

# Wyznaczanie wieloboków Voronoi dla metryki maximum

Adam Kania

Jan Trynda

Algorytmy Geometryczne 2019/2020

# Plan prezentacji

- Krótki opis wykorzystanego algorytmu
- Opis algorytmu w implementacji
- Opis ogólny programu
- Opis problemów napotkanych w trakcie implementacji
- Opis wykorzystanych struktur
- Opis funkcji głównych programu
- Krótki opis pozostałych funkcji programu
- Wizualizacja działania programu
- Przykłady działania
- Bibliografia źródeł

# Krótki opis wykorzystanego algorytmu

- Wykorzystaliśmy Algorytm Fortuny
- Złożoność obliczeniowa algorytmu  $O(n * \log n)$
- Algorytm zmiatania
- Polega na stopniowym dodawaniu kolejnych punktów i punktów przecięć

# Opis algorytmu w implementacji

- Zmiana metryki znacząco skomplikowała zagadnienia geometryczne
- Algorytm Fortuny działa od eventu do eventu – co z tym idzie nie wymaga przechowywania parabol, a jedynie umieszczania punktów w kolejce priorytetowej z odpowiednim kluczem
- Symetralne w metryce maximum składają się z dwóch półprostych i odcinka, lub w szczególnych przypadkach jednej prostej. Zmienia to znacząco wygląd wieloboków.
- W celach implementacyjnych musieliśmy dodać granice (ramkę) okalającą punkty i ograniczającą proste i półproste
- Są 4 rodzaje eventów (środek komórki, załamanie symetralnej(końcówka lub początek odcinka), przecięcie symetralnych, spotkania z granicą obszaru)

# Opis algorytmu w implementacji (opis eventów)

- Środek komórki – nowy środek komórki zostaje dodany do zbioru aktywnych komórek oraz obliczane są symetralne punktu i środków komórek sąsiednich.
- Załamanie symetralnej – obliczony zostaje punkt przecięcia z symetralnymi leżącymi po przeciwnych stronach punktów wyznaczających rozważaną symetralną, do kolejki eventów dodane zostają punkty przecięcia i/lub spotkania z granicą obszaru.
- Przecięcie symetralnych – sprawdzone zostaje, czy należy usunąć środkową z komórek wyznaczających punkt przecięcia i jeżeli tak, to jest ona usuwana. W miejscu przecięcia zostają dodane dwa zdarzenia będące załamaniami symetralnej (lub jedno zdarzenie jeżeli jedna z komórek została usunięta ze zbioru aktywnych komórek).
- Spotkanie z granicą obszaru – dodaje linie dochodzące do granicy rozważanego obszaru do diagramu Voronoi.

# Opis ogólny programu

- Program implementuje podział przestrzeni dwuwymiarowej na wieloboki Voronoi w Metryce Maximum z wykorzystaniem Algorytmu Fortuny
- Program składa się z kilku plików z których VoronoiCalculator.py jest plikiem wykonywalnym
- Program napisany jest w całości w języku Python w wersji 3.7
- Wizualizacja przygotowana została w oparciu o narzędzia do wizualizacji stworzone przez mgr inż. Krzysztofa Podsiadło na potrzeby laboratoriów z Algorytmów Geometrycznych
- Program zachowuje złożoność Algorytmu Fortuny to jest  $O(n * \log n)$

# Opis problemów napotkanych w trakcie implementacji

- Dokładność zmiennoprzecinkowa wymusiła na nas dodawanie współczynnika tolerancji w wielu miejscach w programie.
- Zarówno wyznaczanie symetralnych jak i wyznaczanie ich przecięć wymagało, bardziej niż się spodziewaliśmy, skomplikowanego modelu matematyczno/algorytmicznego.

# Opis wykorzystanych struktur

- Kolejka priorytetowa sortująca punkty wedle klucza wysokości Y a drugiej kolejności szerokości X w celu przechowywania eventów
- Drzewo czerwono-czarne, w celu wydajnego dostępu do komórek (znajdowanie sąsiadów) i związanych z nimi eventami, implementujące funkcje:
  - Znajdującą poprzednika
  - Znajdującą następcę
  - Dodawania
  - Znajdowania elementu
  - Usuwania elementu
- Listy elementów np. Linia to lista odcinków składających się z listy dwóch punktów składających się z listy dwóch współrzędnych
- Struktury z pliku DataType:
  - Event – przechowuje współrzędne eventu, klucz, komórki z nim powiązane, flagę czy nadal należy go rozpatrywać
  - Cell – przechowuje położenie, symetralne, eventy i punkty powiązane z komórką



# Opis Funkcji głównych programu

- Funkcja `__init__` w pliku `VoronoiCalculator`:
  - Tworzy obiekty typu `Voronoi` (klasa implementująca wszystkie metody potrzebne do działania algorytmu i sam algorytm)
  - Dodaje punkty początkowe jako eventy
  - Tworzy potrzebne struktury
- Funkcja `process` w pliku `VoronoiCalculator`:
  - Zawiera implementację głównej części algorytmu
  - Dla kolejnych eventów klasyfikuje je i wywołuje odpowiednią podfunkcję
  - Działa aż nie zostaną już żadne eventy podmiotem do rozpatrzenia

