実装演習レポート【機械学習】 線形回帰モデル

要点のまとめ

- 線形回帰モデルとは

  ① 回帰問題を解くための機械学習モデルの一つ
  ② 教師あり学習
  ③ 入力とm次元パラメータの線形結合を出力するモデル

- 線形結合とは入力ベクトルと未知のパラメータの各要素を掛け算し足し合わせたもの。
  線形回帰モデルではこの線形結合に加え、切片（y軸との交点）も足し合わせる。
  パラメータ（回帰係数や切片）は最小二乗法により推定される。

- 線形回帰モデルを式で表すと下記のようになる。

目的変数、説明変数は**既知**（入力データ）
切片、回帰係数は**未知**（学習で決める）

単回帰モデル（説明変数が１つ）

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

目的変数　切片　回帰係数　説明変数　誤差

重回帰モデル（説明変数が２つ以上）

$$y = w_0 + w_1 x_1 + w_2 x_2 + \varepsilon$$

目的変数　切片　回帰係数　説明変数　回帰係数　説明変数　誤差

実装演習レポート【機械学習】 線形回帰モデル

実装演習結果

- 設定
  - ボストンの住宅データセットを線形回帰モデルで分析
  - 適切な査定結果が必要
    - 高すぎても安すぎても会社に損害がある
- 課題
  - 部屋数が4で犯罪率が0.3の物件はいくらになるか？

⇒ 4240ドル

【サマリーと考察】

・ 必要なライブラリとデータをロードし、データ処理を実行。
今回は説明変数が2つ（部屋数、犯罪率）があるので、重回帰分析を実施。
パラメータ推定したものから部屋数4、犯罪率が0.3のときの価格を4240ドルと予測。

回帰係数は部屋数が8.39、犯罪率が-0.26。
部屋数が多くなると、価格は上昇。部屋数が1つ増えると、価格が8390ドル上昇。
犯罪率が多くなると、価格は下落。犯罪率が1.00増えると、価格が260ドル減少。

実装演習レポート【機械学習】線形回帰モデル

実装演習結果（コード）

ライブラリをロード

```
[1] # ライブラリをロード
    import pandas as pd
    import numpy as np
    from sklearn.linear_model import LinearRegression
    from sklearn.datasets import load_boston
```

データをロード

```
[2] # データをロード
    boston = load_boston()
```

データフレームに変換

```
[3] df = pd.DataFrame(data=boston.data , columns = boston.feature_names)
```

```
[4] df["PRICE"] = np.array(boston.target)
```

重回帰分析（2変数）

```
[5] # 説明変数
    data = df.loc[:,["CRIM","RM"]].values
    # 目的変数
    target = df.loc[:,["PRICE"]].values
```

```
[6] # オブジェクトを作成
    model = LinearRegression()
```

```
[7] # fit関数でパラメータ推定
    model.fit(data,target)
```

```
    LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
[8] # 予測（犯罪率 0.3 , 部屋数 7）
    model.predict([[0.3,4]])
```

```
    array([[4.24007956]])
```

回帰係数と切片の値を確認

```
[9] # 回帰係数
    model.coef_
```

```
    array([[-0.26491325,  8.39106825]])
```

```
[10] # 切片
     model.intercept_
```

```
     array([-29.24471945])
```

実装演習レポート【機械学習】 線形回帰モデル

参考図書レポート

- 回帰モデルの評価について

  回帰モデルを評価するのによく使用されるのが平均二乗誤差（MSE）と決定係数（R^2）。

  - MSEは、予測値と真の値の距離を2乗したものの平均値。MSEが大きいということは、誤差を2乗したもの総計が大きいということなので、良くないモデルということになる。

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$$

$\hat{y}_i$：予測値　y_i：真の値

  - 決定係数とはモデルによって説明されるターゲットベクトルの分散の量を表す。
    1.0に近いほど、良いモデルということになる。

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2}$$

$\hat{y}_i$：予測値　y_i：真の値

$\bar{y}_i$：ターゲットベクトルの平均値

実装演習レポート【機械学習】 非線形回帰モデル

要点のまとめ

- データ構造を線形で捉えられる場合は限られる。非線形な構造を捉えられる仕組みが必要。
  非線形回帰モデルでは、基底展開法を用いる。

- 基底展開法とは

  ① 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形結合を使用
  ② 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

- 未学習と過学習

  未学習：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
  【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。
  過学習：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
  【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

