

Asocjacyjna reprezentacja danych i wnioskowanie

Wykorzystane technologie

- ▶ JetBrains PyCharm 5.0.4
- ▶ Python 3.5



Struktura drzewa

GRAPH

PARAM

ID1

params

param_name_1: param_value_1
param_name_2: param_value_2
⋮
param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

ID2

params

param_name_1: param_value_1
param_name_2: param_value_2
⋮
param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

...

ID_N

params

param_name_1: param_value_1
param_name_2: param_value_2
⋮
param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

Struktura drzewa

GRAPH

PARAM

ID1

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

ID2

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

...

ID_N

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

Struktura drzewa

PARAM

ATTRIBUTE_NAMES

NAME_1

ATTRIBUTE_VALUES

VAL_1 OBJECT_IDS

VAL_2 OBJECT_IDS

⋮

VAL_N OBJECT_IDS

NAME_2

ATTRIBUTE_VALUES

VAL_1 OBJECT_IDS

VAL_2 OBJECT_IDS

⋮

VAL_N OBJECT_IDS

...

NAME_N

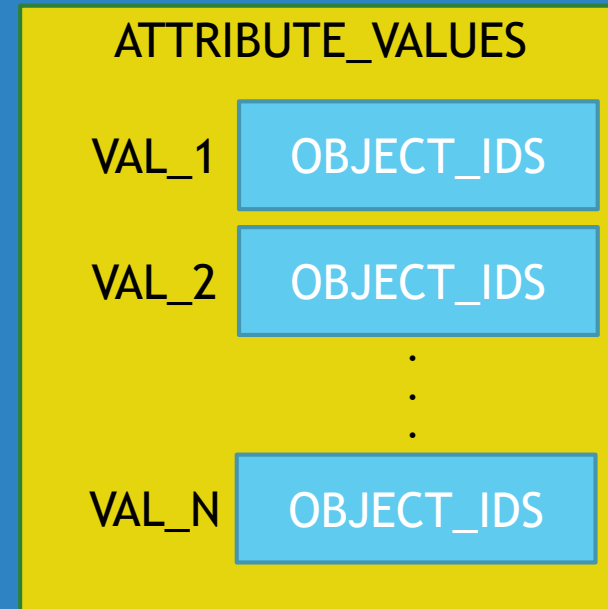
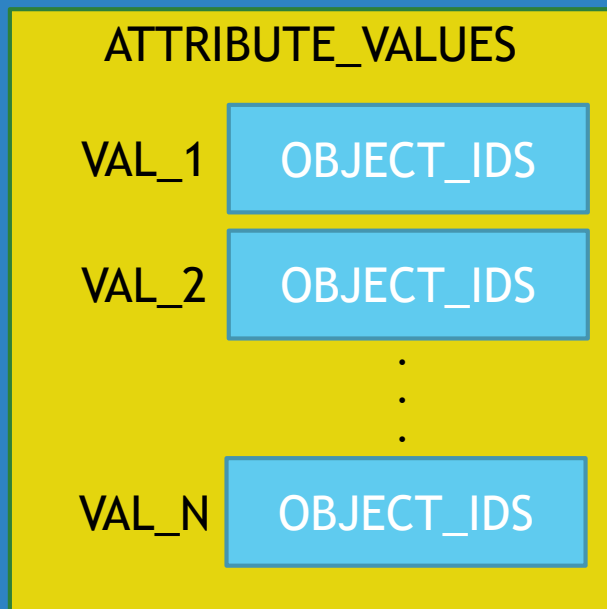
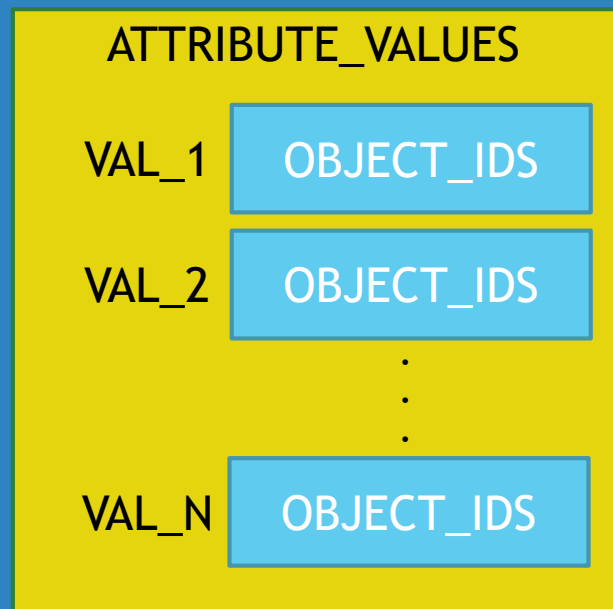
ATTRIBUTE_VALUES

VAL_1 OBJECT_IDS

VAL_2 OBJECT_IDS

⋮

VAL_N OBJECT_IDS



Struktura drzewa

GRAPH

PARAM

ID1

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

ID2

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

...

ID_N

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

Struktura drzewa

GRAPH

PARAM

ID1

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

ID2

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

...