# Opis Funkcji głównych programu

- Funkcja bisector z pliku MaxMetric:
  - Zwraca symetralną odległości między zadanymi punktami
  - Symetralna jest w postaci listy odcinków
  - W związku z charakterystyką Metryki Maximum symetralna (poza szczególnymi przypadkami) składa się z odcinka nachylonego pod kątem  $(+/-) \pi/4$ , odcinka poziomego/pionowego oraz kolejnego odcinka nachylonego pod kątem  $(+/-) \pi/4$
  - Wykorzystuje podfunkcje dla większej czytelności kodu
- Funkcja cross z pliku MaxMetric:
  - Zwraca punkt przecięcia się dwóch linii (lista odcinków) lub False, gdy taki punkt nie istnieje
  - Ma za zadanie znajdować przecięcia symetralnych, które następnie są kolejnymi eventami (tyle że z odpowiednio mniejszym kluczem)
  - Wykorzystuje podfunkcję do znajdowania przecięcia linii i podfunkcję do sprawdzania czy przecięcie leży w zakresie odcinka

# Krótki opis pozostałych funkcji programu

- Z pliku VoronoiCalculator

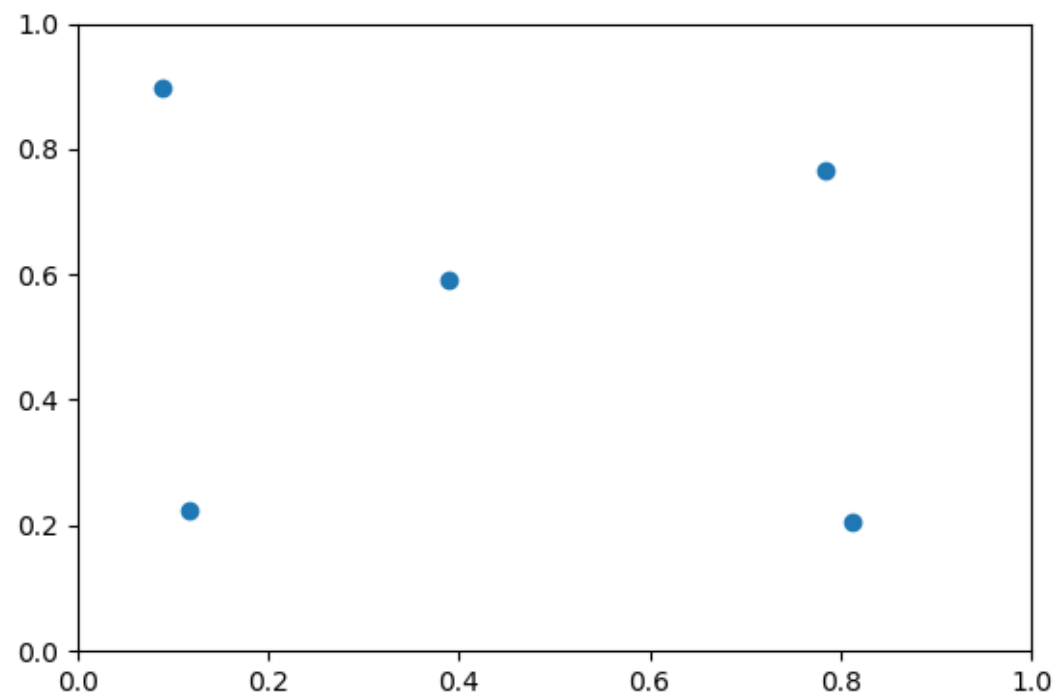
- `_intersection_to_event` - Dodaje przewidywany punkt przecięcia do listy eventów
- `_process_cell` - przetwarza eventy będące środkami komórek. Dodaje komórkę do struktury przechowującej aktywne komórki i szuka symetrycznych oraz ewentualnych punktów przecięcia.
- `_process_intersection` - przetwarza eventy będące punktami przecięcia. Jeżeli jakaś komórka przestaje być aktywna usuwa ją ze struktury aktywnych komórek. Szuka ewentualnych punktów przecięcia.
- `_process_bend` - przetwarza eventy będące zgięciami linii. Funkcja dodaje początkowy kawałek linii do diagramu Voronoi i szuka punktu przecięcia.
- `_invalidate_events` - oznacza przestarzałe eventy jako nieważne, aby nie zostały przetworzone.
- `_extract_line_part(line, a, b)` zwraca część linii znajdującą się między a i b
- `_make_scene()` dodaje krok do wizualizacji

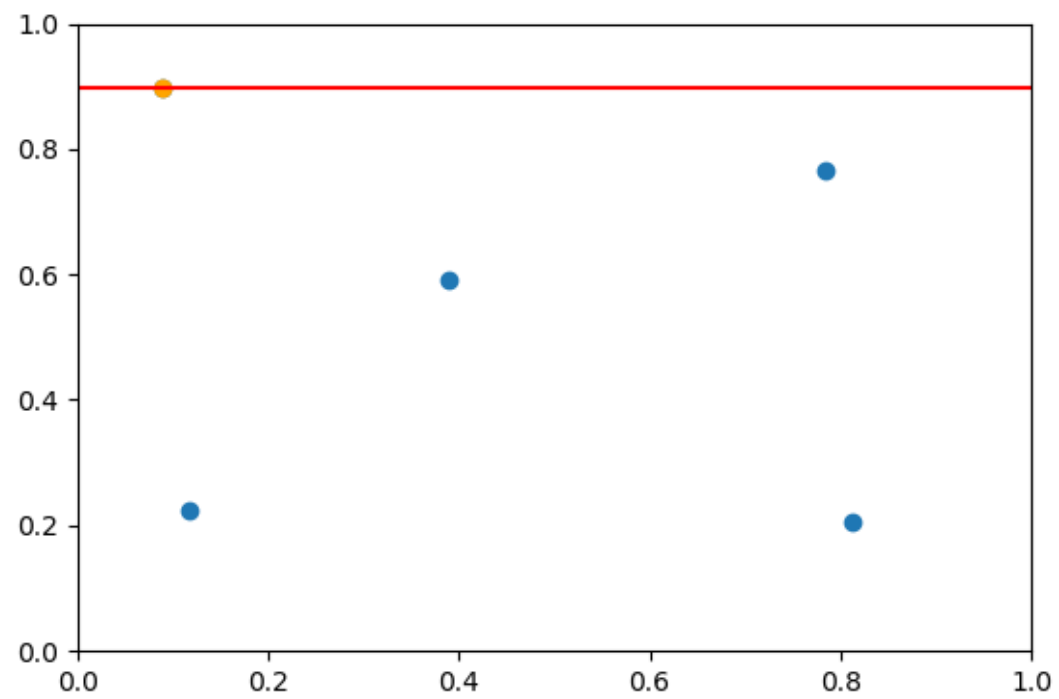
# Krótki opis pozostałych funkcji programu