対策2、対策3は
モデルの煩雑さを
調整する方法

実装演習レポート【機械学習】 非線形回帰モデル

要点のまとめ

- 汎化性能とは学習に使用した入力だけでなく、新たな入力に対する予測性能
  学習誤差ではなく汎化誤差が小さいものが良い性能を持ったモデル。
  汎化誤差は通常、学習データとは別に収集された検証データでの性能を測ることで推定。

- モデルの汎化性能測定

  ① ホールドアウト法 ： 有限のデータを学習用とテスト用の２つに分割して使用。
  手元にデータが大量にないと、良い性能評価を与えない欠点有り。

  ② クロスバリデーション ： ホールドアウト法の欠点に対応。
  （交差検証） データをいくつかに分割し、学習用とテスト用に分ける。
  この分け方は数パターン準備し、各パターンで精度を検証。
  これらの精度の平均値で汎化性能を測定。

実装演習レポート【機械学習】 非線形回帰モデル

実装演習結果

- 設定
  - ボストンの住宅データセットを線形回帰モデルで分析
  - 適切な査定結果が必要
    - 高すぎても安すぎても会社に損害がある
- 課題
  - 部屋数が4で犯罪率が0.3の物件はいくらになるか？

⇒ 17353ドル

【サマリーと考察】

・ 線形回帰モデルの実装演習課題での各説明変数に多項式特徴量（x^2）を作成。
非線形な関係の学習を実施。

パラメータ推定したものから部屋数4、犯罪率が0.3のときの価格を17353ドルと予測。
線形学習モデルでの予測値4240ドルと大きく変化。
多項式特徴量を追加することでの予測値への影響有り。
線形回帰モデルもしくは非線形回帰モデルの選択は適切に判断する。

実装演習レポート【機械学習】非線形回帰モデル

実装演習結果（コード）

ライブラリをロード

```
[3]  # ライブラリをロード
     import pandas as pd
     import numpy as np
     from sklearn.linear_model import LinearRegression
     from sklearn.datasets import load_boston
     from sklearn.preprocessing import PolynomialFeatures
```

データをロード

```
[4]  # データをロード
     boston = load_boston()
```

データフレームに変換

```
[5]  df = pd.DataFrame(data=boston.data , columns = boston.feature_names)
```

```
[6]  df["PRICE"] = np.array(boston.target)
```

重回帰分析（2変数_多項式）

```
[8]  # 説明変数
     data = df.loc[:,["CRIM","RM"]].values
     # 説明変数（多項式）
     polynomial = PolynomialFeatures(degree=2,include_bias=False)
     data_2 = polynomial.fit_transform(data)
     # 目的変数
     target = df.loc[:,["PRICE"]].values
```

```
[9]  # オブジェクトを作成
     model = LinearRegression()
```

```
[10] # fit関数でパラメータ推定
     model.fit(data_2,target)

     LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
[21] # 予測（犯罪率 0.3 , 部屋数 4）
     model.predict([[0.3,4,0.09,1.2,16]])

     array([[17.35342482]])
```

実装演習レポート【機械学習】非線形回帰モデル

参考図書レポート

- 正則化について

  モデルを学習する際に、複雑さが増すことに対するペナルティを設け、過学習を防ぐ。
  機械学習で一般的に使用されるのは、L1正則化とL2正則化。

  ・L1正則化（Lasso回帰）：特定のデータの重みを0にし、不要なデータを削除。
  ・L2正則化（リッジ回帰）：データの大きさに応じて0に近づけて、滑らかなモデルにする。

  但し、正則化しすぎても性能がでない（under fitting）。
  バリアンスは小さくなるが、バイアスが大きくなる。

  バリアンスとはモデルの分散＝複雑さを意味し、バイアスは予測値と真値のズレを意味する。
  この2つはトレードオフの関係。バリアンスとバイアスのバランスの取れたモデル選択が重要。

実装演習レポート 【機械学習】 ロジスティック回帰モデル

要点のまとめ

- 分類問題と解くための教師あり機械学習モデル
  入力とm次元パラメータの線形結合をシグモイド関数に入力して、y＝1になる確率を出力。
  その確率の値で結果を予測する。

確率の値を算出。その値が閾値に対して大きいか小さいかで分類する。

$$P(Y=1|\boldsymbol{x}) = \underbrace{\sigma}_{\text{シグモイド関数}} \times \underbrace{(w_0 + w_1 x_1 + \cdots + w_m x_m)}_{\text{入力(x)とパラメータ(w)の線形結合}}$$

