知的情報処理論
第２回レポート

濱崎 直紀

令和元年６月30日

1　Work1

問題

何らかの識別問題または回帰問題を設定し，それを機械学習により解く．さらに，評価データとして，学習データをそのまま使用，学習データとは異なるデータを使用，の 2 つの場合の性能を比較する．

1.1　概要

MNIST の 0〜9 の数字が描かれた画像に対して，機械学習によって描かれている数字を分類する分類器を作成し，その精度を測った．機械学習の手法には SVM(Support Vector Machine) を用いた．以下ではその結果と考察を述べる．

1.2　結果

分類の精度を以下に示す．

Data	Accuracy
Training data	99.0%
Test data	97.9%

1.3　考察

実際に精度を測る際に、学習データをそのまま用いて測ることをオープンテスト（open test），学習に使わずに用意しておいたテスト用のデータを用いて測ることをクローズドテスト（closed test）と言う．今回の分類に関しても両者で精度を測り，結果として，オープンテストでは 99.0%，クローズドテストでは 97.9% という結果が得られた．比較してみると，オープンテストの方がクローズドテストよりもわずかに高い精度が達成された．学習の際には学習用のデータをうまく分類できるように学習するため，学習用のデータに対しては精度が高く，それと比較して，未知のデータであるテスト用のデータに対しては低くなるのは妥当な結果であると言えるだろう．

今回のように，データを学習用とテスト用のデータに分けて，それぞれのデータを用いて学習とテストを行うことを交差検証と言う．これは，学習用のデータを分類することに適合し過ぎるあまり，未知のデータに対する分類がうまくいかない（過学習）といった問題を防ぐことが期待される．

2　Work2

問題

統計的パターン認識において，確率密度関数 $p(\boldsymbol{x}|C_i)$ が多次元正規分布で表される場合において，以下の (a) から (c) について示す．

ここで，多次元正規分布は以下の式で表される．

$$p(\boldsymbol{x}|C_i) = \frac{1}{(2\pi)^{d/2}\left|\sum_i\right|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-m_i)^t {\sum_i}^{-1}(\boldsymbol{x}-m_i)\right\}$$

(a) 識別関数 $g_i(\boldsymbol{x})$ は $\boldsymbol{x}$ の 2 次関数となる．

(b) 共分散行列が全クラスで等しい ($\sum_i = \sum_0$) と仮定した場合，識別関数 $g_i(\boldsymbol{x})$ は $\boldsymbol{x}$ の一次関数，つまり線形識別関数となる．

(c) $\sum_0$ を単位行列であるとし，事前確率が各クラスで等しい（$P(C_i) = \frac{1}{c}$；c はクラス数）とすると，識別関数 $g_i(x)$ は Nearest Neighbour 法と同じ形になる．この時のクラス C_i のプロトタイプがどのように求められるか示す。

解答

(a)

$$\begin{aligned}
g_i(\boldsymbol{x}) &= \log P(\boldsymbol{x}|C_i) + \log P(C_i) \\
&= \log \frac{1}{(2\pi)^{d/2}\left|\sum_i\right|^{1/2}} \exp\left\{-\frac{1}{2}(\boldsymbol{x}-m_i)^t {\sum_i}^{-1}(\boldsymbol{x}-m_i)\right\} + \log P(C_i) \\
&= -\frac{1}{2}\left\{(\boldsymbol{x}-m_i)^t {\sum_i}^{-1}(\boldsymbol{x}-m_i)\right\} - \frac{d}{2}\log 2\pi - \frac{1}{2}\log\left|\sum_i\right| + \log P(C_i)
\end{aligned}$$

よって，識別関数 $g_i(\boldsymbol{x})$ は $\boldsymbol{x}$ の 2 次関数となる．

(b) クラス i, j に関して考えると，共分散行列が全クラスで等しいことから

$$p(C_i|\boldsymbol{x}) = p(C_j|\boldsymbol{x})$$

つまり

$$p(\boldsymbol{x}|C_i)p(C_i) = p(\boldsymbol{x}|C_j)p(C_j)$$

両辺の対数をとって

$$\log p(\boldsymbol{x}|C_i) + \log p(C_i) = \log p(\boldsymbol{x}|C_j) + \log p(C_j) \tag{1}$$

ここで $\lambda_i = \frac{1}{(2\pi)^{d/2}\left|\sum_i\right|^{1/2}}$ とおくと $p(\boldsymbol{x}|C_i) = \lambda_i \exp\left\{-\frac{1}{2}(\boldsymbol{x}-m_i)^t {\sum_i}^{-1}(\boldsymbol{x}-m_i)\right\}$ となるので

$$\begin{aligned}
\log p(\boldsymbol{x}|C_i) &= \log\lambda_i - \frac{1}{2}(\boldsymbol{x}-m_i)^t {\sum_i}^{-1}(\boldsymbol{x}-m_i) \\
&= \log\lambda_i - \frac{1}{2}\left\{\boldsymbol{x}^t {\sum_0}^{-1}\boldsymbol{x} - 2m_i^t \sum^{-1}\boldsymbol{x} + m_i^t \sum^{-1} m_i\right\}
\end{aligned}$$