- Z pliku MaxMetric
  - findCross, line\_intersection to funkcje pomocnicze do funkcji cross
  - rightEnd, leftEnd, eq, samepoint to funkcje pomocnicze do funkcji bisector
- Z pliku RBTREE
  - insert, remove – dodają i usuwają konkretny węzeł do drzewa
  - find – znajdują konkretny węzeł w drzewie
  - successor i predecessor znajdują poprzedni i kolejny wezeł w drzewie

# Wizualizacja Działania algorytmu

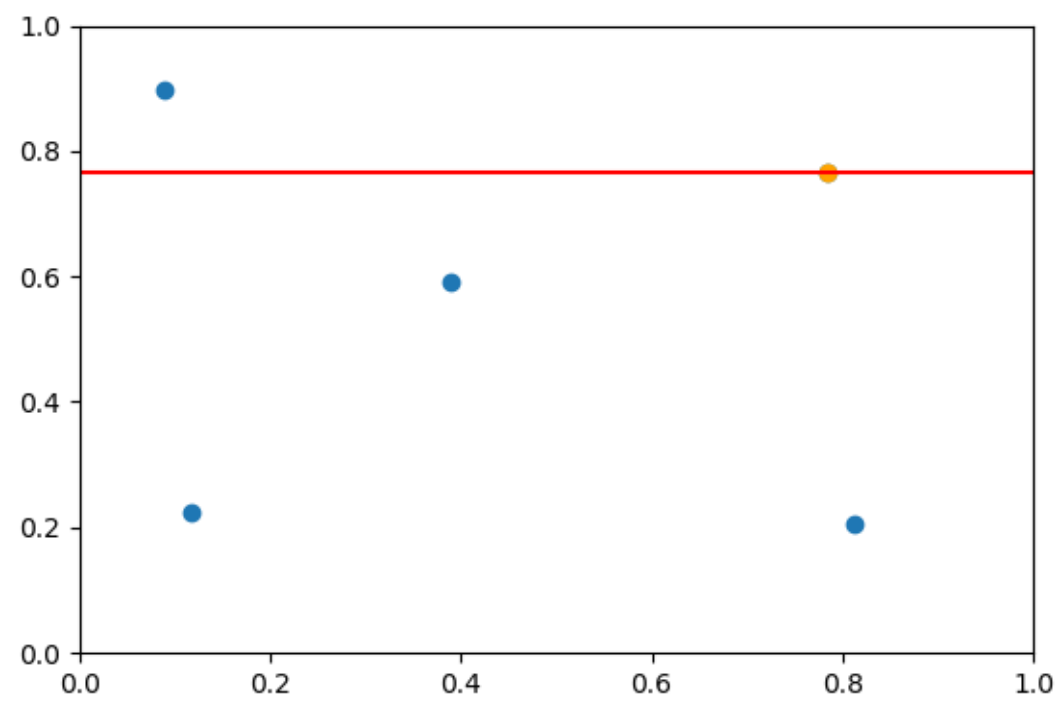
Przykład dla 5 punktów





Previous

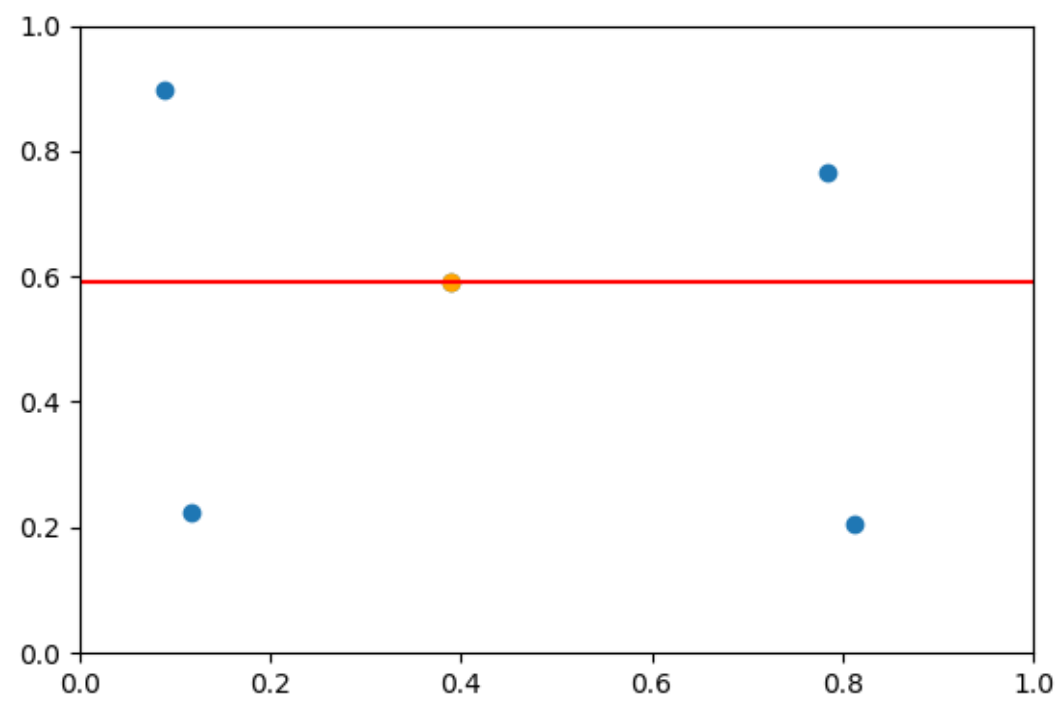
Next



Previous

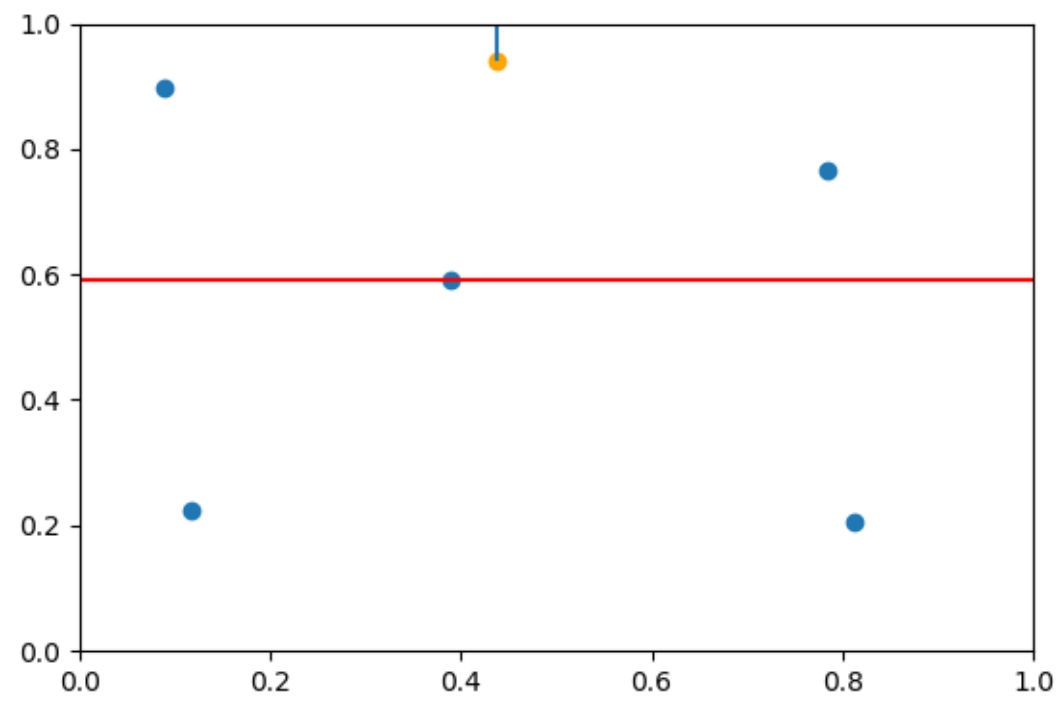
Next