- シグモイド関数は以下の式。出力は必ず0～1の値をとり、確率を表現する。

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

実装演習レポート【機械学習】ロジスティック回帰モデル

要点のまとめ

- パラメータの推定
  ロジスティック回帰の確率分布はベルヌーイ分布を使用。
  （確率pで1、確率1-pで0をとる離散確率分布）
  データからデータを生成したであろう尤もらしい分布（パラメータ）を推定したい。
  尤度関数Eを最大にするようなパラメータを探索する、この手法を最尤推定という。

$$P(y_1, y_2, \cdots, y_n | w_0, w_1, \cdots, w_m) = \prod_{i=1}^{n} p_i^{y_i} (1 - p_i)^{1-y_i}$$

尤度関数

：既知　：未知

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

$$= L(\boldsymbol{w})$$

尤度関数はパラメータのみに依存する関数

実装演習レポート 【機械学習】 ロジスティック回帰モデル

要点のまとめ

- 勾配降下法（Gradient descent）
  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、以下の式で計算。
  パラメータが更新されなくなったとき、そのパラメータが最適な解ということになる。

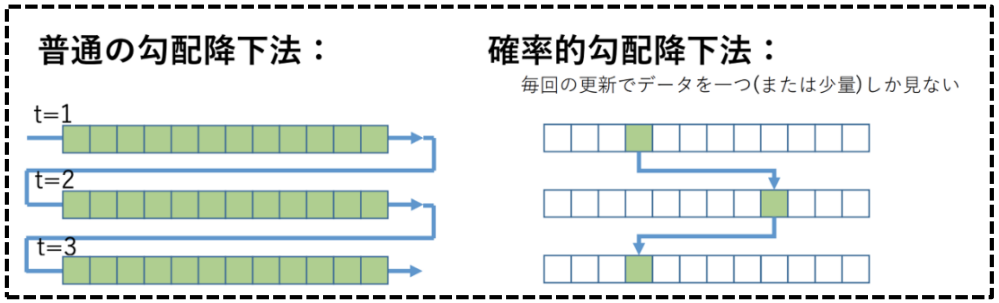
$$\boldsymbol{w}(k+1) = \boldsymbol{w}^k - \eta \frac{\partial E(\boldsymbol{w})}{\partial \boldsymbol{w}} = \boldsymbol{w}^{(k)} + \eta \sum_{i=1}^{n} (y_i - p_i)\boldsymbol{x}_i$$

η：学習率
（パラメータの収束のしやすさを調整）

  勾配降下法では、パラメータの更新にN個全てのデータに対する和を求める必要がある。
  nが巨大になったときにメモリ容量が足りない、計算時間が莫大になるという問題がある。
  これに対しては、確率的勾配降下法を利用して解決する。

- 確率的勾配降下法
  データを一つずつランダムに選んでパラメータを更新。
  効率良く最適な解を探索可能。

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



実装演習レポート 【機械学習】 ロジスティック回帰モデル

要点のまとめ

- 分類の評価方法
  評価方法には「正解率」が主に使用される。

$$\text{正解率} = \frac{TP + TN}{TP + FN + FP + TN}$$

混同行列

		検証用データの結果	
		positive	negative
モデルの予測結果	positive	真陽性 (True Positive) ~正しくpositiveと判別した個数	偽陰性 (False Positive) ~間違えてpositiveと判別した個数
	negative	偽陽性 (False Negative) ~間違えてNegativeと判別した個数	真陰性 (True Negative) ~正しくNegativeと判別した個数

実装演習レポート【機械学習】ロジスティック回帰モデル

実装演習結果

- 設定
  - タイタニックの乗客データを利用しロジスティック回帰モデルを作成
  - 特徴量抽出をしてみる
- 課題
  - 年齢が30歳で男の乗客は生き残れるか？

⇒ 生き残れない。

【サマリーと考察】

・ 学習データの特徴量を年齢と性別の２つに抽出。
年齢には欠損値があった為、欠損値から中央値に置換えを実施。
このデータでロジスティック回帰モデルを作成し、学習。

学習したモデルで年齢が30歳で男性の乗客の生存を予測した結果は、「生き残れない」。
生き残れない確率は約80%と高い。

実装演習レポート【機械学習】ロジスティック回帰モデル

実装演習結果（コード）

ライブラリ　インポート

```
[1]  import pandas as pd
     import numpy as np
```

```
[2]  cd "/content/drive/My Drive/study_ai/titanic"
```

```
/content/drive/My Drive/study_ai/titanic
```