ID_N

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

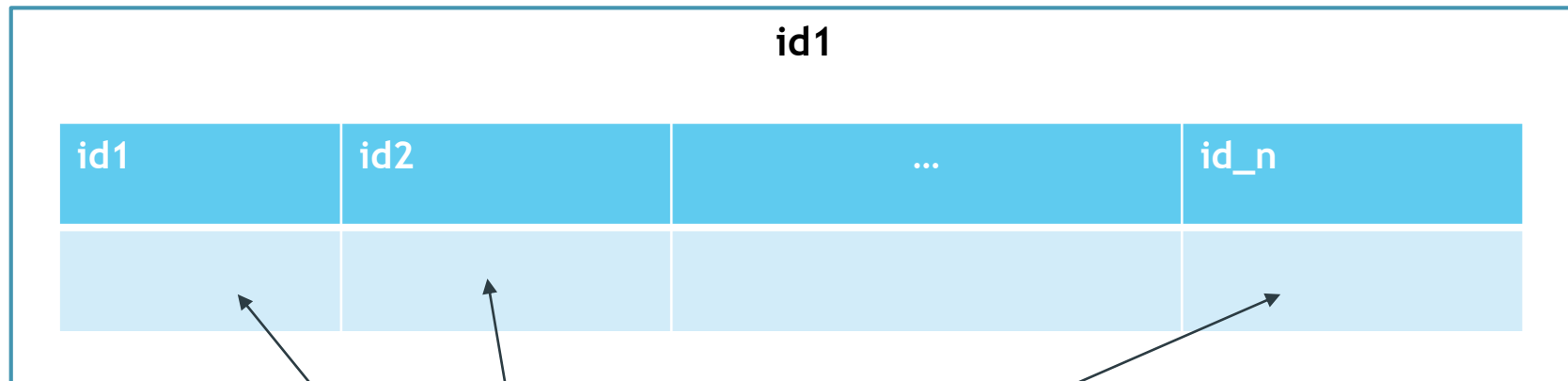
strong_similarities

id1	id2	...	Id_n

Similarities - Wagi podobieństw

Każdy obiekt ma tablicę podobieństw o rozmiarze równym ilości obiektów w bazie

Każdy element tablicy odpowiada wartości podobieństwa danego obiektu i obiektu o id = indeksowi tablicy



$$w_{a,b} = \sum_{i=0}^n \left(1 - \frac{|a - b|}{MAX_{P_i} - MIN_{P_i}} \right)$$

Struktura drzewa

GRAPH

PARAM

ID1

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

ID2

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

...

ID_N

params

param_name_1: param_value_1
param_name_2: param_value_2

⋮

param_name_n: param_value_n

similarities

id1	id2	...	Id_n

strong_similarities

id1	id2	...	Id_n

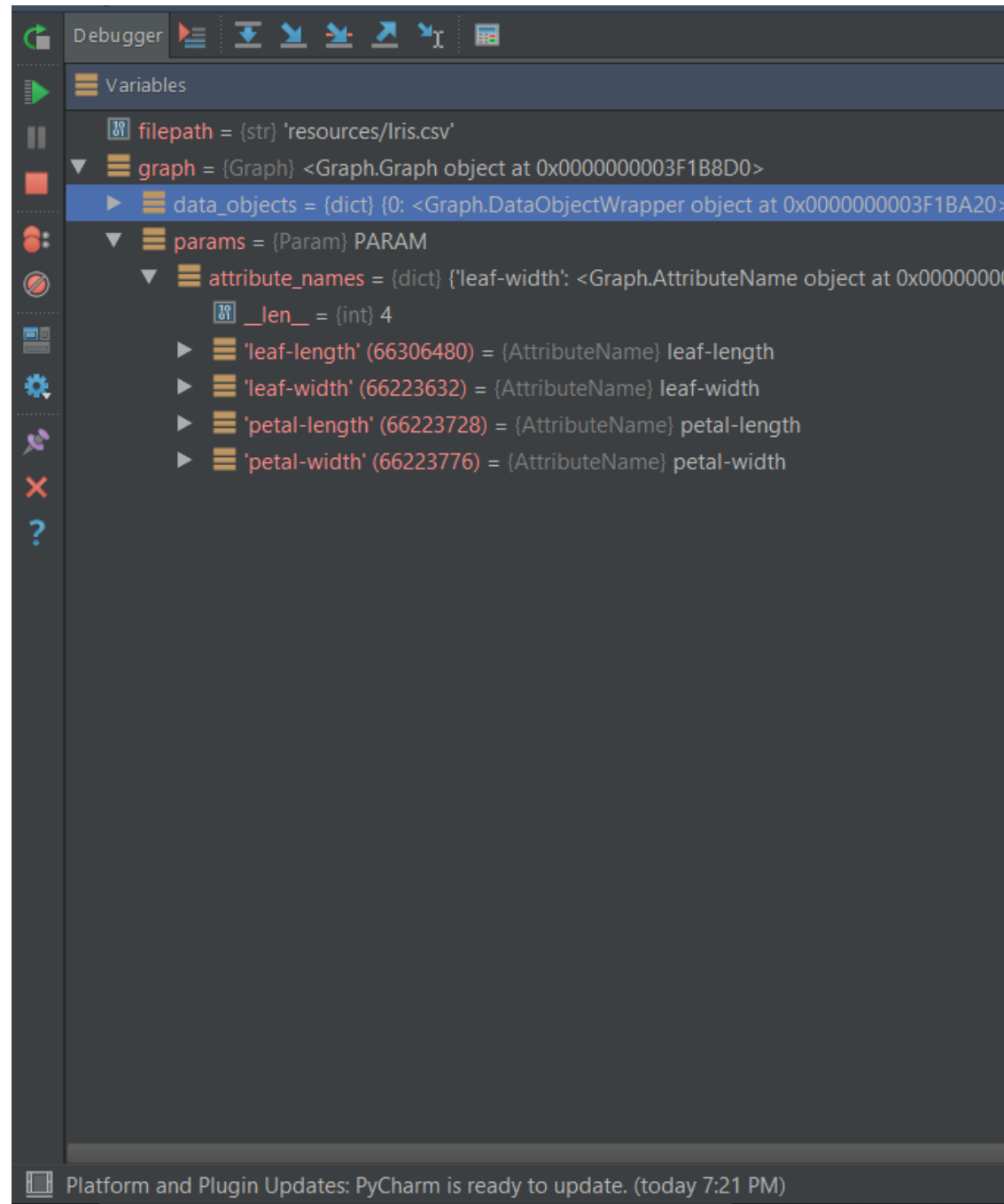
Strong similarities - Wagi silnych podobieństw