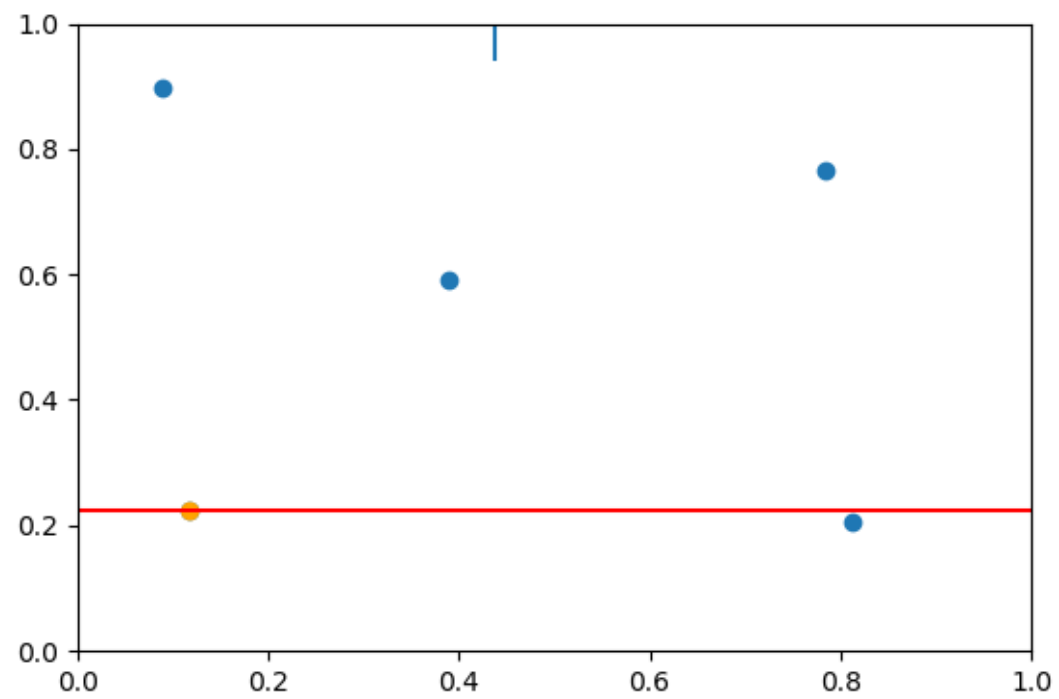
Previous

Next



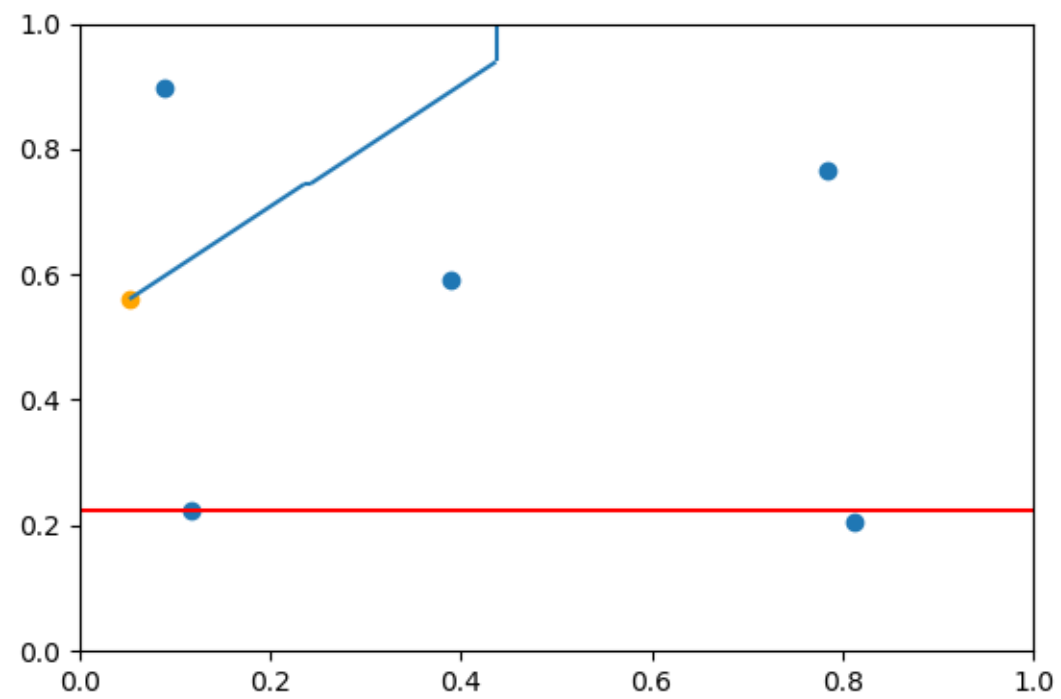
Previous

Next



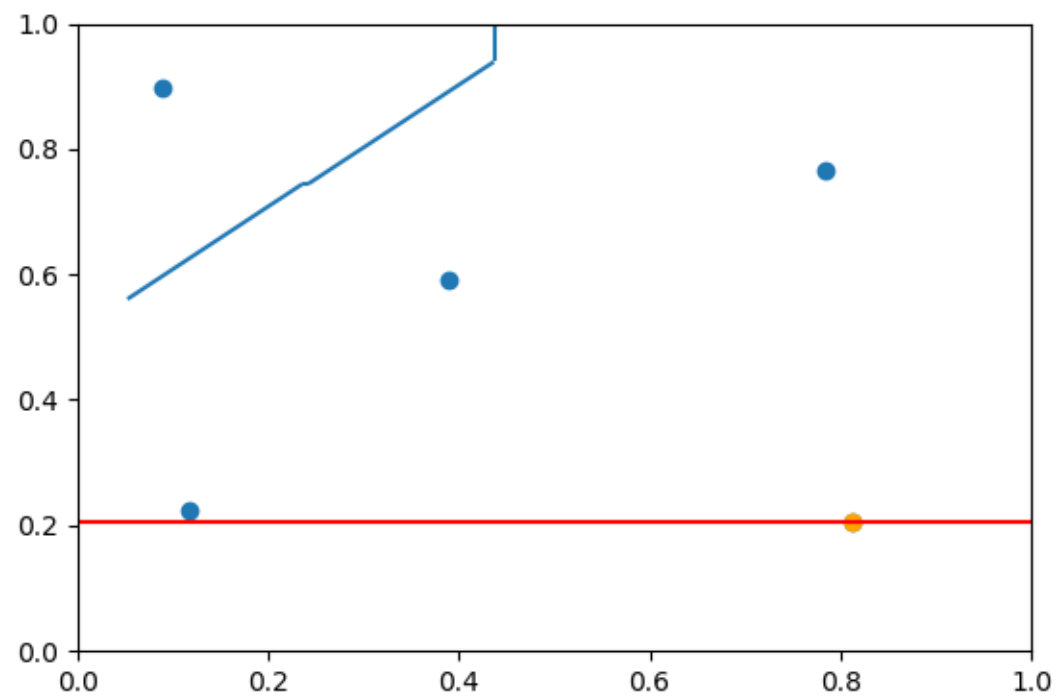
Previous

Next



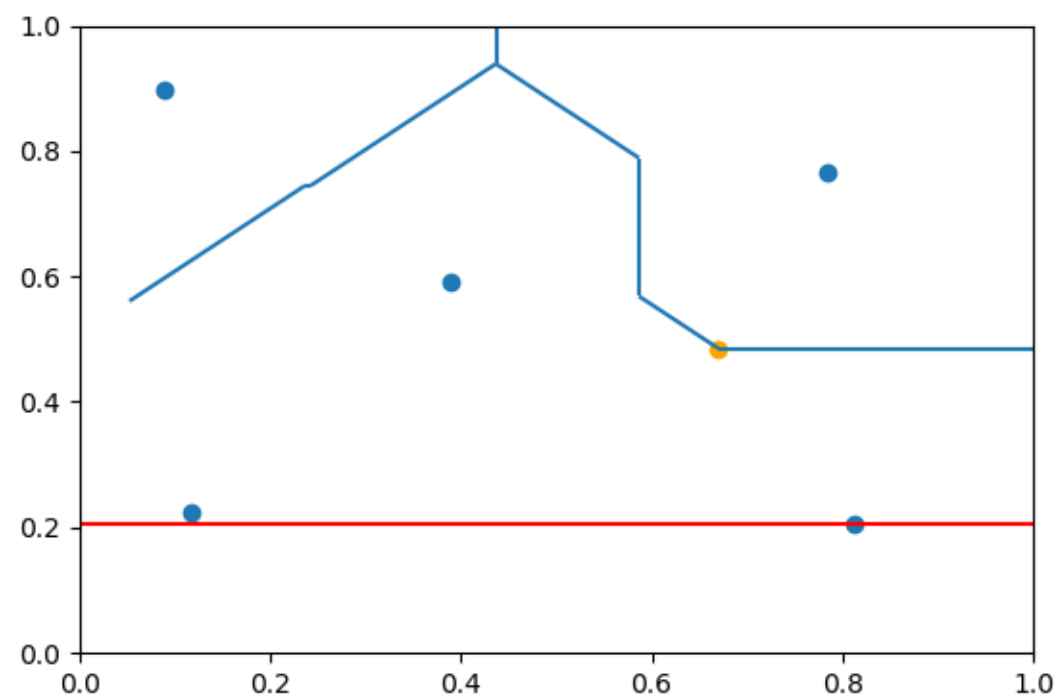
