Софийски университет
“Св. Климент Охридски”
Факултет по математика и информатика

Проект
По
“Размити множества и приложения”
на тема

„Размита класификация на данни от Gas Dataset“

Изготвен от: Ивайло Иванов
фн. 9MI3400162, ИИОЗ, 1 курс

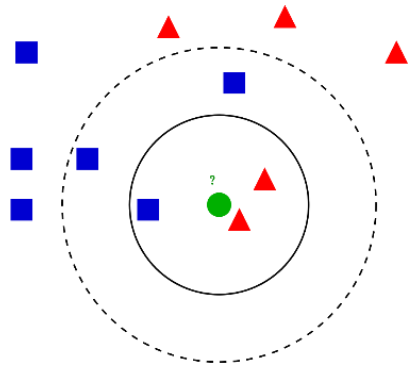
1. Въведение в решавания проблем и цел на проекта

Проектът има за цел да имплементира алгоритъм за размита класификация на данни от измервания на газове, както и сравнението на резултати от класификационен алгоритъм и размит класификационен алгоритъм. По-точно ще бъдат използвани алгоритмите K-Nearest Neighbor и Fuzzy K-Nearest Neighbor.

2. Теоретична постановка и използван алгоритъм

Класификационните алгоритми взимат репрезентация на белезите на обекти/концепти и ги свързва с класификационни етикети. Един такъв класификационен алгоритъм е алгоритъмът K-Nearest Neighbor, който намира k на брой най-близките съседи на обект, който искаме да класифицираме, след което избира най-често срещаният клас като клас на обекта. За разстояние най-често се използва евклидово разстояние.

В примера на Фигура 1. се търси класа на зелената точка, показани са 2 разглеждания на възможни разстояния (двете окръжности с център зелената точка). Спрямо вътрешната окръжност избраният клас е червен триъгълник, спрямо външната – син квадрат.



Фигура 1. Пример за KNN

Разгледаният размит алгоритъм също намира k на брой най-близките съседи на търсения елемент, но при избора на клас се търси принадлежността на елемента към всеки един от класовете. Принадлежността към даден клас се изчислява по следната формула:

$$u_i(x) = \frac{\sum_{j=1}^{K} u_{ij}\left(1/\|x-x_j\|^{2/m-1}\right)}{\sum_{j=1}^{K}\left(1/\|x-x_j\|^{2/m-1}\right)},$$

Където $u_i(x)$ – принадлежността на x към клас i, u_{ij} – принадлежността на j-тия вектор към i-тия клас, $\|x - x_j\|$ – разстоянието от вектора x до x_j.

За имплементиране на споменатите алгоритми и анализ на данните са използвани Python и пакетите NumPy, Pandas и Sklearn.

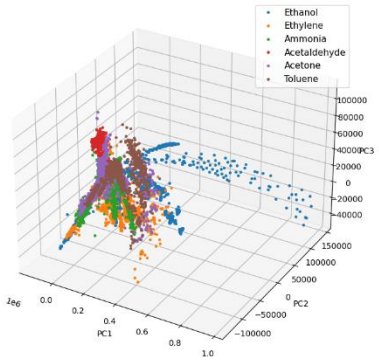
3. Описание на данните

Данните представляват експериментални измервания на газове. Определен газ бива подаден на постановка със сензори, след което измерванията на сензорите се записват. В експеримента са използвани 6 различни газа - етанол, етилен, амоняк, ацеталдехид, ацетон, толуен, използват се 16 сензора, всеки от които допринася с измерването на 8 свойства. От тук получаваме, че едно измерване има 16 * 8 = 128 свойства.

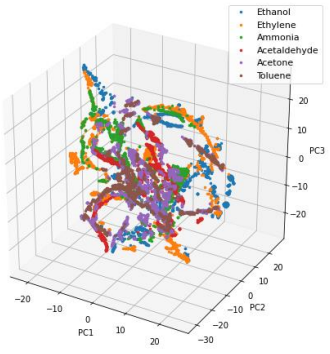
Налични са 10 файла, който сумарно съдържат 13910 реда измервания и 128 колони с дани от сензорите + 1 колона за отбелязване на измерения газ.

В препроцесната обработка, се премахват излишните номерации на измерванията. Направени са и тестове след нормализация на стойностите на векторите.

На Фигури 2 и 3 виждаме пространственото представяне на базата данни чрез използване на алгоритмите PCA и T_SNE. На графиките виждаме, че данните не могат да бъдат клъстеризирани само в 6 клъстера.



Фигура 2. Графика на данните обработени с PCA



Фигура 3. Графика на данните обработени с t_SNE

4. Експериментални резултати

Направени са експерименти с различен брой най-близки съседи (k).