Wartość podobieństwa inkrementowana jest o 1, w przypadku gdy oba badane obiekty mają tą samą wartość danego parametru

Wartości w tabeli mogą być tylko całkowite z zakresu (0,n), gdzie n - ilość parametrów

id1			
id1	Id2	...	id_n
4 (4 takie same parametry)	3 (3 takie same parametry)		1 (1 taki sam parametr)

Realizacja



The image shows a screenshot of a Python debugger interface. The top bar includes a 'Debugger' label and several icons for navigation and actions. Below this is a 'Variables' panel with a tree view of the current state:

- `filepath` = {str} 'resources/Iris.csv'
- `graph` = {Graph} <Graph.Graph object at 0x0000000003F1B8D0>
 - `data_objects` = {dict} {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>}
 - `params` = {Param} PARAM
 - `attribute_names` = {dict} {'leaf-width': <Graph.AttributeName object at 0x0000000003F1B8D0>}
 - `__len__` = {int} 4
 - `'leaf-length'` (66306480) = {AttributeName} leaf-length
 - `'leaf-width'` (66223632) = {AttributeName} leaf-width
 - `'petal-length'` (66223728) = {AttributeName} petal-length
 - `'petal-width'` (66223776) = {AttributeName} petal-width

At the bottom of the window, a notification reads: 'Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)'

Realizacja

The screenshot shows a Python debugger window titled "Debug graphs". The "Variables" pane displays the following structure:

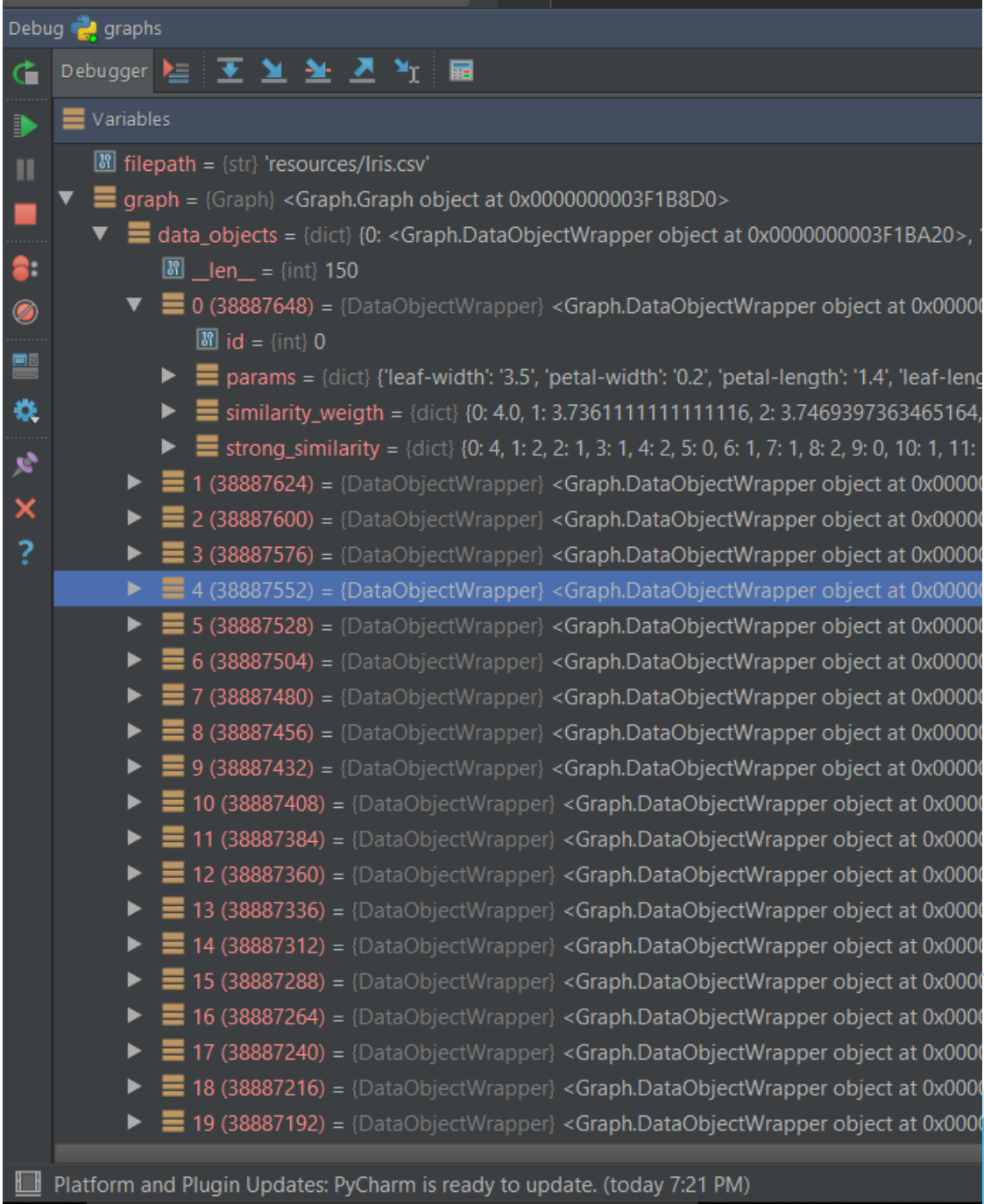
- `filepath` = (str) 'resources/Iris.csv'
- `graph` = (Graph) <Graph.Graph object at 0x0000000003F1B8D0>
 - `data_objects` = (dict) {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, ...}
 - `params` = (Param) PARAM
 - `attribute_names` = (dict) {'leaf-width': <Graph.AttributeName object at 0x0000000003F1B8D0>, ...}
 - `__len__` = (int) 4
 - `'leaf-length' (66306480)` = (AttributeName) leaf-length
 - `attribute_name` = (str) 'leaf-length'
 - `attribute_values` = (dict) {'7.4': <Graph.AttributeValue object at 0x0000000003F1B8D0>, ...}
 - `__len__` = (int) 35
 - '4.3' (62231536) = (AttributeValue) 4.3
 - '4.4' (62230576) = (AttributeValue) 4.4
 - '4.5' (61695600) = (AttributeValue) 4.5
 - '4.6' (61840720) = (AttributeValue) 4.6
 - '4.7' (61841240) = (AttributeValue) 4.7
 - '4.8' (62233776) = (AttributeValue) 4.8
 - '4.9' (61841120) = (AttributeValue) 4.9
 - '5' (40807520) = (AttributeValue) 5
 - '5.1' (61840560) = (AttributeValue) 5.1
 - '5.2' (61695360) = (AttributeValue) 5.2
 - '5.3' (61695400) = (AttributeValue) 5.3
 - '5.4' (62230976) = (AttributeValue) 5.4
 - '5.5' (61695880) = (AttributeValue) 5.5
 - '5.6' (61884416) = (AttributeValue) 5.6
 - '5.7' (62234416) = (AttributeValue) 5.7
 - '5.8' (62233376) = (AttributeValue) 5.8
 - '5.9' (61884336) = (AttributeValue) 5.9
 - '6' (39179128) = (AttributeValue) 6

Realizacja

The screenshot displays a Python debugger window titled "Debug graphs". The "Variables" pane shows the following structure:

- `filepath` = (str) 'resources/Iris.csv'
- `graph` = (Graph) <Graph.Graph object at 0x0000000003F1B8D0>
 - `data_objects` = (dict) {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, ...}
 - `params` = (Param) PARAM
 - `attribute_names` = (dict) {'leaf-width': <Graph.AttributeName object at 0x0000000003F1B8D0>, ...}
 - `__len__` = (int) 4
 - `'leaf-length'` (66306480) = (AttributeName) leaf-length
 - `attribute_name` = (str) 'leaf-length'
 - `attribute_values` = (dict) {'7.4': <Graph.AttributeValue object at 0x0000000003F1B8D0>, ...}
 - `__len__` = (int) 35
 - '4.3' (62231536) = (AttributeValue) 4.3
 - '4.4' (62230576) = (AttributeValue) 4.4
 - '4.5' (61695600) = (AttributeValue) 4.5
 - '4.6' (61840720) = (AttributeValue) 4.6
 - '4.7' (61841240) = (AttributeValue) 4.7
 - '4.8' (62233776) = (AttributeValue) 4.8
 - `attribute_itemList` = (list) [11, 12, 24, 30, 45]
 - `__len__` = (int) 5
 - 0 = (int) 11
 - 1 = (int) 12
 - 2 = (int) 24
 - 3 = (int) 30
 - 4 = (int) 45
 - `attribute_value` = (str) '4.8'
 - '4.9' (61841120) = (AttributeValue) 4.9
 - '5' (40807520) = (AttributeValue) 5
 - '5.1' (61840560) = (AttributeValue) 5.1
 - '5.2' (61695360) = (AttributeValue) 5.2

