ED – Laboratorium 2

k-NN

1 k-NN

Algorytm **k najbliższych sąsiadów**, w literaturze określany zazwyczaj jako **k-NN** (od ang. *k Nearest Neighbours*), to pierwszy bardziej skomplikowany algorytm uczenia z którym się zapoznamy. Jego główna idea to klasyfikacja poprzez analogię/podobieństwo do wcześniej poznanych przypadków. Wyobraź sobie, że jesteś lekarzem i chcesz wykorzystać uczenie maszynowe by proponowało Tobie wstępną diagnozę pacjentów. Masz do dyspozycji informacje o objawach wszystkich poprzednich pacjentów (np. rodzaj kaszlu, kataru, itd.) i ostatecznych diagnozach (np. grypa, astma). k-NN dla nowego pacjenta wyszuka dokładnie k najbardziej podobnych jeżeli chodzi o objawy „historycznych" pacjentów, a następnie sprawdzi, która choroba występowała wśród nich najczęściej – będzie ona ostateczną decyzją, jaką zwróci ten klasyfikator.

1.1 Miara odległości

W algorytmie k-NN kluczowe jest zdefiniowanie miary odległości między przypadkami. Posłuży ona do wyboru najbliższych sąsiadów. Dla atrybutów liczbowych stosuje się często **odległość euklidesową**. Dla dwóch przypadków A i B opisanych na n atrybutach odległość ta opisana jest wzorem:

$$d(A,B) = \sqrt{(A_1 - B_1)^2 + (A_2 - B_2)^2 + \cdots + (A_n - B_n)^2},$$

gdzie X_i oznacza wartość i-tego atrybutu dla przypadku X.

Alternatywnie, wykorzystać można **odległość taksówkową** (nazywaną również *odległością Manhattan*, np. w WEKA), która sumuje tylko bezwzględne różnice między poszczególnymi wartościami:

$$d(A,B) = |A_1 - B_1| + |A_2 - B_2| + \cdots + |A_n - B_n|$$

Przykład 1.1 — Liczenie odległości euklidesowej. Załóżmy, że mamy następujące przypadki:

#Example	Atr1	Atr2	Atr3	Decision
A	0	-2	6	NO
B	3	2	5	NO

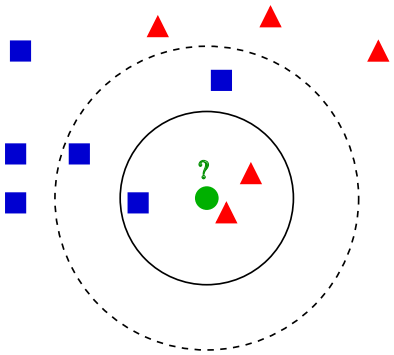
Odległość euklidesową między A i B policzymy jako:

$$d(A,B) = \sqrt{(0-3)^2 + (-2-2)^2 + (6-5)^2} = \sqrt{(-3)^2 + (-4)^2 + 1^2} = \sqrt{9+16+1} = \sqrt{26} \approx 5.1$$

■

Przykład 1.2 — Liczenie odległości taksówkowej. Dla danych z Przykładu 1.1 odległość taksówkową policzymy jako:

$$d(A,B) = |0-3| + |-2-2| + |6-5| = |-3| + |-4| + |1| = 3+4+1 = 8$$



Rysunek 1: Wizualizacja działania k-NN na płaszczyźnie dla różnych wartości k (3 i 5). Źródło: `https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm`.

1.2 Zarys działania

W algorytmie k-NN wszystkie przypadki uczące są po prostu zapamiętywane. To wszystko, co musi zostać zrobione w fazie uczenia.

Proces klasyfikacji nowego przypadku X przebiega następująco:

1. Oblicz odległość między X a każdym zapamiętanym przypadkiem uczącym.
2. Posortuj odległości rosnąco i weź pierwszych k pozycji (k jest parametrem algorytmu podawanym na starcie przez użytkownika). Na pozycjach tych znajdować się będą najbliżsi sąsiedzi.
3. Wśród wybranych przypadków sprawdź przydzielone decyzje i wybierz tą, która występuje najczęściej. Jeżeli jest remis, to wybierz dowolną.

Wybrana większościowa decyzja w ostatnim punkcie stanowi ostateczny wynik zwrócony przez klasyfikator.

1.3 Pozostałe uwagi

- Parametr k zazwyczaj ustalany jest jako liczba nieparzysta, ponieważ dzięki temu maleje liczba remisów, które trzeba rozstrzygać.

1.4 Przykład 1-wymiarowy

Przedstawimy teraz k-NN na najprostszym możliwym przykładzie danych opisanych tylko jednym atrybutem. Taka sytuacja raczej nie wystąpi w praktyce, ale pozwala na bardzo czystą prezentację ogólnego działania algorytmu. Tabelka z danymi wyglądać będzie następująco:

#Example	Atr1	Decision
A	1	YES
B	2	NO
C	4	YES
D	6	NO
E	8	NO
F	12	NO

Załóżmy, że dostajemy do zaklasyfikowania następujący przypadek: $(X, 3)$. Wyliczymy teraz przy pomocy odległości taksówkowej odległości między X a przykładami uczącymi. Posortujemy je też od razu rosnąco ze względu na odległość.

#	Przykład	Odległość od X	Decyzja
1.	B	$\|3-2\|=1$	NO
1.	C	$\|3-4\|=1$	YES
3.	A	$\|3-1\|=2$	YES
4.	D	$\|3-6\|=3$	NO
5.	E	$\|3-8\|=5$	NO
6.	F	$\|3-12\|=9$	NO

Widzimy więc, że wynik klasyfikacji zależy mocno od wybranej wartości k.

- Dla $k=1$ mamy remis między B i C (mają tę samą odległość od X) i musimy arbitralnie któryś wybrać, np. losowo[1]. W takim wypadku k-NN w 50% przypadków zwróciłby YES a w 50% przypadków NO.
- Dla $k=3$ weźmiemy dla X elementy B, C i A. Widzimy, że YES występuje wśród nich częściej, tak więc taką właśnie odpowiedź zwróci klasyfikator.
- Dla $k=5$ weźmiemy dla X elementy B, C, A, D, E. Częściej występuje NO.
- Dla $k>5$ będzie zawsze więcej decyzji NO.

Dla większej liczby wymiarów (atrybutów) w naszych danych jedyną różnicą w powyższej procedurze byłoby wyliczanie odległości od X.

[1] Równie dobrze można by np. próbować rozstrzygnąć taki konflikt patrząc na następnego w kolejności sąsiada. Zachowanie w takich wypadkach nie jest formalnie zdefiniowane i leży w gestii programisty.