На Фигура 4 виждаме резултатите от K-Nearest Neighbor (crisp). При избор на 50 на брой най-близки съседа има най-добър резултат.

k	50	150	200	250	500	750	1000
accuracy	93%	87.20%	84.20%	81.40%	79.30%	73.20%	71.70%

Фигура 4. Таблица с резултати от crisp класификация при различен брой съседи

На Фигура 5 виждаме резултатите от Fuzzy K-Nearest Neighbor. Параметъра *m* е същия като във формулата. От резултатите виждаме, че при по-голямо *k* и по-малко *m* получаваме най-добрите резултати.

k \ m	1.05	1.1	1.2	1.4	1.6	1.8	2	3	4	5	6	7	8	9
50		60.5%	60.5%	60.9%	60.7%	60.3%	59.0%	57.4%	55.9%	52.0%	48.7%	45.2%	43.3%	40.9%
100							73.0%	69.9%	63.5%	52.7%	44.3%	39.5%	35.4%	32.3%
150							76.8%	59.7%	45.1%	34.1%	28.1%	23.9%	22.5%	22.0%
200							79.1%	65.6%	55.4%	48.3%	43.1%	40.6%	36.8%	33.9%
250			81.6%	82.1%	81.9%	81.2%	80.7%	68.1%	55.2%	46.3%	41.6%	39.9%		
300			84.0%	84.5%	84.4%	84.5%	84.5%	74.8%	56.9%	48.2%	42.1%	39.5%		
350			85.9%	86.1%	85.6%	85.3%	84.0%							
400			87.4%	87.3%	86.7%	86.6%	84.6%							
450			89.4%	89.2%	88.9%	88.6%	86.7%							
500			89.4%	89.2%	89.1%	88.5%	87.4%							
600			90.2%	89.8%	88.9%	88.4%	86.8%							
700			91.7%	91.3%	90.2%	89.6%	88.3%							
800			92.3%	91.8%	90.7%	90.1%	89.3%							
900			92.7%	92.4%	91.4%	90.2%	89.2%							
1000			93.2%	92.7%	92.2%	90.9%								
1100			93.7%	93.2%	92.9%	91.8%								
1200		93.7%	93.5%	93.3%										
1300		94.2%	94.0%	93.6%										
1400		94.5%	94.2%	93.9%										
1500		95.0%	94.8%	94.2%										
2000	96.2%													
3000	97.3%						93.7%							

Фигура 5. Таблица с резултати от размита класификация при различни параметри

След нормализиране на входните данни получаваме резултатите на Фигура 4. Може да забележим, че резултатите са със средно 3 процента по-точни отколкото преди нормализацията.

k	50	100	150	200	250
accuracy	97.30%	93.70%	90.70%	89.20%	87%

Фигура 4. Таблица с "crisp" резултати след нормализация на данните

На Фигура 5 виждаме резултатите от размитата класификация след нормализация на данните. Отново може да забележим, че резултатите спрямо параметрите са с около 3 процента по-точни отколкото преди нормализация, поне за изследваните параметри.

k \ m	1.1	1.2	1.4	1.6	1.8
350	92.3%	92.2%	91.5%	89.1%	85.5%
500	93.8%	94.0%	93.1%		
750	95.6%	95.4%	94.8%		
1000	97.1%	96.9%	96.8%		
1250	98.0%	98.0%	97.7%		
1500	97.9%	97.9%	97.3%		
2000	98.3%	98.3%	98.0%		

Фигура 5. Таблица с резултати от размита класификация след нормализизация на данните

5. Основни изводи

От получените резултати виждаме, че размитият класификатор се представя по-добре отколкото традиционния класификатор. След нормализиране на данните разликата с точността на двата класификатора намалява, но размитият класификатор продължава да дава по-добри резултати.

Възможност за бъдеща работа над темата би бил по-обширен анализ на параметрите преди и след нормализация на входните данни. Друга възможност е сравнение на разгледаните алгоритми с алгоритъм използващ разстоянието на центроидите на идентифицираните клъстери (Nearest prototype classifier, Nearest centroid classifier). Друга възможност представлява анализирането на други функции за претегляне на разстоянията на съседите.

6. Списък на използваната литература

[1] "A Fuzzy K-Nearest Neighbor Algorithm" James M. Keller; Michael R. Gray; James A. Givens

[2] Gas Sensor Array Drift Dataset at Different Concentrations Data Set

[3] "Chemical gas sensor drift compensation using classifier ensembles." A Vergara, S Vembu, T Ayhan, M Ryan, M Homer, R Huerta.

7. Приложение

- GitHub repository
- Код на размития класификатор:

```python
class FuzzyKNNClassifier(KNNClassifier):
    def __init__(self, k=1, m=2, verbose=False):
        super().__init__(k, verbose)
        self.m = m

    def predict_strategy(self, row):
        #find the k nearest neighbours
        distances_to_all = [ (self.distance(row, neighbour), label) for neighbour, label in zip(self.X_train, self.y_train)]
        nearest_k = distances_to_all[:self.k]
        if self.verbose: print(nearest_k[0], nearest_k[-1])

        # compute membership values of row for each class
        result = {}
        denominator = sum([1 / (neighbour[0] ** (2/(self.m-1))) for neighbour in nearest_k])
        if self.verbose: print('denom:\n', denominator)
        for cl in range(1, self.num_classes+1):
            class_membership_sum = sum([1 / (neighbour[0] ** (2/(self.m-1))) for neighbour in nearest_k if neighbour[1] == cl])
            result[cl] = class_membership_sum / denominator
        if self.verbose: print('result:\n', result)

        # defuzzify answer
        defuzzified_result = max(result, key=result.get)
        if self.verbose: print('defuzzyfied:\n', defuzzified_result)

        return defuzzified_result, result
```

Figure 2. Код на размития класификатор