Realizacja



Realizacja

The screenshot shows the PyCharm debugger interface. The top bar indicates the current session is for a file named 'graphs'. Below the toolbar, the 'Variables' window is open, displaying the state of a Python object. The object is a 'Graph' object, specifically a '<Graph.Graph object at 0x0000000003F1B8D0>'. It contains a 'data_objects' dictionary with 15 entries. Each entry is a 'DataObjectWrapper' object. The selected entry (index 4) has an 'id' of 4 and a 'params' dictionary with values: 'leaf-width': '3.6', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-length': '4.7'. The 'similarity_weight' and 'strong_similarity' attributes are also visible for this entry. The bottom status bar shows a notification: 'Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)'.

```
Debug graphs
Debugger
Variables
filepath = (str) 'resources/Iris.csv'
graph = (Graph) <Graph.Graph object at 0x0000000003F1B8D0>
  data_objects = (dict) {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 1: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 2: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 3: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 4: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 5: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 6: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 7: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 8: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 9: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 10: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 11: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 12: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 13: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 14: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, 15: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>}
  __len__ = (int) 150
  0 (38887648) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
    id = (int) 0
    params = (dict) {'leaf-width': '3.5', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-length': '4.7'}
    similarity_weight = (dict) {0: 4.0, 1: 3.7361111111111116, 2: 3.7469397363465164, 3: 3.7577793726190218, 4: 3.768618009291527, 5: 3.7794566459640324, 6: 3.7902952826365378, 7: 3.801133919309043, 8: 3.8119725559815484, 9: 3.8228111926540538, 10: 3.8336498293265592, 11: 3.8444884660000646, 12: 3.85532710267257, 13: 3.8661657393450754, 14: 3.8770043760175808, 15: 3.8878430126900862}
    strong_similarity = (dict) {0: 4, 1: 2, 2: 1, 3: 1, 4: 2, 5: 0, 6: 1, 7: 1, 8: 2, 9: 0, 10: 1, 11: 1, 12: 2, 13: 1, 14: 2, 15: 1}
  1 (38887624) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  2 (38887600) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  3 (38887576) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  4 (38887552) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
    id = (int) 4
    params = (dict) {'leaf-width': '3.6', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-length': '4.7'}
    similarity_weight = (dict) {0: 3.9305555555555554, 1: 3.7222222222222223, 2: 3.7330608593750003, 3: 3.7439000000000002, 4: 3.7547391456250001, 5: 3.7655782862500001, 6: 3.7764174268750001, 7: 3.7872565725000001, 8: 3.7980957131250001, 9: 3.8089348587500001, 10: 3.8197740043750001, 11: 3.8306131500000001, 12: 3.8414522956250001, 13: 3.8522914412500001, 14: 3.8631305868750001, 15: 3.8739697325000001}
    strong_similarity = (dict) {0: 2, 1: 2, 2: 1, 3: 1, 4: 4, 5: 0, 6: 1, 7: 2, 8: 2, 9: 0, 10: 1, 11: 1, 12: 2, 13: 1, 14: 2, 15: 1}
  5 (38887528) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  6 (38887504) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  7 (38887480) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  8 (38887456) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  9 (38887432) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  10 (38887408) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  11 (38887384) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  12 (38887360) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  13 (38887336) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  14 (38887312) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  15 (38887288) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)
```

Realizacja

The screenshot shows a Python debugger window titled "Debug graphs". The "Variables" pane displays the following structure:

- `filepath` = (str) 'resources/Iris.csv'
- `graph` = {Graph} <Graph.Graph object at 0x0000000003F1B8D0>
 - `data_objects` = {dict} {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, ...}
 - `__len__` = (int) 150
 - 0 (38887648) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000...>
 - `id` = (int) 0
 - `params` = {dict} {'leaf-width': '3.5', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-len...
 - `similarity_weigth` = {dict} {0: 4.0, 1: 3.7361111111111116, 2: 3.7469397363465164...
 - `strong_similarity` = {dict} {0: 4, 1: 2, 2: 1, 3: 1, 4: 2, 5: 0, 6: 1, 7: 1, 8: 2, 9: 0, 10: 1, 11: ...}
 - 1 (38887624) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000...>
 - 2 (38887600) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000...>
 - 3 (38887576) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000...>
 - 4 (38887552) = {DataObjectWrapper} <Graph.DataObjectWrapper object at 0x0000...> (highlighted)
 - `id` = (int) 4
 - `params` = {dict} {'leaf-width': '3.6', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-len...}
 - `__len__` = (int) 4
 - 'leaf-length' (66306480) = (str) '5'
 - 'leaf-width' (66223632) = (str) '3.6'
 - 'petal-length' (66223728) = (str) '1.4'
 - 'petal-width' (66223776) = (str) '0.2'
 - `similarity_weigth` = {dict} {0: 3.9305555555555554, 1: 3.7222222222222223, 2: 3.7...
 - `__len__` = (int) 150
 - 0 (38887648) = (float) 3.9305555555555556
 - 1 (38887624) = (float) 3.7222222222222222
 - 2 (38887600) = (float) 3.73305084746
 - 3 (38887576) = (float) 3.66360640301
 - 4 (38887552) = (float) 4.0
 - 5 (38887528) = (float) 3.62970809793