データ読込み

```
[3]  data = pd.read_csv("train.csv")
```

```
[4]  train = pd.DataFrame()
     train["Age"] = data["Age"]
     train["Gender"] = data["Sex"].map({"female" : 1 , "male" : 0}).astype(int)
```

```
[5]  # 欠損値確認
     train.isnull().sum()
```

```
Age       177
Gender      0
dtype: int64
```

```
[6]  # 欠損値を中央値に置換え
     train["Age"] = train["Age"].fillna(train["Age"].mean())
```

```
[7]  # ターゲットデータ作成
     target = pd.DataFrame(data["Survived"])
     target = target.values
     target = target.flatten()
```

モデル作成、学習、予測

```
[8]  # ロジスティック回帰学習
     from sklearn.linear_model import LogisticRegression
     LogisticRegression = LogisticRegression()
     model = LogisticRegression.fit(train,target)
```

```
[9]  # 予測
     model.predict([[30,0]])
```

```
array([0])
```

```
[10] # 予測（確率）
     model.predict_proba([[30,0]])
```

```
array([[0.80668123, 0.19331877]])
```

実装演習レポート 【機械学習】 ロジスティック回帰モデル

参考レポート

- ロジスティック回帰とは、ある事象の発生率を判別すること。

  ロジスティック回帰から求めた「事象の起こる確率」をy とすると、
  「事象の起こらない確率」は（1-y）となる。
  この起こる確率と起こらない確率の比を「オッズ比」といい、
  確率と同時に事象が起こる確実性を表します。

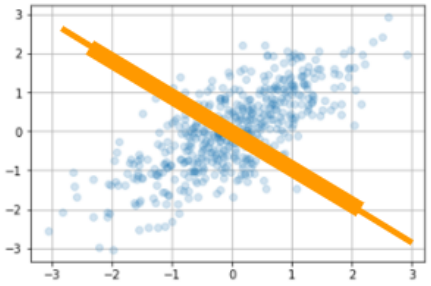
  このオッズを利用すれば各説明変数が目的変数に与える影響力を調べることが可能
  （オッズ比の値が大きいほど、その説明変数によって目的変数が大きく変動する。）

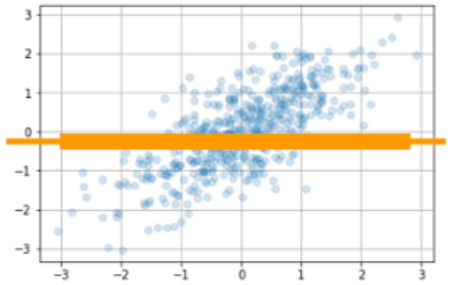
実装演習レポート【機械学習】 主成分分析

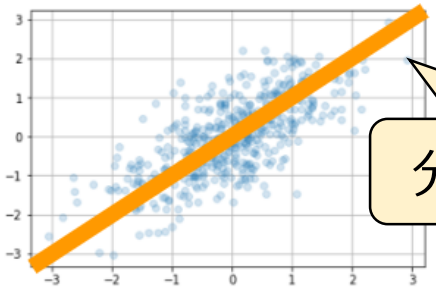
要点のまとめ

- 主成分分析とは多変量データを持つ構造をより少数個の指標に圧縮すること。
  但し、変数の個数を減らすことに伴う、情報量の損失はなるべく小さくする。
  情報量を分散の大きさと捉え、分散が最大となる射影軸を探索する。

$$\boldsymbol{s}_j = (s_{1j}, \cdots, s_{nj})^T = \bar{X}\boldsymbol{a}_j \qquad \boldsymbol{a}_j \in \mathbb{R}^m$$







分散最大の射影軸

線形変換後の分散

$$Var(\boldsymbol{s}_j) = \frac{1}{n}\boldsymbol{s}_j^T\boldsymbol{s}_j = \frac{1}{n}(\bar{X}\boldsymbol{a}_j)^T(\bar{X}\boldsymbol{a}_j) = \frac{1}{n}\boldsymbol{a}_j^T\bar{X}\bar{X}\boldsymbol{a}_j = \boldsymbol{a}_j^T Var(\bar{X})\boldsymbol{a}_j$$

