1}^n \left(\underbrace{\frac{y_i}{p_i}}_{\text{red dashed}} - \underbrace{\frac{(1-y_i)}{(1-p_i)}}_{\text{blue dashed}} \right) \underbrace{p_i(1-p_i)}_{\text{orange dashed}} \vec{x}_i \\
 &= -\sum_{i=1}^n \left(\frac{y_i(1-p_i) - p_i(1-y_i)}{\cancel{p_i(1-p_i)}} \right) \cdot \cancel{p_i(1-p_i)} \vec{x}_i \\
 &= -\sum_{i=1}^n \{ y_i(1-p_i) - p_i(1-y_i) \} \vec{x}_i \\
 &= -\sum_{i=1}^n (y_i - p_i) \vec{x}_i
 \end{aligned}$$

図7 対数尤度関数の微分

4.3 勾配降下法

反復学習によりパラメータを逐次的に更新するアプローチの一つである。勾配降下法は、目的関数の勾配を用いて、目的関数を最小化するようなパラメータを求める方法である。

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} - \eta \frac{\partial E(\mathbf{w})}{\partial \mathbf{w}} \quad (36)$$

$$= \mathbf{w}^{(t)} + \eta \sum_{i=1}^n (\sigma(y_i - p_i) \mathbf{x}_i) \quad (37)$$

で表される。ただし、 η は学習率であり、パラメータの更新量を調整するためのハイパーパラメータである。

37 式は、 n 個のデータを用いてパラメータを更新するため、コンピュータのメモリに対する負荷が大きくなる可能性がある。そのため、データを 1 つずつ用いてパラメータを更新する方法を確率的勾配降下法 (SGD) と呼ぶ。確率的勾配降下法は、データを 1 つずつ用いてパラメータを更新するため、計算コストが低いという利点がある。ただし、収束が遅くなるという欠点がある。

5 主成分分析

5.1 概要

主成分分析は、多次元データを低次元データに変換する手法である。主成分分析は、データの分散が最大となるような軸を求めることで、データの情報を最大限に保持するような低次元データを求めることができる。

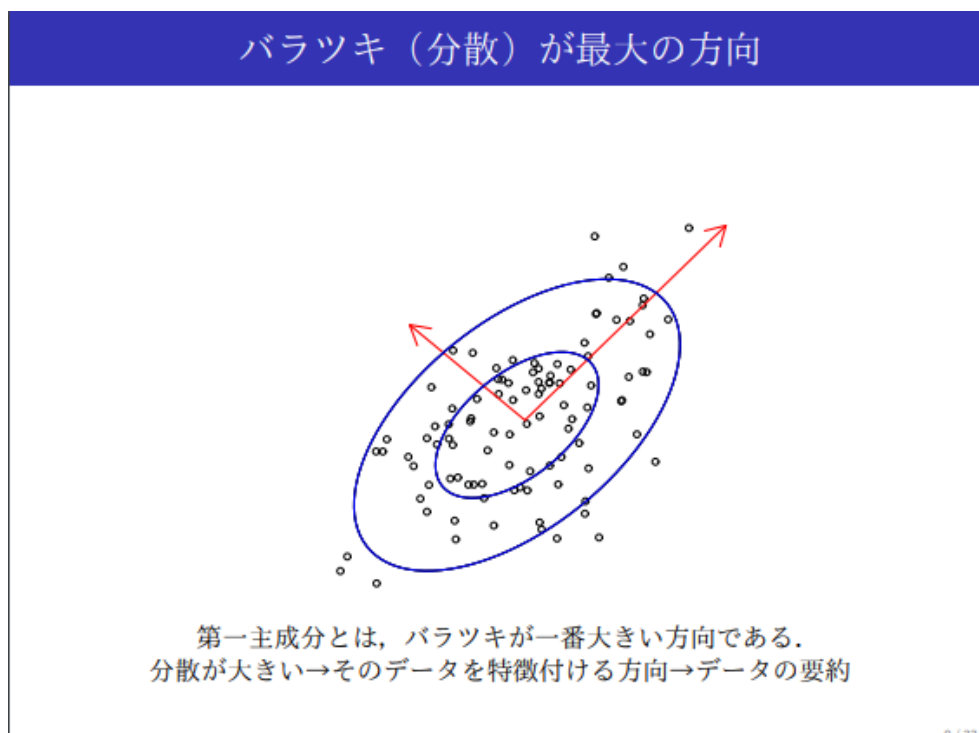


図8 主成分分析のイメージ (<https://ibis.t.u-tokyo.ac.jp/suzuki/lecture/2015/dataanalysis/L7.pdf> より引用)

