Прилагане на kNN алгоритъма стъпка по стъпка в Python

В темата ще бъдат разгледани следните основни въпроси:

- Кратък преглед на алгоритъма kNN
- Дефиниране на „най-близък“ с помощта на математическа дефиниция на разстоянието
- Намиране на k най-близки съседи
- Средно за регресия
- Особености при прилагане на kNN за класификация

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Прилагане на kNN алгоритъма стъпка по стъпка

В тази част ще бъде разгледан как работи алгоритъмът kNN. Алгоритъмът kNN има два основни математически компонента, които трябва да се разберат.

1. Кратък преглед на алгоритъма kNN

Алгоритъмът kNN е малко нетипичен в сравнение с други алгоритми за машинно обучение. Както видяхте по-рано, всеки модел на машинно обучение има своя специфична формула, която трябва да бъде оценена. Спецификата на алгоритъма k-най-близки съседи е, че тази формула се изчислява не в момента на напасване, а по-скоро в момента на прогнозиране. Това не е така за повечето други модели за машинно обучение.

Когато пристигне нова точка от данни, алгоритъмът kNN, както показва името, ще започне с намирането на най-близките съседи на тази нова точка от данни. След това взема стойностите на тези съседи и ги използва като прогноза за новата точка от данни.

Като интуитивен пример защо това работи, може да се разгледа примера с вашите съседи. Вашите съседи често са относително подобни на вас. Те вероятно са в същата социално-икономическа група като вас. Може би имат същия вид работа като вас, може би техните деца ходят в същото училище като вашите и т.н. Но за някои задачи този вид подход не е толкова полезен. Например, няма да има смисъл да се гледа любимия цвят на съседа, за да предскажете вашия.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Алгоритъмът kNN се основава на идеята, че може да се предвидят характеристиките на точка от данни въз основа на характеристиките на нейните съседи. В някои случаи този метод на прогнозиране може да бъде успешен, докато в други случаи може да не е. След това ще бъде разгледано математическото описание на „най-близките“ точки от данни и методите за комбиниране на множество съседи в една прогноза.

2. Дефиниране на „най-близък“ с помощта на математическа дефиниция на разстоянието

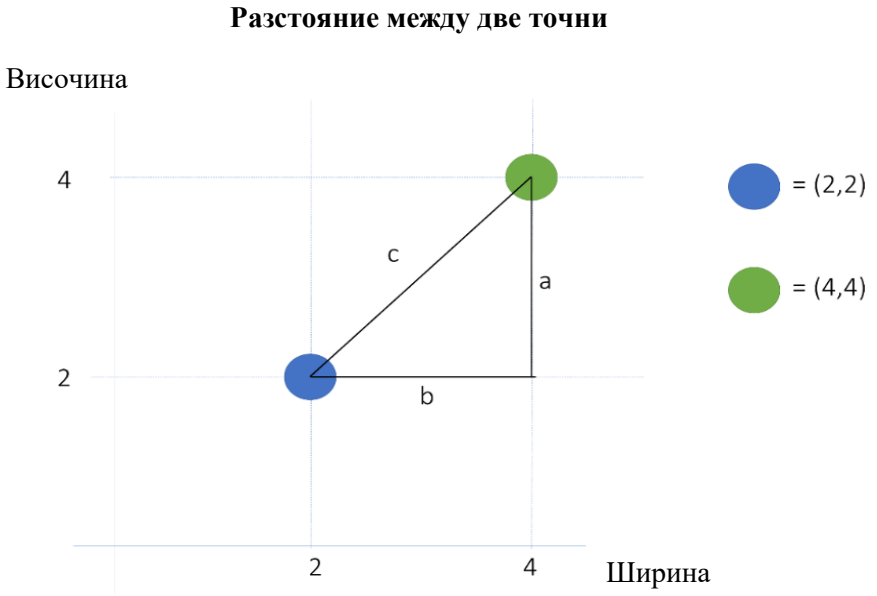
За да се намерят точките от данни, които са най-близо до точката, която трябва да се предвиди, може да се използва математическа дефиниция на разстоянието, наречено Евклидово разстояние.

За да се стигне до определението за Евклидово разстояние, първо трябва да се припомни какво означава разлика между два вектора. Ето един пример (фиг. 10.1).

На фигура 10.1 са изобразени две точки от данни: синя с координати (2,2) и зелена с координати (4,4). За да се изчисли разстоянието между тях, може да се започне с добавяне на два вектора. Вектор **a** преминава от точка (4,2) към точка (4,4), а вектор **b** преминава от точка (4,2) към точка (2,2). Краищата им са обозначени с цветни точки. Имайте предвид, че те са под ъгъл от 90 градуса.

