**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. **Feedforward Feature Selection**

เป็นกระบวนการวนซ้ำซึ่งเริ่มต้นด้วยการไม่มีคุณสมลักษณะในแบบจำลอง หรือ โมเดล ซึ่งในการทำซ้ำแต่ละครั้งจะทำการเพิ่มคุณลักษณะที่ช่วยปรับปรุงโมเดลจนได้โมลเดลที่ดีที่สุด โดยมีผังการทำงานดังนี้

Diagram, text

Description automatically generated with medium confidence

รูปที่ 1 แผนภาพการทำงาน Feedforward feature selection

* 1. **Euclidean distance**

ระยะทางแบบยุคลิด คือ ระยะทางปกติระหว่างจุดสองจุดในแนวเส้นตรง โดยในรายงานเล่มนี้ใช้ในการหาระยะห่างระหว่างข้อมูลใดๆ โดยมีสมการดังนี้

* 1. **K-Nearest Neighbour Algorithm (K-NN)**

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

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

1. **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

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

รูปที่ 3 confusion matrix และ accuracy ของ subset ที่ดีที่สุด

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

รูปที่ 4 confusion matrix และ accuracy ของ subset ที่แย่ที่สุด

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

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

1. **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

1. **Appendix**
2. **import** numpy **as** np
3. **import** matplotlib**.**pyplot **as** plt
4. **def** readfile**():**
5. filename **=** 'ionosphere.csv'
6. features **=** np**.**genfromtxt**(**filename**,** delimiter**=**','**)[**1**:,** **:-**1**]**
7. targets **=** np**.**unique**(**np**.**genfromtxt**(**filename**,** delimiter**=**','**,** dtype**=str)[**
8. 1**:,** **-**1**],** return\_inverse**=True)[**1**]** # g = 1 , b = 0
9. data **=** np**.**concatenate**((**features**,** targets**.**reshape**(-**1**,** 1**)),** axis**=**1**)**
10. features **=** data**[:,** **:-**1**]**
11. targets **=** data**[:,** **-**1**]**
12. **return** features**,** targets
13. **def** plotBar**(**X**):**
14. plt**.**bar**(**np**.**arange**(**X**.**shape**[**1**]),** mean\_abs\_diff**(**X**),** color**=**'red'**)**
15. plt**.**xlabel**(**"feature"**)**
16. plt**.**ylabel**(**"value"**)**
17. plt**.**title**(**"mean absolute difference"**)**
18. plt**.**show**()**
19. **def** euclidean\_distance**(**p1**,** p2**,** label**):**
20. distance **=** **[]**
21. **for** i **in** **range(**p1**.**shape**[**0**]):**
22. dis **=** np**.**sqrt**(**np**.sum((**p1**[**i**,** **:]-**p2**)\*\***2**,** axis**=**1**))**
23. distance**.**append**(list(zip(**dis**,** label**)))**
24. **return** np**.**array**(**distance**)**
25. **def** knn**(**distance**,** k**=**3**):**
26. k\_nearest **=** **[]**
27. **for** i **in** **range(**distance**.**shape**[**0**]):**
28. # sorted
29. dis\_sorted **=** distance**[**i**][**distance**[**i**][:,** 0**].**argsort**()]**
30. # k-nearest distances
31. unique**,** counts **=** np**.**unique**(**dis\_sorted**[:**k**,** **:][:,** 1**],** return\_counts**=True)**
32. k\_nearest**.**append**(**unique**[**counts **==** counts**.max()].**item**())**
33. **return** np**.**array**(**k\_nearest**)**
34. **def** cross\_validations\_split**(**shape**,** folds**):**
35. fold\_size **=** **int(**shape **\*** folds**/**100**)**
36. k **=** 0
37. index **=** **[]**
38. **for** i **in** **range(**1**,** folds**+**1**):**
39. index**.**append**([**k**,** i**\***fold\_size**])** **if** **(**i **<**
40. folds**)** **else** index**.**append**([**k**,** shape**])**
41. k **=** i**\***fold\_size
42. **return** index
43. **def** confusion\_matrix**(**y\_pred**,** y\_true**):**
44. matrix **=** np**.**zeros**(((**np**.**amax**(**y\_true**))+**1**,** **(**np**.**amax**(**y\_true**))+**1**))**
45. **for** i **in** **range(**y\_pred**.**shape**[**0**]):**
46. matrix**[(**y\_true**[**i**]),** **(**y\_pred**[**i**])]** **+=** 1
47. **return** matrix
48. **if** \_\_name\_\_ **==** "\_\_main\_\_"**:**
49. X**,** Y **=** readfile**()**
50. **max** **=** 0
51. **min** **=** 100
52. id\_max **=** **[**0**,** 0**]**
53. id\_min **=** **[**0**,** 0**]**
54. **for** m **in** **range(**X**.**shape**[**1**]):**
55. **for** n **in** **range(**m**,** X**.**shape**[**1**]):**
56. **print(**"index: "**,** m**,** n**)**
57. data **=** np**.**concatenate**((**X**[:,** m**:**n**+**1**],** Y**.**reshape**(-**1**,** 1**)),** axis**=**1**)**
58. conf\_arr **=** **[]**
59. acc\_arr **=** **[]**
60. accuracy **=** 0
61. **for** i**,** j **in** cross\_validations\_split**(**data**.**shape**[**0**],** 10**):**
62. train **=** np**.**concatenate**((**data**[:**i**],** data**[**j**:])).**copy**()**
63. test **=** data**[**i**:**j**].**copy**()**
64. x\_train**,** y\_train **=** train**[:,** **:-**1**],** train**[:,** **-**1**]**
65. x\_test**,** y\_test **=** test**[:,** **:-**1**],** test**[:,** **-**1**]**
66. # find eculidean distance
67. distance **=** euclidean\_distance**(**x\_test**,** x\_train**,** y\_train**)**
68. # calculate KNN
69. pred **=** knn**(**distance**,** 3**)**
70. result **=** confusion\_matrix**(**pred**.**astype**(int),** y\_test**.**astype**(int))**
71. conf\_arr**.**append**(**result**)**
72. acc\_arr**.**append**(**np**.**trace**(**result**)\***100**/**np**.sum(**result**))**
73. accuracy **+=** np**.**trace**(**result**)\***100**/**np**.sum(**result**)**
74. **print(**"----------------------------------------------------------------"**)**
75. accuracy **/=** 10
76. conf\_arr **=** np**.**array**(**conf\_arr**)**
77. acc\_arr **=** np**.**array**(**acc\_arr**)**
78. **print(**accuracy**)**
79. **if** accuracy **>** **max:**
80. **max** **=** accuracy
81. id\_max **=** **[**m**,** n**]**
82. **print(**">> new max: "**,** **max)**
83. np**.**savetxt**(**"max\_conf.csv"**,** conf\_arr**.**reshape**(**
84. 10**\***2**,** 2**),** delimiter**=**","**,** fmt**=**'%d'**)**
85. **if** accuracy **<** **min:**
86. **min** **=** accuracy
87. id\_min **=** **[**m**,** n**]**
88. **print(**">> new min: "**,** **min)**
89. np**.**savetxt**(**"min\_conf.csv"**,** conf\_arr**.**reshape**(**
90. 10**\***2**,** 2**),** delimiter**=**","**,** fmt**=**'%d'**)**
91. **print(**"id\_max: "**,** id\_max**)**
92. **print(**"id\_min: "**,** id\_min**)**