実装演習レポート【機械学習】 主成分分析

要点のまとめ

- 分散が最大となる最適解を算出する為に、ラグランジェ関数を使用する。

ラグランンジェ関数 ： $E(\boldsymbol{a}_j) = \boldsymbol{a}_j^T Var(\bar{X})\boldsymbol{a}_j - \lambda(\boldsymbol{a}_j^T \boldsymbol{a}_j - 1)$

制約条件

ラグランジェ関数を微分して、最適解を算出（微分した式の解が0になる係数を求める）。

微分 ： $\dfrac{\partial E(\boldsymbol{a}_j)}{\partial \boldsymbol{a}_j} = 2Var(\bar{X})\boldsymbol{a}_j - 2\lambda \boldsymbol{a}_j = 0$

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

分散共分散行列の固有値と固有ベクトルが最適解。

実装演習レポート【機械学習】 主成分分析

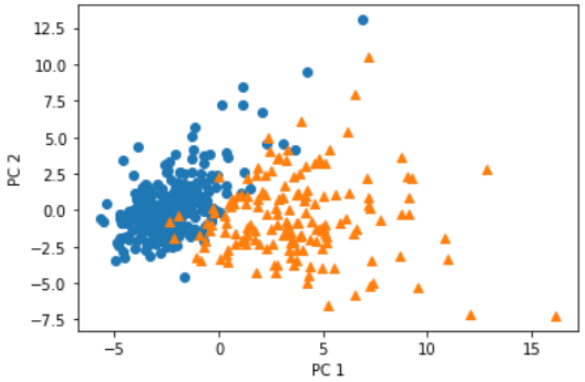
実装演習結果

- 設定
  - 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
  - 主成分を利用し2次元空間上に次元圧縮
- 課題
  - 32次元のデータを2次元上に次元圧縮した際に、うまく判別できるかを確認

【サマリーと考察】

・ 乳がん検査データでロジステッ回帰モデルを作成して、検証。精度としては十分と確認。
その後、主成分分析で2次元上に次元圧縮をして、結果を確認。

```
X_train_pca shape: (426, 2)
explained variance ratio: [0.43315126 0.19586506]
Text(0, 0.5, 'PC 2')
```



● ：良性
▲ ：悪性

次元圧縮しても、大きくは分類することが可能。
但し、境界付近では曖昧になってくる。

実装演習レポート【機械学習】主成分分析

実装演習結果（コード）

データ取得、前処理

```
[1] from sklearn.datasets import load_breast_cancer
    data_breast_cancer = load_breast_cancer()
```

```
[2] target = data_breast_cancer["target"]
```

```
[3] import pandas as pd
    y_all = pd.DataFrame(data_breast_cancer["target"],columns=["target"])
    y_all = y_all.replace({0:data_breast_cancer["target_names"][0], 1:data_breast_cancer["target_names"][1]})
    y_all.head()
```

```
[4] X_all = pd.DataFrame(data_breast_cancer["data"],columns=data_breast_cancer["feature_names"])
    X_all.head()
    X_all.shape
```

```
(569, 30)
```

```
[5] y_all = pd.get_dummies(y_all)
    y_all = y_all.drop("target_benign",axis=1)
```

```
[6] # 学習データと評価データに分ける。
    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X_all, y_all, random_state=0)
```

```
[7] # 標準化
    from sklearn.preprocessing import StandardScaler
    scaler = StandardScaler()
    X_train_scaler = scaler.fit_transform(X_train)
    X_test_scaler = scaler.fit_transform(X_test)
```

ロジスティック回帰

```
[8] from sklearn.linear_model import LogisticRegression
    LogisticRegression = LogisticRegression()
    model = LogisticRegression.fit(X_train_scaler,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:760: DataCon
  y = column_or_1d(y, warn=True)
```

検証

```
[9] print("Train score : [:.3f]".format(model.score(X_train_scaler,y_train)))
    print("Test score : [:.3f]".format(model.score(X_test_scaler,y_test)))
    from sklearn.metrics import confusion_matrix
    print("Confusion matrix:¥n[]".format(confusion_matrix(y_test,model.predict(X_test_scaler))))
```

```
Train score : 0.991
Test score : 0.958
Confusion matrix:
[[86  4]
 [ 2 51]]
```