Разликата между тези вектори е векторът **c**, който преминава от края на вектор **a** към края на вектор **b**. Дължината на вектора **c** представлява разстоянието между двете точки от данни.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.



Фиг. 10.1. Разлика между два вектора

Дължината на вектора се нарича **норма**. Нормата е положителна стойност, която показва големината на вектора. Нормата на вектор може да изчисли, като се използва формулата на Евклид:

$$d(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - b_n)^2}. \qquad (10.1)$$

В тази формула разстоянието се изчислява, като се съберат квадратите на разликите във всяко измерение и след това се вземе корен квадратен от сбора на тези стойности. В този случай трябва да се изчисли нормата на вектора на разликата **c**, за да се получи разстоянието между точките от данни.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Сега, за да се приложи тази теория към данните за морското ухо, трябва да се разбере, че точките от данни всъщност са вектори. След това може да се изчисли разстоянието между тях, като се изчисли нормата на вектора на тяхната разлика.

Нормата на разликата на двата вектора от фиг. 10.1 може да изчисли в Python, като се използва linalg.norm() от NumPy. Пример за изчисляване на нормата на разликата на двата вектора, отговарящи на точните от фиг. 10.1, е показан на фиг. 10.2.

```
import numpy as np
a = np.array([2, 2])
b = np.array([4, 4])
print(np.linalg.norm(a - b))
```

а)

```
2.8284271247461903
```

б)

Фиг. 10.2. Изчисляване на нормата на разликата на двата вектора
а. Код на Python **б.** Изход от програмата

В този сорс блок се дефинират двете точки от данни като вектори. След това се изчислява нормата върху разликата между двете точки от данни посредством метода .norm(). По този начин директно се получава разстоянието между две многомерни точки. Въпреки че точките са многомерни, разстоянието между тях е скалар или една стойност.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

3. Намиране на k най-близки съседи

Сега, когато има математически начин за изчисляване на разстоянието от всяка точка до всяка друга точка, това може да се използва, за да се намерят най-близките съседи на точка, за която се иска да се направи прогноза.

Трябва да се намерят k на брой съседи. Минималната стойност на k е 1. Това означава да се използва само един съсед за прогнозата. Максимумът е броят точки с данни, с които се разполага. Това означава да се използват всички съседи. Стойността на k е нещо, което потребителят определя. Инструментите за оптимизация могат да помогнат с по-точното определяне на стойността на k.

Необходимо е отново да се припомни, че множеството от данни Abalone е представено като pandas DataFrame. От фиг. 10.2 се вижда, че за да се намерят най-близките съседи се използва NumPy. Затова, първо трябва данните да се преобразуват от pandas DataFrame в масив NumPy, като се използва атрибута .values (фиг. 10.3).

```
X = abalone.drop("Rings", axis=1)
X = X.values
y = abalone["Rings"]
y = y.values
```

Фиг. 10.3. Преобразуване на данните от pandas DataFrame в масив NumPy

Този сорс код генерира два обекта, които вече съдържат данни, подходящи за kNN алгоритъма: X и y. X е независимата променлива, а y е зависимата променлива на модела. Обикновено се използва главна буква за

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

X, но малка буква за y. Това често се прави в код за машинно обучение, тъй като математическата нотация обикновено използва главна буква за матрици и малка буква за вектори.

Сега вече може да се приложи kNN с k = 3 върху ново морско ухо, което има следните физически измервания и не е в базата данни (таблица 10.1).

Таблица 10.1. Физически характеристики на ново морско ухо

Променлива	**Стойност**
Length	0.569552
Diameter	0.446407
Height	0.154437
Whole weight	1.016849
Shucked weight	0.439051
Viscera weight	0.222526
Shell weight	0.291208

Създава се NumPy масив за тази точка от данни, както следва (фиг. 10.4).

```
new_data_point = np.array([0.569552, 0.446407, 0.154437,
1.016849, 0.439051, 0.222526, 0.291208])
```

Фиг. 10.4. Създаване на NumPy масив с данни от таблица 10.1

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Следващата стъпка е да се изчислят разстоянията между тази нова точка от данни и всяка от точките от данни в множеството от данни Abalone, като се използва следният код (фиг. 10.5)

```
distances = np.linalg.norm(X - new_data_point, axis=1)
```