Realizacja

Debug graphs

Debugger

Variables

```
filepath = (str) 'resources/Iris.csv'
graph = (Graph) <Graph.Graph object at 0x0000000003F1B8D0>
data_objects = (dict) {0: <Graph.DataObjectWrapper object at 0x0000000003F1BA20>, ...}
  __len__ = (int) 150
  0 (38887648) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
    id = (int) 0
    params = (dict) {'leaf-width': '3.5', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-length': '5'}
    similarity_weigth = (dict) {0: 4.0, 1: 3.7361111111111116, 2: 3.7469397363465164, ...}
    strong_similarity = (dict) {0: 4, 1: 2, 2: 1, 3: 1, 4: 2, 5: 0, 6: 1, 7: 1, 8: 2, 9: 0, 10: 1, 11: ...}
  1 (38887624) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  2 (38887600) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  3 (38887576) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
  4 (38887552) = (DataObjectWrapper) <Graph.DataObjectWrapper object at 0x0000000003F1BA20>
    id = (int) 4
    params = (dict) {'leaf-width': '3.6', 'petal-width': '0.2', 'petal-length': '1.4', 'leaf-length': '5'}
      __len__ = (int) 4
      'leaf-length' (66306480) = (str) '5'
      'leaf-width' (66223632) = (str) '3.6'
      'petal-length' (66223728) = (str) '1.4'
      'petal-width' (66223776) = (str) '0.2'
    similarity_weigth = (dict) {0: 3.9305555555555554, 1: 3.7222222222222223, 2: 3.7361111111111116, ...}
    strong_similarity = (dict) {0: 2, 1: 2, 2: 1, 3: 1, 4: 4, 5: 0, 6: 1, 7: 2, 8: 2, 9: 0, 10: 1, 11: ...}
      __len__ = (int) 150
      0 (38887648) = (int) 2
      1 (38887624) = (int) 2
      2 (38887600) = (int) 1
      3 (38887576) = (int) 1
      4 (38887552) = (int) 4
```

Platform and Plugin Updates: PyCharm is ready to update. (today 7:21 PM)

Funkcjonalności

- ▶ Dodanie nowego obiektu do grafu
- ▶ Szukanie obiektów o zadanej wartości danego atrybutu
- ▶ Filtracja obiektów po wartościach zadanych atrybutów
- ▶ Znajdywanie obiektu po id
- ▶ Znajdywanie najbardziej podobnych n obiektów do obiektu o zadanym id
- ▶ Znajdywanie najbardziej podobnych n obiektów do obiektu o zadanych atrybutach
- ▶ Znajdywanie najmniej podobnych n obiektów do obiektu o zadanym id
- ▶ Znajdywanie najmniej podobnych n obiektów do obiektu o zadanych atrybutach
- ▶ Znajdywanie obiektu o maksymalnej wartości zadanego atrybutu
- ▶ Znajdywanie obiektów o wartościach atrybutu z podanego zakresu

Porównanie czasowe

Funkcja	Bez grafu	Z zastowaniem grafu
Obiekty o zadanym parametrze	0.00169351051901	2.99357920027e-06
Obiekty o zadanych parametrach	0.000679542478461	1.19743168011e-05
10 obiektów najbardziej podobnych do obiektu o zadanym id	0.125938620987493	0.00565144987594
10 obiektów najbardziej podobnych do obiektu o zadanych parametrach	0.634647574562377	0.0151462277908
Obiekt o największej wartości zadanego parametru	0.009387928783985	2.99357920027e-06
Obiekty o parametrach w zadanym przedziale	0.019794387329487	3.46399878888e-05

Podsumowanie

Zastosowanie struktur AGDS znacznie przyspieszyło pracę algorytmu. Dla niektórych funkcjonalności możemy zaobserwować nawet, że algorytm działa około 1000 razy szybciej!

Ponadto zastosowanie grafu pozwala na przejrzyste przedstawienie danych i ich łatwe przeszukiwanie.

Eksploracja Danych

Eksploracja danych

- ▶ Wprowadzenie zbioru danych
- ▶ Obliczenie wsparcia dla danych - częstotliwości wystąpień wzorca lub zbioru elementów

```
ZADANIE 1  
{'kawa': 0.4444444444444444, 'ser': 0.3333333333333333, 'maslo': 0.3333333333333333, 'cukier': 0.4444444444444444, 'jajka': 0.4444444444444444,
```

- ▶ Określenie progu wsparcia, na którego podstawie usunięte są zbiory rzadkie

```
{'kawa': 0.4444444444444444, 'ser': 0.3333333333333333, 'maslo': 0.3333333333333333, 'cukier': 0.4444444444444444, 'jajka': 0.4444444444444444,
```

- ▶ Określenie reguł asocjacyjnych dla zbiorów częstych

```
(('chleb', 'maslo'): (0.4444444444444444, 0.6666666666666666), ('chleb', 'miod'): (0.5555555555555556, 0), ('orzeszki', 'jajka'): (0.6666666666666667, 0.3333333333333333),
```

- ▶ Ekwiwalentna transformacja klas

```
{'kawa': [1, 3, 2, 6], 'ser': [3, 8, 9], 'maslo': [5, 8, 3], 'cukier': [2, 1, 4, 3], 'jajka': [4, 5, 2, 8], 'miod': [4, 7], 'platki': [4, 7],
```

