K-Nearest Neighbor
with Feedforward Feature Selection

จัดทำโดย

นายปัณณวิชญ์ พันธ์วงศ์ 600610752

เสนอ

รศ.ดร.ศันสนีย์ เอื้อพันธ์วิริยะกุล

รายงานเล่มนี้เป็นส่วนหนึ่งของวิชา

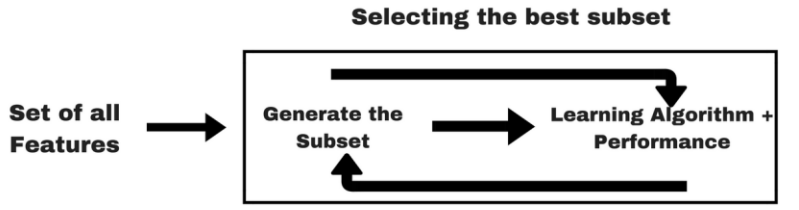
261754

ภาคเรียนที่ 1 ปีการศึกษา 2564

1. Theories and Related Method

1.1. Feedforward Feature Selection

เป็นกระบวนการวนซ้ำซึ่งเริ่มต้นด้วยการไม่มีคุณสมลักษณะในแบบจำลอง หรือ โมเดล ซึ่งในการทำซ้ำแต่ละครั้งจะทำการเพิ่มคุณลักษณะที่ช่วยปรับปรุงโมเดลจนได้โมลเดลที่ดีที่สุด โดยมีผังการทำงานดังนี้



รูปที่ 1 แผนภาพการทำงาน Feedforward feature selection

1.2. Euclidean distance

ระยะทางแบบยุคลิด คือ ระยะทางปกติระหว่างจุดสองจุดในแนวเส้นตรง โดยในรายงานเล่มนี้ใช้ในการหาระยะห่างระหว่างข้อมูลใดๆ โดยมีสมการดังนี้

$$d(p,q) = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$$

1.3. K-Nearest Neighbour Algorithm (K-NN)

ขั้นตอนวิธีการเพื่อนบ้านใกล้ที่สุด เป็นวิธีที่ใช้ในการจัดแบ่งคลาส โดยเทคนิคนี้จะตัดสินใจว่าคลาสใดที่จะแทนเงื่อนไขหรือกรณีใหม่ๆ ได้บ้าง โดยการตรวจสอบจำนวนบางจำนวน (“ K” ในขั้นตอนวิธีการเพื่อนบ้านใกล้ที่สุด) ของกรณีหรือเงื่อนไขที่เหมือนกันหรือใกล้เคียงกันมากที่สุด โดยจะหาผลรวม (Count Up) ของจำนวนเงื่อนไข หรือกรณีต่างๆ สำหรับแต่ละคลาส และกำหนดเงื่อนไขใหม่ๆ ให้คลาสที่เหมือนกันกับคลาสที่ใกล้เคียงกันมากที่สุด โดยมีขั้นตอนวิธีการดังนี้

1. กำหนดขนาดของ K
2. คำนวณระยะห่างของข้อมูลที่ต้องการพิจารณากับข้อมูลตัวอย่าง โดยใช้ระยะทางแบบยุคลิด เป็นต้น
3. จัดเรียงลำดับของระยะห่าง และเลือกพิจารณาชุดข้อมูลที่ใกล้จุดที่ต้องการพิจารณาตามจำนวน K ที่กาหนดไว้
4. พิจารณาข้อมูลจำนวน k ชุด และสังเกตว่ากลุ่ม (class) ไหนที่ใกล้จุดที่พิจารณาเป็นจำนวนมากที่สุด
5. กำหนด class ให้กับจุดที่พิจารณา (class) ที่ใกล้จุดพิจารณามากที่สุด

1.4. Cross-validation Test

การวัดประสิทธิภาพด้วยวิธี Cross-validation นี้จะทำการแบ่งข้อมูลออกเป็นหลายส่วน (มักจะแสดงด้วยค่า k) เช่น 5-fold cross-validation คือ ทำการแบ่งข้อมูลออกเป็น 5 ส่วน โดยที่แต่ละส่วนมีจำนวนข้อมูลเท่ากัน หรือ 10-fold cross-validation คือ การแบ่งข้อมูลออกเป็น 10 ส่วน โดยที่แต่ละส่วนมีจำนวนข้อมูลเท่ากัน หลังจากนั้นข้อมูลหนึ่งส่วนจะใช้เป็นตัวทดสอบประสิทธิภาพของโมเดล ทำวนไปเช่นนี้จนครบจำนวนที่แบ่งไว้ ซึ่งในการทดลองนี้จะใช้ 10-fold cross-validation