ある方向ベクトルを $\mathbf{v}(\|\mathbf{v}\| = 1)$ とおくと、この方向への \mathbf{x} の長さは、

$$\mathbf{x}^T \mathbf{v} \quad (38)$$

で求まる。この分散は、

$$\text{Var}(\mathbf{x}^T \mathbf{v}) = \frac{1}{n-1} \sum_{i=1}^n [\mathbf{v}^T (\mathbf{x}_i - \hat{\boldsymbol{\mu}})]^2 \quad (39)$$

$$= \mathbf{v}^T \left(\frac{1}{n-1} \sum_{i=1}^n (\mathbf{x}_i - \hat{\boldsymbol{\mu}})(\mathbf{x}_i - \hat{\boldsymbol{\mu}})^T \right) \mathbf{v} \quad (40)$$

$$=: \mathbf{v}^T \boldsymbol{\Sigma} \mathbf{v} \quad (41)$$

で表される。この時、 $\boldsymbol{\Sigma}$ は、データの共分散行列である。この分散を最大にするような \mathbf{v} を求めることが、主成分分析の目的である。

6 k 近傍法

6.1 概要

k 近傍法は、分類問題や回帰問題において、新しいデータが与えられた時に、そのデータに最も近い k 個のデータを用いて、そのデータがどのクラスに属するかを予測する手法である。k 近傍法には、kd-tree、近似最近傍探索といった手法が存在する。

k が小さいとき、より局所的な情報を用いるため、識別面が入り組んだ形状となる。一方で、k が大きいとき、より大域的な情報を用いるため、識別面が滑らかな形状となるが、k をあまりにも大きくすると、識別精度が下がったり、距離計算のコストが大きくなるという欠点がある。

また、高次元の空間では、データのフィルタリングと精度のバランスをとることが重要である。その際に近似最近傍探索を用いることで、計算コストを削減することができる。その一つが kd-tree である。

kd-tree は、データを分割することで、データの探索を効率化する手法である。kd-tree は、データを分割する際に、データの分散が最大となるような軸や、データの中央値を選択し、データを分割する。この操作を繰り返すことで、データを高速・効率的に探索することができる。ただし、高次元の計算には適さず、あくまで多次元のデータに対して適している。

6.2 距離計算

k 近傍法において、データ間の距離を計算する方法には、コサイン距離、ユークリッド距離、マンハッタン距離、Lp 距離、マハラノビス距離などがある。

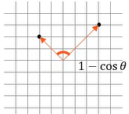

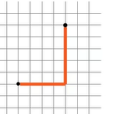
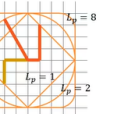
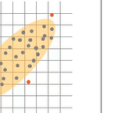
	コサイン距離	ユークリッド距離	マンハッタン距離	Lp距離	マハラノビス距離
模式図					
数式	$d(x, y) = 1 - \cos \theta$ $= 1 - \frac{x \cdot y}{\ x\ \ y\ }$	$d(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$	$d(x, y) = \sum_{i=1}^n x_i - y_i $	$d(x, y) = \sqrt[p]{\sum_{i=1}^n (x_i - y_i)^p}$	$d(x, y) = \sqrt{(x - y)^T S^{-1} (x - y)}$ S ⁻¹ : 共分散行列
説明	<ul style="list-style-type: none"> 2つのベクトルがなす角のコサイン値 	<ul style="list-style-type: none"> いわゆる“通常の”距離 L₂距離とも呼ぶ 	<ul style="list-style-type: none"> 各座標における差の絶対値の総和 L₁距離とも呼ぶ 	<ul style="list-style-type: none"> L_p空間におけるL_p距離を一般化したもの 図は中心点からの距離が同じ箇所を示したもの 	<ul style="list-style-type: none"> 統計学で用いられる データの散らばりを考慮し、データ群からの距離を示す
事例	<ul style="list-style-type: none"> テキストマイニング 自然言語処理 	<ul style="list-style-type: none"> クラスタリング 画像処理 特徴量の比較 	<ul style="list-style-type: none"> 特徴量の比較 ルート探索 	<ul style="list-style-type: none"> k近傍法の一般化 特徴量の正規化 次元削減 	<ul style="list-style-type: none"> 異常検知 特徴量の選択

図 9 距離計算の手法

7 k-means 法

7.1 概要

k-means 法は、教師なし学習のクラスタリング手法の一つで、データを k 個のクラスタに分割する手法である。k-means 法は、各クラスタの中心を求め、各データが最も近いクラスタに割り当てることで、データをクラスタに分割する。初期値が近いと上手くクラスタリングできないことがあるため、複数回の初期値設定を行い、最も良いクラスタリング結果を選択する方法がある。