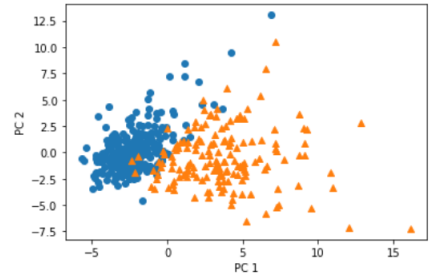
主成分分析

```
# PCA 次元数2まで圧縮
from sklearn.decomposition import PCA
pca = PCA(n_components = 2)
X_train_pca = pca.fit_transform(X_train_scaler)
print("X_train_pca shape: []".format(X_train_pca.shape))

# 寄与率
print("explained variance ratio: []".format(pca.explained_vari

# 散布図にプロット
import matplotlib.pyplot as plt
temp = pd.DataFrame(X_train_pca)
temp["Outcome"] = y_train.values
b = temp[temp["Outcome"]==0]
m = temp[temp["Outcome"]==1]
plt.scatter(x=b[0],y=b[1],marker="o")
plt.scatter(x=m[0],y=m[1],marker="^")
plt.xlabel("PC 1")
plt.ylabel("PC 2")
```

```
X_train_pca shape: (426, 2)
explained variance ratio: [0.43315126 0.19586506]
Text(0, 0.5, 'PC 2')
```



実装演習レポート【機械学習】 主成分分析

参考レポート

- 主成分分析（PCA：Principal component analysis）は、広く用いられている線形次元削減手法。

  PCAは、教師なし学習手法の一つ。

  PCAは観測値の分散を維持したまま、観測値をより少ない（ことが期待される）主成分でできた特徴量行列に射影することができる。

  pca = PCA(n_components=0.99)

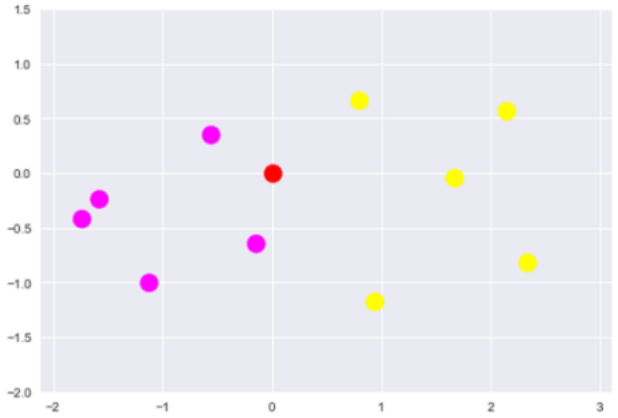
分散を99%維持しながら特徴量を削減する。
（一般的には0.95か0.99を指定する。）