Previous

Next



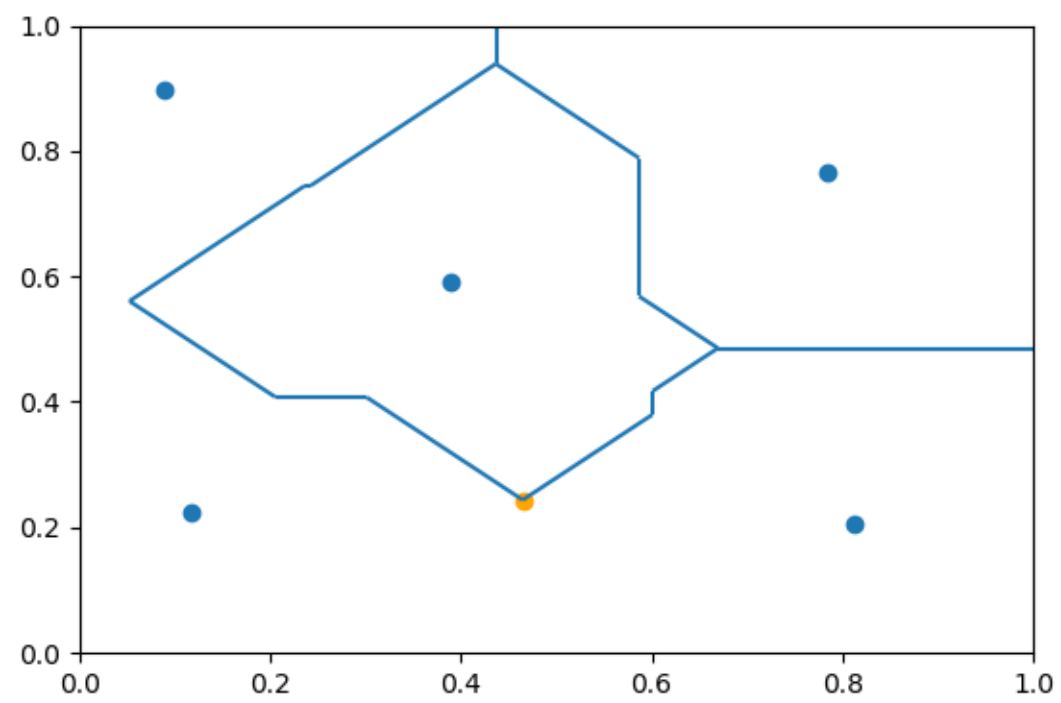
Previous

Next



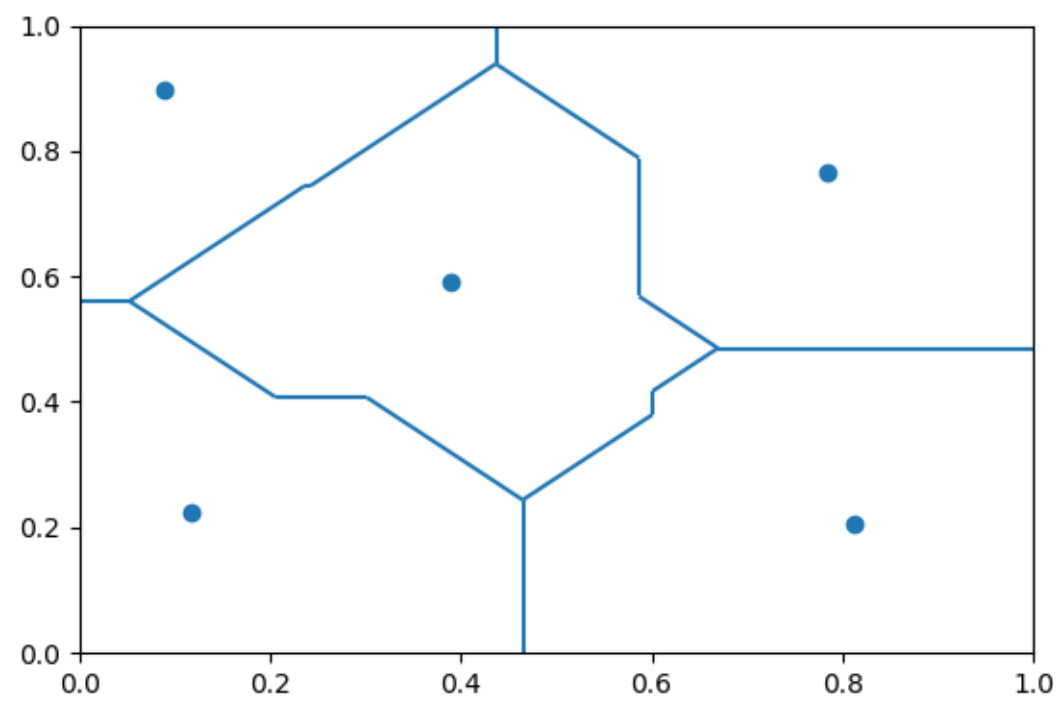
Previous

Next



Previous

Next

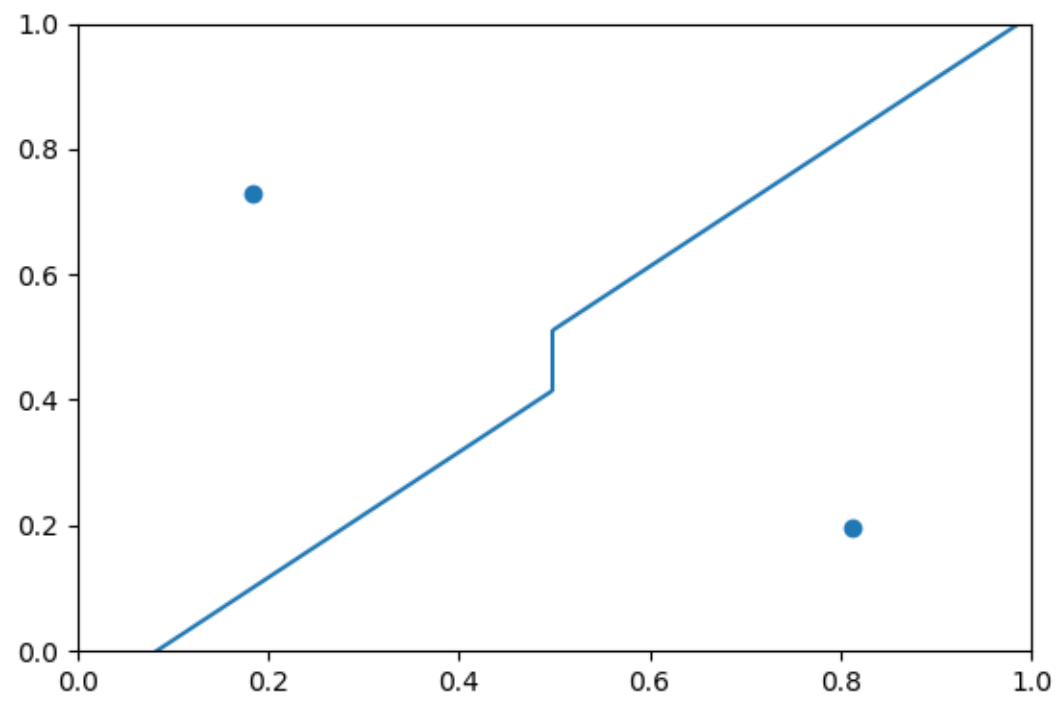


