レポート提出票

科目名: 情報工学実験３

実験課題名: 課題２　パターン認識

実施日: 2024年 5月 23日

学籍番号: 4622045

氏名: 小澤　翼

共同実験者: ＿＿＿＿＿＿　＿＿＿＿＿＿

＿＿＿＿＿＿　＿＿＿＿＿＿

＿＿＿＿＿＿　＿＿＿＿＿＿

＿＿＿＿＿＿　＿＿＿＿＿＿

1　背景

今回扱うパターン認識は、特徴を抽出し数式にする段階と特徴に基づいてデータをあらかじめ定められたクラスに分類する段階があり、特徴をクラスに分類するために識別器を作成する必要がある。

2　目的

- 課題を通じて kNN の理解を深める
- numpy の基本的な使い方を習得する
- 実際に得られた結果から kNN の特徴を理解する

3　課題

3.1　課題 1

3.1.1　課題 1-1

学習データとテストデータとの L2 ノルムを計算せよ。
以下はそのソースコードである。

ソースコード 1: 課題 1-1

```
  #課題 1-1:学習データとテストデータとのL2 ノルムを計算します。
#np.linalg.norm を用いる
def get_dist(X_train,X_test):

    dist = np.linalg.norm(X_train - X_test, axis=1)
    return dist #L2 ノルムを返す
#実装結果をテスト
#各学習データとテストデータ 1サンプルのL2 ノルムを 10 個出力
print(get_dist(X_train,X_test[0])[:10]) #出力: [ 32.55520235 ...
```

このソースコードの結果は以下の通りである。

ソースコード 2: 課題 1-1 の結果

```
 [ 32.55520235 159.95907945 152.45350406 70.44572663 195.2440017
 520.20120665 1015.03242165 180.25164216 560.05190018 53.99136598]
```

3.1.2　課題 1-2

課題 1-1 で求めた L2 ノルムを昇順にソートし、最も小さいものから k 個 (今回は 5) のデータ点のクラスを調べよ。
以下はそのソースコードである。

ソースコード 3: 課題 1-2

```
#課題 1-2: 求めたL2 ノルムを昇順にソートし、最も小さい k 個(今回は 5)のデータを求め
    、そのクラスを調べてください。
#np.argsort, スライスを用いる
def get_knn_classes(y_train,dist,k):
    ##ここに実装
    #求めたL2 ノルムを昇順にソート
    vec_sort = np.argsort(dist)
    #最も小さいものからk 個のデータ点のクラスを調べる
    #ヒント 1: y_train(学習データのクラスをもつ配列)を使用します

    knn_class = y_train[vec_sort[:k]]
    return knn_class #クラスを返す
#実装結果をテスト
k = 5

#結果を 10個出力
for x_test in X_test[:10]:
    dist=get_dist(X_train,x_test)
    s_idx = get_knn_classes(y_train, dist, 5)
    print(s_idx) #出力: [2 2 2 1 1].....
```

このソースコードの結果は以下の通りである。

ソースコード 4: 課題 1-2 の結果

```
[2 2 2 1 1]
[2 1 1 1 2]
[2 2 2 1 2]
[1 1 1 1 1]
[1 1 1 1 1]
[0 0 1 0 0]
[1 1 2 1 1]
[1 2 2 2 1]
[1 1 1 1 2]
[0 0 0 0 0]
```

3.1.3 課題 1-3

課題 1-2 で求めた 5 個のクラスの中で、最も多く現れるクラスを求めよ。
以下はそのソースコードである。

ソースコード 5: 課題 1-3

```
#課題 1-3:課題 1-2で求めた 5個のデータ点の中で、最も多く現れるクラスを求める
#np.unique, np.argmax を用いる
def get_class(knn_class):
    ##ここに実装
    uni, fr = np.unique(knn_class, return_counts=True)
    pred = uni[np.argmax(fr)]
    return pred #最も多く現れるクラスを返す
#実装結果をテスト
k = 5
```

```
10
11 #結果を10個出力
12 for x_test in X_test[:10]:
13     dist=get_dist(X_train,x_test)
14     s_idx = get_knn_classes(y_train, dist, k)
15     pred_class = get_class(s_idx)
16     print(pred_class) #出力: 2,1,2,1...
```