実装演習レポート【機械学習】 アルゴリズム

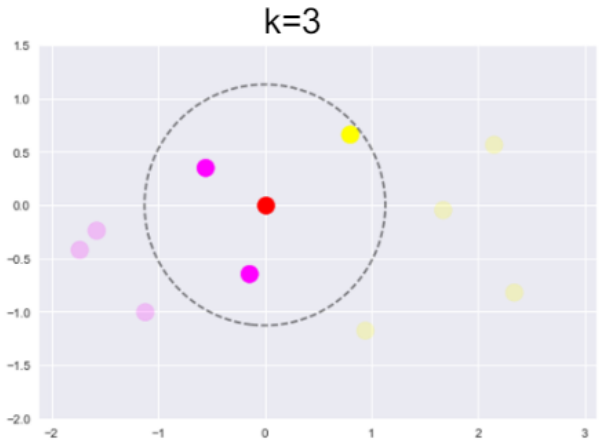
要点のまとめ

- k近傍法（KNN）
  分類問題のための機械学習手法。

  最近傍のデータをk個取ってきて、それらがもっとも多く所属するクラスに識別。



新しいデータ(赤色)を分類する



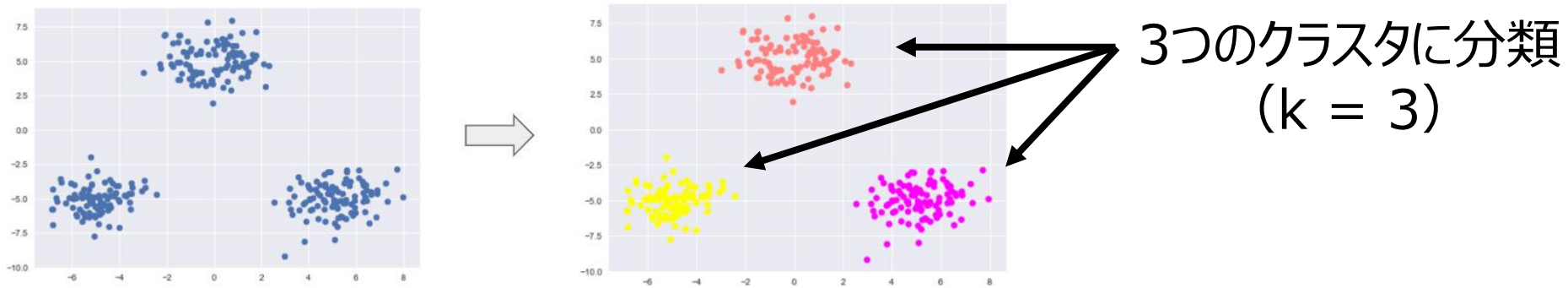
近傍の点は紫2個、黄1個なので
紫クラスに分類する

k の値を変化させると結果も変化する。また、kを大きくすると決定境界は滑らかになる。

実装演習レポート【機械学習】 アルゴリズム

要点のまとめ

- k-means
  教師なし機械学習のクラスタリング手法。与えられたデータをk個のクラスタに分類する。
  中心の初期値やkの値によってクラスタリングの結果が変化する。



k-meansのアルゴリズムは以下。

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。

実装演習レポート【機械学習】アルゴリズム

参考レポート

- k近傍法（KNN）での距離測定方法

  k近傍法での距離測定方法は下記に示す３つある。

  ① ユークリッド距離 $d_{euclidean} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

  ② マンハッタン距離 $d_{manhattan} = \sum_{i=1}^{n} |x_i - y_i|$

  ③ ミンコフスキー距離 $d_{manhattan} = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}$

  距離計量は全ての特徴量が同じスケールになっていることを前提にしているので、
  KNNクラス分類器にかける前に、特徴量を標準化しておくことが重要。

実装演習レポート【機械学習】 アルゴリズム

参考レポート

- k-means 注意点

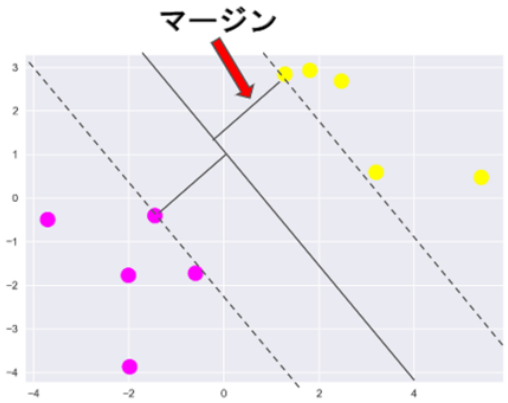
  k-means法にはいくつかの注意点がある。
  まず、k-means法クラスタリングは、クラスが（円や球のような）凸形状であることを
  仮定している。
  更に、全ての特徴量が同じスケールであることも仮定している。特徴量を標準化することで
  この仮定を満たすようにする。
  また、グループが均等である（つまりおおよそ同じくらいの数の観測値が含まれる）ことを
  仮定している。

  これらの条件が満たされるかがわからないのであれば、別のクラスタリング手法を
  検討した方がよい。

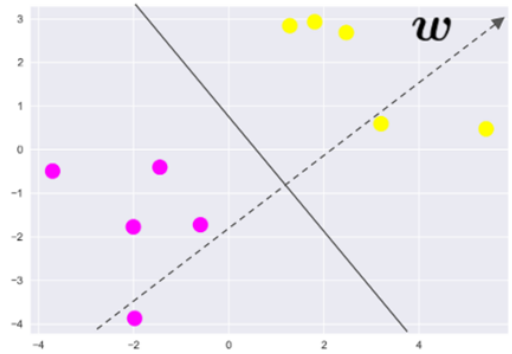
実装演習レポート【機械学習】 サポートベクターマシン

要点のまとめ

- 2クラス分類のための機械学習手法。線形モデルの正負で2値分類。
- 線形判別関数と最も近いデータ点の距離をマージンという。
  このマージンが最大となる線形判別関数を求める。



線形判別関数　$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$



この目的関数と制約条件は以下となる。

目的関数：$\min_{\boldsymbol{w},b}\frac{1}{2}||\boldsymbol{w}||^2$　　制約条件：$t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)\geq 1$　　$(i=1,2,...,n)$