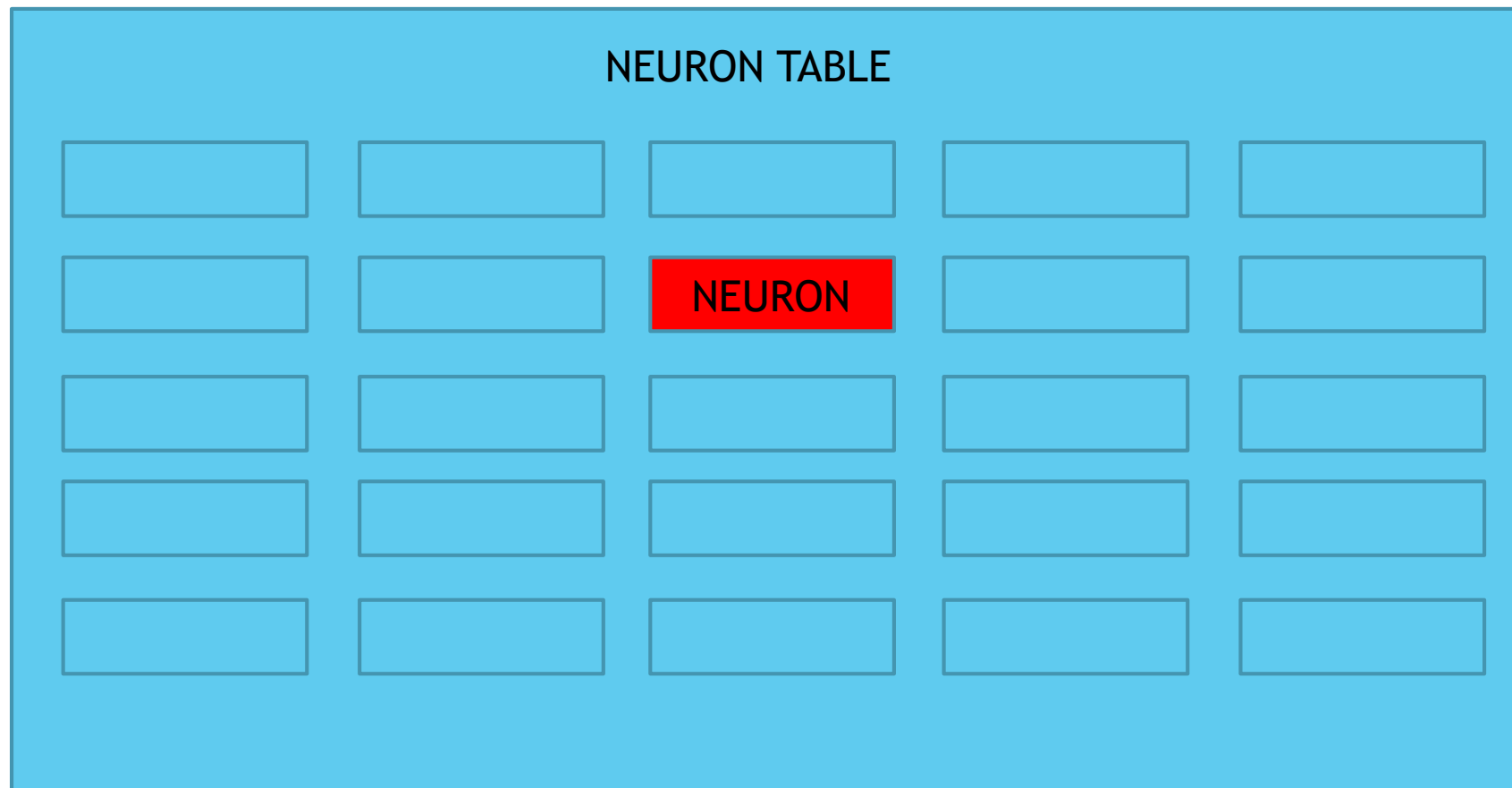
SOM-MAPY SAMOORGANIZUJĄCE SIĘ

SOM to przykład sieci neuronowych uczony metodą uczenia nienadzorowanego, czyli bez określenia np. klasy, do której należą wzorce wejściowe. Sieć SOM na podstawie podobieństwa wzorców wejściowych aktualizuje wagi neuronów na zasadach rywalizacji w taki sposób, iż neuron najbardziej pobudzony ze względu na największe podobieństwo do wzorca uczącego staje się zwycięzcą, którego wagi zostają najmocniej zmodyfikowane w porównaniu z sąsiednimi neuronami, dla których przyjmujemy zwykle mniejsze modyfikacje wag. Dzięki modyfikacjom wag sąsiednich neuronów uzyskujemy możliwość sąsiedniej reprezentacji podobnych klas wzorców.