2. Algorithm (Flow Chart)

อันดับแรกจะทำการ shuffle ข้อมูลทั้งหมด จากนั้นจะทำการทดลองโดยจะทำการแบ่ง 10-fold cross-validation จากข้อมูล 351 records กล่าวคือ ข้อมูลสำหรับการฝึกสอนจัดกลุ่ม ของคลาสใดๆ มี 316 และสำหรับทดสอบ 31 records โดยข้อมูลสำหรับฝึกสอนจะจัดกลุ่มโดยใช้ K-NN โดย มี K = 3 จากนั้นจะทำการวัดผลลัพธ์

Shuffle data

For ***subset*** *in* ***Feedforward selection***

data = subset_feature

For ***i, j*** *in K-****fold cross validation***

- *train_sample: data [0 -> i] U data [j -> N]*
- *test_sample: data [i -> j]*
- *distance: Calculate Euclidean distance (test_sample, train_sample)*
- *sort_distance: sorted(distance)*
- *predict: KNN (sort_distance, k=3)*
- *Show Result*

Evaluation Result

รูปที่ 2 pseudo code ของระบบ

3. Experimental

ข้อมูล Ionosphere data set มีทั้งหมด 351 samples โดยแบ่งออกเป็น 2 class คือ good (g) และ bad (b) โดย g class มี 224 sample และ b class มี 127 sample ใน data set ชุดนี้มี features ทั้งหมด 34 features

ผู้จัดทำได้ทำการทดลองในการหา subset features ที่ดีที่สุด โดยใช้ Feedforward selection ซึ่งผู้จัดทำได้ทำการบันทึก subset features ที่ดีที่สุด และที่แย่ที่สุด โดยใช้มาตรวัดความแม่นยำในการเปรียบเทียบโดยจะแสดงเป็น Confusion Matrix และ Accuracy

Best subsets features					
Fold 1	**Predict**	**Acc**	**Fold 2**	**Predict**	**Acc**
Actual	15 2 0 18	0.943	**Actual**	15 3 1 16	0.886
Fold 3	**Predict**	**Acc**	**Fold 4**	**Predict**	**Acc**
Actual	12 5 0 18	0.857	**Actual**	13 4 2 16	0.829
Fold 5	**Predict**	**Acc**	**Fold 6**	**Predict**	**Acc**
Actual	16 2 1 16	0.914	**Actual**	16 1 2 16	0.914
Fold 7	**Predict**	**Acc**	**Fold 8**	**Predict**	**Acc**
Actual	15 3 0 17	0.914	**Actual**	3 1 0 31	0.971
Fold 9	**Predict**	**Acc**	**Fold 10**	**Predict**	**Acc**
Actual	0 0 2 33	0.943	**Actual**	0 0 0 36	1.000

รูปที่ 3 confusion matrix และ accuracy ของ subset ที่ดีที่สุด

Worst subsets features					
Fold 1	**Predict**	**Acc**	**Fold 2**	**Predict**	**Acc**
Actual	17 0 18 0	0.486	**Actual**	0 18 0 17	0.486
Fold 3	**Predict**	**Acc**	**Fold 4**	**Predict**	**Acc**
Actual	0 17 0 18	0.514	**Actual**	0 17 0 18	0.514
Fold 5	**Predict**	**Acc**	**Fold 6**	**Predict**	**Acc**
Actual	0 18 0 17	0.486	**Actual**	0 17 0 18	0.514
Fold 7	**Predict**	**Acc**	**Fold 8**	**Predict**	**Acc**
Actual	0 18 0 17	0.486	**Actual**	0 4 0 31	0.886
Fold 9	**Predict**	**Acc**	**Fold 10**	**Predict**	**Acc**
Actual	0 0 0 35	1.000	**Actual**	0 0 0 36	1.000

รูปที่ 4 confusion matrix และ accuracy ของ subset ที่แย่ที่สุด

จากรูปที่ 3 subset features ที่เลือกใช้ คือ feature ที่ 4 ถึง 8 พบว่าให้ accuracy ที่ดีที่สุด โดยเฉลี่ยแล้ว 10 folds มีค่าความแม่นยำถึง 91.7%

จากรูปที่ 4 subset features ที่เลือกใช้ คือ feature ที่ 1 พบว่าให้ accuracy ที่แย่ที่สุด โดย เฉลี่ยแล้ว 10 folds มีค่าความแม่น 63.7%