7.2 k-means++

k-means 法は初期値に応じて、結果が大きく変わることがある。そのため、初期値を適切に設定することが重要な要素となる。k-means++ は、k-means 法における初期値依存問題の克服を目指したアルゴリズムである。k-means++ は初期のクラスタの中心同士が離れていた方がよい、という考え方に基づいて設計されている。クラスタの割り振り方は、以下の通りである。

1. 各データ点 x_i の中からランダムに 1 点を選び、クラスタ中心とする
2. 各点 x_i に関して、既存のクラスタ中心の中から最も近いクラスタ中心との距離 $D(x)$ を計算する
3. 各点 x_i に関して、重み付き確率分布 $P(x_i) = \frac{D(x_i)^2}{\sum_j D(x_j)^2}$ に従って、新たなクラスタ中心を選ぶ。ただし、 j は既存のクラスタ中心のインデックスである。
4. 2. と 3. の工程をクラスタ中心が選定できるまで行う

ポイントは、重み付き確率分布 $\frac{D(x_i)^2}{\sum_j D(x_j)^2}$ の確率が最も高くなる場所は、既存のクラスタ中心から最も遠い点になるということである。以下に、k-means++ の初期化のアルゴリズムを示す。

```
probabilities = np.repeat(1/n, n)
centroids = np.zeros((k, 2))
distances = np.zeros((n, k))

for i in range(k):
    centroids[i] = X[np.random.choice(np.arange(n), p=probabilities, size=(1))]
    distances[:, i] = np.sum((data - centroids[i])**2, axis=1)
    # ある点が次のクラスタ中心になりうる確率 = (ある点の各クラスタからの距離の合計)/(すべての点の各クラスタからの距離の合計)
    probabilities = np.sum(distances, axis=1)/np.sum(distances)
```

8 SVM(サポートベクターマシン)

8.1 概要

SVM は、分類問題において、データを 2 つのクラスに分類する手法である。SVM は、データを分割する直線を決定するため、最適な直線を求めることが目的である。SVM は、マージン最大化という考え方に基づいており、データを分割する直線と、その直線に最も近いデータとの距離が最大となるような直線を求めることが目的である。SVM は、線形 SVM と非線形 SVM に分けられる。線形 SVM は、データを分割する直線が直線である場合であり、非線形 SVM は、データを分割する直線が曲線である場合である。

8.2 カーネル法

非線形 SVM において、トレーニングデータを射影関数 ϕ を使って高次元の特徴空間に射影して学習し、学習後に同じ射影関数 ϕ を使って元に戻して解く方法をカーネル法と呼ぶ。

$$f(x) = b + \sum_{i=1}^m \alpha_i x^T x^{(i)} \quad (42)$$

$$= b + \sum_{i=1}^m \alpha_i \cdot \phi(x) \cdot \phi(x^{(i)}) \quad (43)$$

ただし、 α_i は、ラグランジュ乗数である。この方法は、射影する特徴空間の次元数が非常に高くなってしまいうことが多く、新しい特徴量 $\phi(x), \phi(x^{(i)})$ を生成する計算コストが非常に高くなってしまいう。そこで、射影関数の内積結果 $\phi(x) \cdot \phi(x^{(i)})$ を既知の変数だけで計算する手法をカーネルトリックと言い、その条件を満たす関数 $f(x, x^{(i)}) = \phi(x) \cdot \phi(x^{(i)})$ をカーネル関数と呼ぶ。

代表的なカーネル関数は、以下の通りである。

- 多項式カーネル: $K(x^{(i)}, x^{(j)}) = (x^{(i)T} x^{(j)} + c)^d$
- ガウシアンカーネル: $K(x^{(i)}, x^{(j)}) = \exp(-\frac{\|x^{(i)} - x^{(j)}\|^2}{2\sigma^2})$
- シグモイドカーネル: $K(x^{(i)}, x^{(j)}) = \tanh(\alpha x^{(i)T} x^{(j)} + c)$

■参考文献

1. 機械学習・自然言語処理の勉強メモ- 識別モデルと生成モデル (<https://kento1109.hatenablog.com/entry/2018/01/10/111249>) 2024 年 4 月 21 日閲覧
2. k-means++ を理解する (@gk) (<https://qiita.com/g-k/items/e1d558ffcdc833e6382c>) 2024 年 6 月 11 日閲覧
3. 機械学習/カーネル法 (@jun40vn) (<https://qiita.com/jun40vn/items/07c3b4d242f3e7e66265>) 2024 年 6 月 19 日閲覧