上記の最適化問題をラグランジュ未定乗数法で解く。

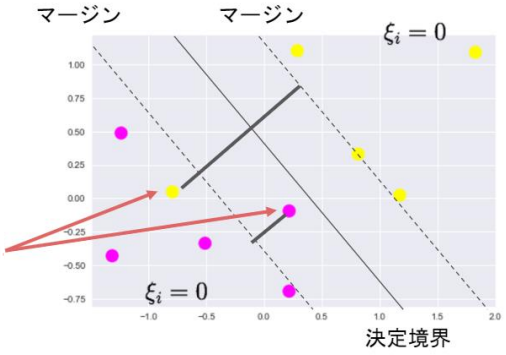
実装演習レポート 【機械学習】 サポートベクターマシン

要点のまとめ

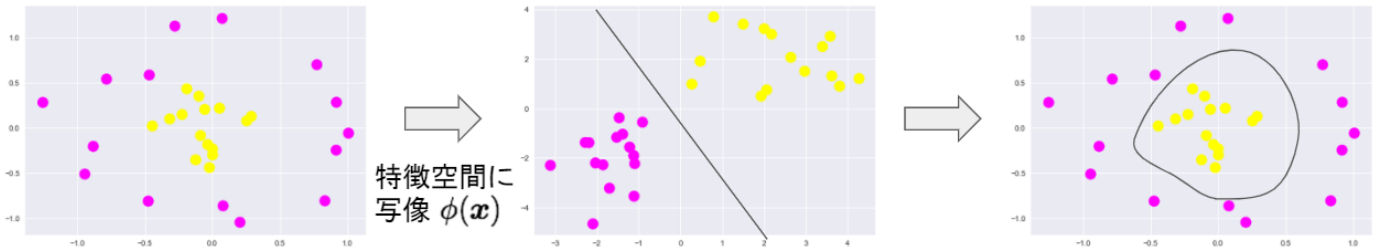
- サンプルを線形分離できないときは、誤差を許容し、誤差に対してペナルティを与える。
  この手法をソフトマージンSVMという。
  パラメータ（誤差の許容度）の大小で決定境界が変化する。

マージン内に入るデータや誤分類されたデータに対して誤差を表す変数 ξ_i を導入する

$$\xi_i = 1 - t_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b) > 0$$



- また、線形分離できないときカーネルトリックという手法で特徴空間に写像し、
  その空間で線形に分離する。



実装演習レポート【機械学習】サポートベクターマシン

実装演習結果

ラグランジュの未定乗数法

- 定義

制約 $g_i(\boldsymbol{x}) \geq 0 (i = 1, 2, ..., n)$ のもとで、$f(\boldsymbol{x})$ が最小となる $\boldsymbol{x}$ は、

変数 $\lambda_i \geq 0 (i = 1, 2, ..., n)$ を用いて新たに定義したラグランジュ関数

$L(\boldsymbol{x}, \boldsymbol{\lambda}) = f(\boldsymbol{x}) - \sum_{i=1}^{n} \lambda_i g(\boldsymbol{x})$ において $\frac{\partial L}{\partial x_j} = 0 \ (j = 1, 2, ..., m)$ を満たす

ある関数f(x)を制約g(x)の下で
最小化するxを求める際に用いられるのが
ラグランジュ関数

定義の意味は何か？ (導出)

$\frac{\partial L}{\partial x_j} = \frac{\partial L}{\partial \lambda_j} = 0$ の $\frac{\partial L}{\partial \lambda_j} = 0$ の部分は、g(x) = 0を言い換えたもの。

$\frac{\partial L}{\partial x_j} = 0$はf(x),g(x)の勾配∇f、∇gが平行であることを意味する。

勾配∇fはf(x)の等高線の法線ベクトルであるので、 ∇f、∇gが平行であるというのは
要するにf(x)の等高線とg(x)の等高線が接していることと同義になる。

実装演習レポート【機械学習】サポートベクターマシン

参考レポート

- サポートベクターマシン（SVC：support vector classifier） 注意点

SVCにはいくつかの注意点がある。
SVCは2クラス分類だけでなく、多クラス分類にも対応している。
実際、多くの場合では完全にクラスを分類することはできない。
このような場合には、超平面のマージンを最大化することと、クラス分類に失敗する
観測値数の最小化の間で、バランスをとる必要がある。
SVCでは、ハイパーパラメータCでクラス分類に失敗した観測値に対するペナルティを
指定することができる。
Cが小さいと、分類に失敗してもあまりペナルティを受けない（バイアスは大、バリアンスは小）。
Cが大きいと、分類に失敗したときのペナルティが大きくなるので、分類失敗をさけるように
なる（バイアスは小、バリアンスは大）。
適切なCを選択することが必要。