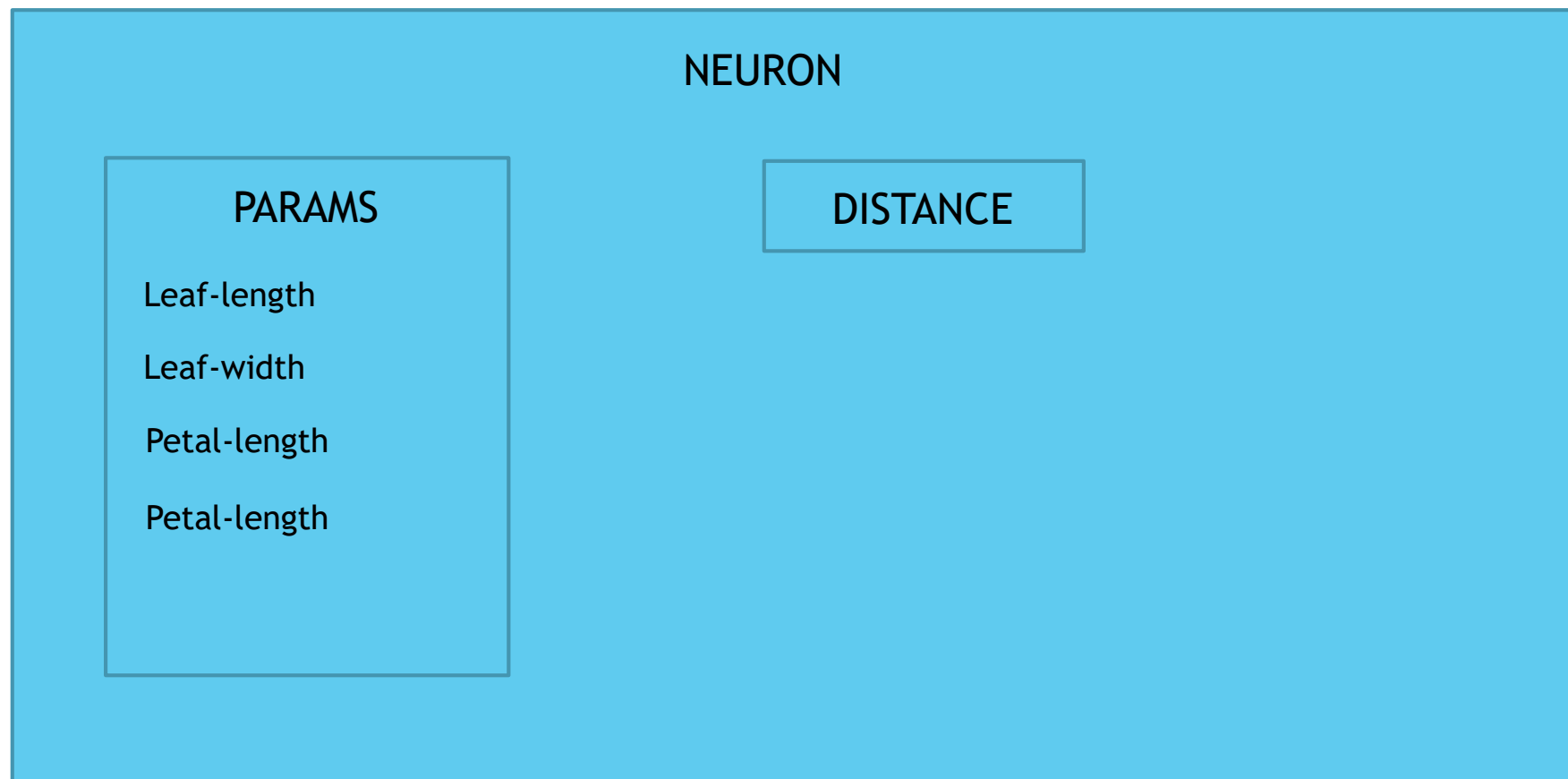
SOM-MAPY SAMOORGANIZUJĄCE SIĘ

NEURON TABLE

SOM-MAPY SAMOORGANIZUJĄCE SIĘ



SOM-MAPY SAMOORGANIZUJĄCE SIĘ



SOM-MAPY SAMOORGANIZUJĄCE SIĘ

1. Utworzenie tablicy neuronów z parametrami zainicjalizowanymi randomowo
2. Obliczenie odległości każdego elementu ze zbioru uczącego do każdego z neuronów w tablicy
3. Wybranie neuronu najbliższego dla neuronu testowanego
4. Aktualizacja parametrów w neuronach. Najbardziej aktualizowany jest neuron najbliższy. Parametry sąsiadów neuronu najmocniej pobudzonego zmieniają się zgodnie ze wzorem biorącym pod uwagę wagę wyliczoną z odległości danego sąsiada od neuronu najbardziej pobudzonego

Walidacja krzyżowa

Walidacja krzyżowa

Walidacja krzyżowa to metoda polegająca na podziale zbioru na zbior uczący i walidacyjny



Każdy z elementów zbioru uczącego (niebieski), ma zaimplementowaną tablicę:

V1	
V2	
V3	

Suma odległości badanego elementu od poszczególnych elementów ze zbioru walidacyjnego, o klasie V1, podzielona przez liczbę zsumowanych elementów

Walidacja krzyżowa

- ▶ Dzięki takiej implemencacji, możemy zbadać do jakiej klasy należą element zbioru uczącego.
- ▶ W moim przypadku po przelosowaniu zbiorów walidacyjnych i uczących dla $k = 10$ otrzymywana skuteczność wynosi 95-100%.
- ▶ dla $k = 20$ skuteczność wynosiła od 90-100%.
- ▶ Przy zwiększającym się k skuteczność spada ponieważ zmniejsza się liczba elementów zbioru walidacyjnego
- ▶ Ostatnim etapem było obliczenie optymalnego k , dla którego skuteczność działania walidacji jest najlepsza. Walidacje przeprowadzono 10 razy na przelosowanych danych. Optymalne $k = 8$. Jest to największe k o stu procentowej skuteczności.

KNN

- ▶ Podstawowa metoda KNN polega na podziale zbioru na zbiór uczący i walidacyjny:



k

- ▶ Następnie obliczana jest odległość danego elementu ze zbioru uczącego do elementu ze zbioru walidacyjnego. W kolejnym kroku obliczana jest ilość wystąpień poszczególnych klas wśród k najbliższych elementów. Klasa która najczęściej się powtarza przypisywana jest elementowi badanemu.
- ▶ Skuteczność tej metody jest znacznie mniejsza niż walidacji krzyżowej. Otrzymywane przez mnie wartości skuteczności są rzędu 70-90%.

Dziękuję za uwagę