これを式（1）に代入すると

$$\begin{aligned}
&\log\lambda_0 - \frac{1}{2}\boldsymbol{x}^t {\sum_0}^{-1}\boldsymbol{x} + m_i^t {\sum_0}^{-1}\boldsymbol{x} - \frac{1}{2}m_i^t {\sum_0}^{-1} m_i + \log P(C_i) \\
&= \log\lambda_0 - \frac{1}{2}\boldsymbol{x}^t {\sum_0}^{-1}\boldsymbol{x} + m_j^t {\sum_0}^{-1}\boldsymbol{x} - \frac{1}{2}m_j^t {\sum_0}^{-1} m_j + \log P(C_j)
\end{aligned}$$

よって

$$
\begin{aligned}
g(\boldsymbol{x}) &= logP(C_i|\boldsymbol{x}) - logP(C_j|\boldsymbol{x}) \\
&= m_i^t {\sum}_0^{-1} \boldsymbol{x} - m_j^t {\sum}_0^{-1} \boldsymbol{x} - \frac{1}{2} m_i^t {\sum}_0^{-1} m_i + \log P(C_i) + \frac{1}{2} m_i^t {\sum}_0^{-1} m_i - \log P(C_i) \\
&= (m_i - m_j)^t {\sum}_0^{-1} \boldsymbol{x} + \left\{ -\frac{1}{2} m_i^t {\sum}_0^{-1} m_i + \log P(C_i) + \frac{1}{2} m_i^t {\sum}_0^{-1} m_i - \log P(C_i) \right\}
\end{aligned}
$$

よって，識別関数 $g_i(\boldsymbol{x})$ は線形識別関数となる．

(c)

$$
\begin{aligned}
g_i(\boldsymbol{x}) &= \log P(C_i|\boldsymbol{x}) \\
&= \log P(\boldsymbol{x}|C_i) + \log P(C_i) \\
&= \log P(\boldsymbol{x}|C_i) - \log c
\end{aligned}
$$

ここで，$\sum_0$ が単位行列，事前確率が各クラスで等しいということから

$$
\begin{aligned}
\log P(\boldsymbol{x}|C_i) &= \log \lambda_i - \frac{1}{2} (\boldsymbol{x} - m_i)^t {\sum}_i^{-1} (\boldsymbol{x} - m_i) \\
&= \log \lambda_0 - \frac{1}{2} ||\boldsymbol{x} - m_i||^2
\end{aligned}
$$

よって

$$
\begin{aligned}
g_i(\boldsymbol{x}) &= \log P(\boldsymbol{x}|C_i) - \log c \\
&= \log \lambda_0 - \frac{1}{2} ||\boldsymbol{x} - m_i||^2 - \log c \\
&= -\frac{1}{2} ||\boldsymbol{x} - m_i||^2 + \log \frac{\lambda_0}{c}
\end{aligned}
$$

ゆえに

$$
\begin{aligned}
\arg \max_i g_i(\boldsymbol{x}) &= \arg \max_i \left\{ -\frac{1}{2} ||\boldsymbol{x} - m_i||^2 + \log \frac{\lambda_0}{c} \right\} \\
&= \arg \max_i \left\{ -||\boldsymbol{x} - m_i||^2 \right\} \\
&= \arg \min_i \left\{ ||\boldsymbol{x} - m_i||^2 \right\}
\end{aligned}
$$

付録 A　プログラム（Work1）

ソースコード 1　SVM

```
# 識別機を学習するプログラム

import matplotlib.pyplot as plt
import os
import pickle
from sklearn.externals import joblib
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
os.chdir(os.path.dirname(os.path.abspath(__file__)))


if __name__ == '__main__':
    with open('dataset/mnist.pkl', 'rb') as f:
        dataset = pickle.load(f)

    # 輝度値を０～１に変換
    dataset['train_img'] = dataset['train_img'] / 255.0
    dataset['test_img'] = dataset['test_img'] / 255.0

    # SVM のインスタンスを作成
    model = SVC(kernel='rbf', gamma='scale', random_state=111)

    print('Learning start...')
    model.fit(dataset['train_img'], dataset['train_label'])
    print('Finish!')

    # トレーニングデータに対する精度
    pred_train = model.predict(dataset['train_img'])
    accuracy_train = accuracy_score(dataset['train_label'], pred_train)
    print('accuracy for training data :{:.4f}'.format(accuracy_train))

    # テストデータに対する精度
    pred_test = model.predict(dataset['test_img'])
    accuracy_test = accuracy_score(dataset['test_label'], pred_test)
    print('accuracy for test data :{:.4f}'.format(accuracy_test))

    while 1:
        save_check = input('save model?(y/n) : ')
        if save_check == 'y' or save_check == 'n':
            break
```

```
            else:
                print('Error. Please, input again.')

        if save_check == 'y':
            # dirname で指定した名前のファイル（出力先のファイル）がなければ作る
            dirname = 'classifier'
            if not os.path.exists('{}'.format(dirname)):
                os.mkdir('{}'.format(dirname))

            # モデルの保存
            joblib.dump(model, '{}/model.joblib'.format(dirname), compress=True)
            print('\"{}\"にモデルを保存'.format(dirname))
```