Фиг. 10.5. Изчисляване на разстоянията между новата точка от данни и всяка друга от данните Abalone

Кодът от фиг. 10.5 получава вектор от разстоянията и е необходимо да се намерят кои са трите най-близки съседни точки. За да се направи това, трябва да се намерят идентификаторите на минималните разстояния. Може да се използва метод, наречен .argsort(), за да се сортира масива от най-ниската до най-високата стойност и да се вземат първите k елемента, за да се получат индексите на k най-близки съседи (фиг. 10.6).

```
k = 3
nearest_neighbor_ids = distances.argsort()[:k]
print(nearest_neighbor_ids)
```

а)

```
[4045, 1902, 1644]
```

б)

Фиг. 10.6. Получаване на индексите на k най-близки съседи
а. Код на Python **б.** Изход от програмата

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Кодът показва индексите на кои трима съседи са най-близо до новата точка new_data_point. В следващия параграф ще се конвертират тези съседи в оценка.

4. Прогнозиране или усредняване на множество съседи

След като са идентифицирани индексите на трите най-близки съседи на новото морско ухо с неизвестна възраст, сега трябва да се комбинират тези съседи в прогноза за новата точка от данни.

Като първа стъпка трябва да се намерят основните зависими променливи **y** (брой пръстени Rings) за тези трима съседи (фиг. 10.15).

```
nearest_neighbor_rings = y[nearest_neighbor_ids]
print(nearest_neighbor_rings)
```

а)

```
[9, 11, 10]
```

б)

Фиг. 10.6. Получаване на индексите на k най-близки съседи
а. Код на Python **б.** Изход от програмата

Сега, когато се разполага със стойностите за тези три съседни точки, те ще се комбинират в прогноза за новата точка от данни. Комбинирането на съседите в прогноза работи по различен начин за задачи за регресия и класификация.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

4.1. Средно за регресия

При проблеми с регресия целевата променлива е числова. Вие Множеството съседи се комбинират в една прогноза, като се вземе средната стойност от техните стойности за целевата променлива. Това може да се направи по следния начин (фиг. 10 16).

```
prediction = nearest_neighbor_rings.mean()
print(prediction)
```

а)

```
10.0
```

б)

Фиг. 10.6. Получаване на средната стойност при регресия

а. Код на Python **б.** Изход от програмата

Получава се стойност 10 за прогнозата. Това означава, че прогнозата за трите най-близки съседи за новата точка от данни е 10. Тази процедура за прогноза може да се повтори за произволен брой нови охлюви.

4.2. Особености при прилагане на kNN за класификация

При проблеми с класификацията целевата променлива е категорична. Както беше обсъдено по-рано, не могат да се вземат средни стойности за категорични променливи. Например, каква би била средната стойност от три прогнозирани марки автомобили? Това би било невъзможно да се каже. Не може да се приложи средна стойност към прогнозите за класа.

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.

Вместо това, в случай на класификация, се приема така наречениият „режим“. Режимът е стойността, която се среща най-често. Това означава, че трябва да се преброят класовете на всички съседи и да се запази най-често срещания клас. При класификация прогнозата е стойността, която се среща най-често сред съседите.

Ако има няколко режима, има множество възможни решения. Може да се избере краен победител на случаен принцип от режимите. Може също така да се вземе окончателното решение въз основа на разстоянията до съседите, като в този случай ще се запази режимът на най-близките съседи.

Режимът може да се изчисли с помощта на функцията .mode() на SciPy. Тъй като примерът с морско ухо не е подходящ на класификация, следващият код показва как може да се изчисли режима за един прост пример (фиг. 10.7).

```
import scipy.stats
import numpy as np
class_neighbors = np.array(["A", "B", "B", "C"])
print(scipy.stats.mode(class_neighbors))
```

а)

```
ModeResult(mode=array(['B'], dtype='<U1'), count=array([2]))
```

б)

Фиг. 10.7. Особености при прилагане на kNN за класификация

а. Код на Python **б.** Изход от програмата

Както се вижда, режимът mode в този пример е “B”, защото това е стойността, която се среща най-често във входните данни (два пъти).

Проект BG05M2OP001-2.016-0003„Модернизация на Национален военен университет "В. Левски" - гр. Велико Търново и Софийски университет "Св. Климент Охридски" - гр. София, в професионално направление 5.3 Компютърна и комуникационна техника“, финансиран от Оперативна програма „Наука и образование за интелигентен растеж“, съфинансирана от Европейския съюз чрез Европейските структурни и инвестиционни фондове.