このソースコードの結果は以下の通りである。

ソースコード 6: 課題 1-3 の結果

```
1 2
2 1
3 2
4 1
5 1
6 0
7 1
8 2
9 1
10 0
```

3.2 課題 2

k の値を 1 から 100 までの数に変更し、kNN を実行せよ。k ごとの結果をグラフへ描画し、また最も正解率が高くなるときの k(候補が複数ある場合は、k が最小のもの) と正解率を表示せよ。

以下はそのソースコードである。

ソースコード 7: 課題 2

```
1 #k の値を 1 から 100 までの数に定める
2 k_range = range(1, 101)
3
4 #各k の正解率を入れる配列
5 accuracy_2 = []
6
7 #k を 1 から 100 まで動かし、それぞれの正解率を求める
8 #ヒント 1:appeend を使うと、配列に要素を追加できる
9 for k in k_range:
10     y_pred = [knn_classify(X_train, y_train, x_test, k) for x_test in X_test
       ]
11     accuracy = accuracy_score(y_test, y_pred)
12     accuracy_2.append(accuracy)
13 #正解率が最大となるk を求める関数
14 def find_k(accuracy):
15     #正解率が最大となるk のインデックスを取得
16
17     #求めたk とその時の正解率を表示する
18     return np.argmax(accuracy)+1, np.max(accuracy)
19 #正解率が最大となるk を求める
20 find_k(accuracy_2)
21 def show_graph(accuracy):
```

```
    #各k に対する Accuracy を折れ線グラフとしてプロット
    fig, ax = plt.subplots(figsize=(8,6))
    ax.plot(k_range, accuracy)
    ax.set_xlabel('Number of Neighbors K', fontsize=14)
    ax.set_ylabel('Accuracy', fontsize=14)
#グラフを表示
show_graph(accuracy_2)
```

このソースコードの結果は以下の通りである。正解率の最大とその時の k の値

図 1: k の値による正解率の変化

正解率: 0.8148148148148148　　k: 20

3.3 課題 3

標準正規化を行ってから課題 2 と同様の実験を行え。標準正規化は以下の式で求められる。

$$\overline{x} = \frac{x - x_{mean}}{\sigma_x}$$

x_{mean}: 平均

σ_x: 標準偏差

numpy の mean 関数を用いることで平均が、std 関数を用いることで標準偏差を求めることでき、テストデータを標準化する際は、学習データの平均と標準偏差を用いて行うようにせよ。

以下はそのソースコードである。

ソースコード 8: 課題 3

```
#学習データの平均を求める(ヒント:mean 関数を使う、axis を指定する)
X_train_a=np.mean(X_train,axis=0)
#学習データの標準偏差を求める(ヒント:std 関数を使う、axis を指定する)
X_train_std=np.std(X_train,axis=0)
```

```
#学習データを標準正規化する
X_train_stn = (X_train - X_train_a) / X_train_std
#テストデータを標準正規化する
X_test_stn = (X_test - X_train_a) / X_train_std
#k の値を 1 から 100 までの数に定める
k_range = range(1, 101)

#各k の正解率を入れる配列
accuracy_3 = []

#k を 1 から 100 まで動かし、それぞれの正解率を求める
#ヒント 1:appeend を使うと、配列に要素を追加できる
for k in k_range:
    #課題 2と同様に実装
    y_pred = [knn_classify(X_train_stn, y_train, x_test_stn, k) for
        x_test_stn in X_test_stn]
    accuracy = accuracy_score(y_test, y_pred)
    accuracy_3.append(accuracy)
#正解率が最大となるk を求める
find_k(accuracy_3)
#グラフを表示
show_graph(accuracy_3)
```

このソースコードの結果は以下の通りである。正解率の最大とその時の k の値

図 2: k の値による正解率の変化

正解率: 0.9814814814814815　　k: 7

この課題 2，3 の結果を比較すると課題 3 の方が k と k+1 の振れ幅が少なく、ほとんど単調減少のような形となって居り安定しているように見える。

3.4　課題 4

3.4.1　課題 4.A

データが不均衡である場合に k を大きくしても近傍のデータポイントが不十分であったり、多数派クラスに属するデータポイントが近傍に多く含まれるため、少数派クラスに属するデータポイントを正しく分類できない場合があり、今回はそのような状況ではないかと推測できる。

3.4.2　課題 4.B

標準偏差を使用することで、近傍のデータポイントの距離のばらつきを考慮できると思われる。

4　まとめ

kNN の k の値による正解率の変動や計算方法の違いでも正解率に変動することを確認することが出来た。