4. Analyze the experiment

จากการทดลองพบว่าการเลือก subset features ที่ดีที่สุด โดยใช้ feedforward selection คือ feature ที่ 4 ถึง 8 และใช้ K-NN ในการจำแนกประเภท พบว่าให้ accuracy โดยเฉลี่ย 91.7% โดยพบว่า fold ที่ 8,9 และ 10 ไม่มีสมาชิกของ class bad อยู่เลย อันเนื่องมาจาก จำนวนชุดข้อมูลของ class good และ class bad มีจำนวนสมาชิกไม่เท่ากัน ซึ่งอาจจะทำให้โอกาสในการจำแนกประเภทที่จะอยู่ใน class good มีมากกว่า class bad

5. Appendix

```
import numpy as np
import matplotlib.pyplot as plt


def readfile():
    filename = 'ionosphere.csv'

    features = np.genfromtxt(filename, delimiter=',')[1:, :-1]

    targets = np.unique(np.genfromtxt(filename, delimiter=',', dtype=str)[
        1:, -1], return_inverse=True)[1]  # g = 1 , b = 0

    data = np.concatenate((features, targets.reshape(-1, 1)), axis=1)

    features = data[:, :-1]
    targets = data[:, -1]
    return features, targets


def plotBar(X):
    plt.bar(np.arange(X.shape[1]), mean_abs_diff(X), color='red')
    plt.xlabel("feature")
    plt.ylabel("value")
    plt.title("mean absolute difference")
    plt.show()


def euclidean_distance(p1, p2, label):
    distance = []

    for i in range(p1.shape[0]):
        dis = np.sqrt(np.sum((p1[i, :]-p2)**2, axis=1))
        distance.append(list(zip(dis, label)))

    return np.array(distance)

```

```
42.
43. def knn(distance, k=3):
44.     k_nearest = []
45.     for i in range(distance.shape[0]):
46.         # sorted
47.         dis_sorted = distance[i][distance[i][:, 0].argsort()]
48.
49.         # k-nearest distances
50.         unique, counts = np.unique(dis_sorted[:k, :][:, 1],
    return_counts=True)
51.         k_nearest.append(unique[counts == counts.max()].item())
52.
53.     return np.array(k_nearest)
54.
55.
56. def cross_validations_split(shape, folds):
57.     fold_size = int(shape * folds/100)
58.     k = 0
59.     index = []
60.     for i in range(1, folds+1):
61.         index.append([k, i*fold_size]) if (i <
62.                                            folds) else index.append([k,
    shape])
63.         k = i*fold_size
64.     return index
65.
66.
67. def confusion_matrix(y_pred, y_true):
68.     matrix = np.zeros(((np.amax(y_true))+1, (np.amax(y_true))+1))
69.     for i in range(y_pred.shape[0]):
70.         matrix[(y_true[i]), (y_pred[i])] += 1
71.     return matrix
72.
73.
74. if __name__ == "__main__":
75.     X, Y = readfile()
76.
77.     max = 0
78.     min = 100
79.     id_max = [0, 0]
80.     id_min = [0, 0]
81.     for m in range(X.shape[1]):
82.         for n in range(m, X.shape[1]):
83.             print("index: ", m, n)
84.             data = np.concatenate((X[:, m:n+1], Y.reshape(-1, 1)), axis=1)
85.
86.             conf_arr = []
87.             acc_arr = []
88.             accuracy = 0
89.             for i, j in cross_validations_split(data.shape[0], 10):
90.                 train = np.concatenate((data[:i], data[j:])).copy()
91.                 test = data[i:j].copy()
92.                 x_train, y_train = train[:, :-1], train[:, -1]
93.                 x_test, y_test = test[:, :-1], test[:, -1]
94.
95.                 # find eculidean distance
```

```
                distance = euclidean_distance(x_test, x_train, y_train)

                # calculate KNN
                pred = knn(distance, 3)

                result = confusion_matrix(pred.astype(int),
y_test.astype(int))
                conf_arr.append(result)
                acc_arr.append(np.trace(result)*100/np.sum(result))
                accuracy += np.trace(result)*100/np.sum(result)

            print("-----------------------------------------------------------
---------")
            accuracy /= 10
            conf_arr = np.array(conf_arr)
            acc_arr = np.array(acc_arr)
            print(accuracy)
            if accuracy > max:
                max = accuracy
                id_max = [m, n]
                print(">> new max: ", max)
                np.savetxt("max_conf.csv", conf_arr.reshape(
                    10*2, 2), delimiter=",", fmt='%d')
            if accuracy < min:
                min = accuracy
                id_min = [m, n]
                print(">> new min: ", min)
                np.savetxt("min_conf.csv", conf_arr.reshape(
                    10*2, 2), delimiter=",", fmt='%d')
    print("id_max: ", id_max)
    print("id_min: ", id_min)
```