Previous

Next

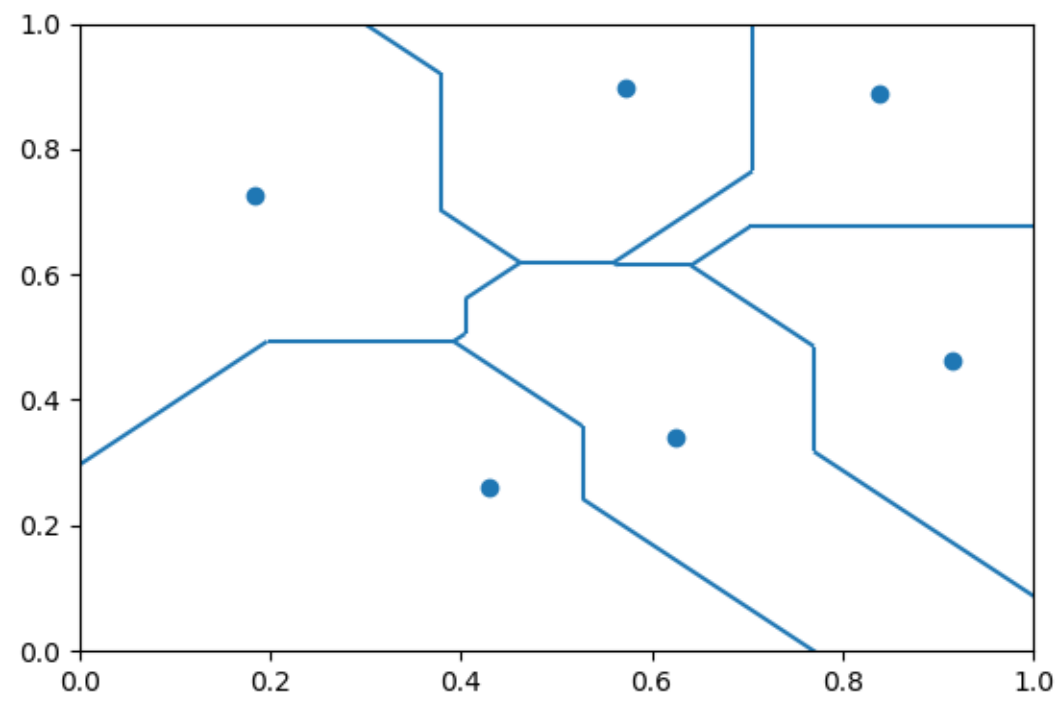


Przykłady działania



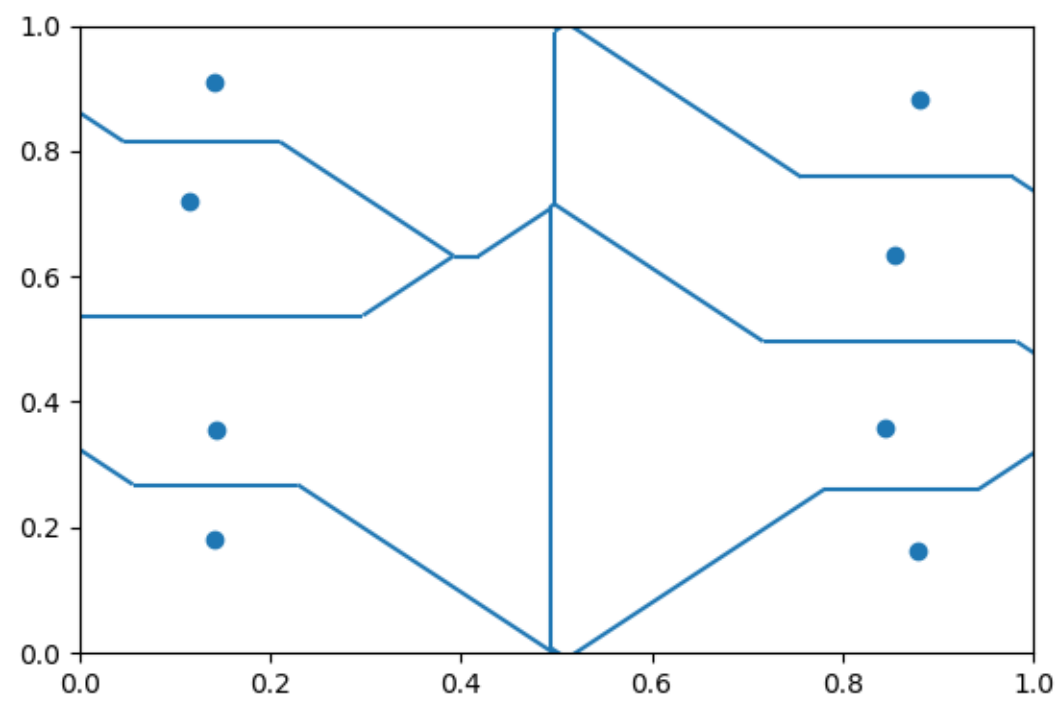
Previous

Next



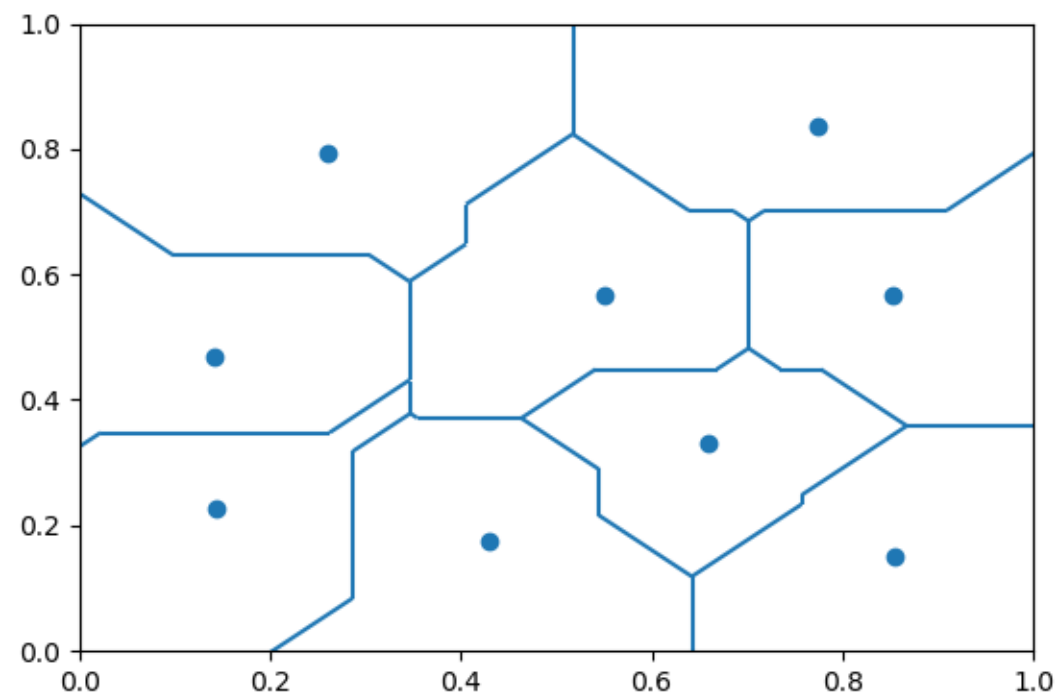
Previous

Next



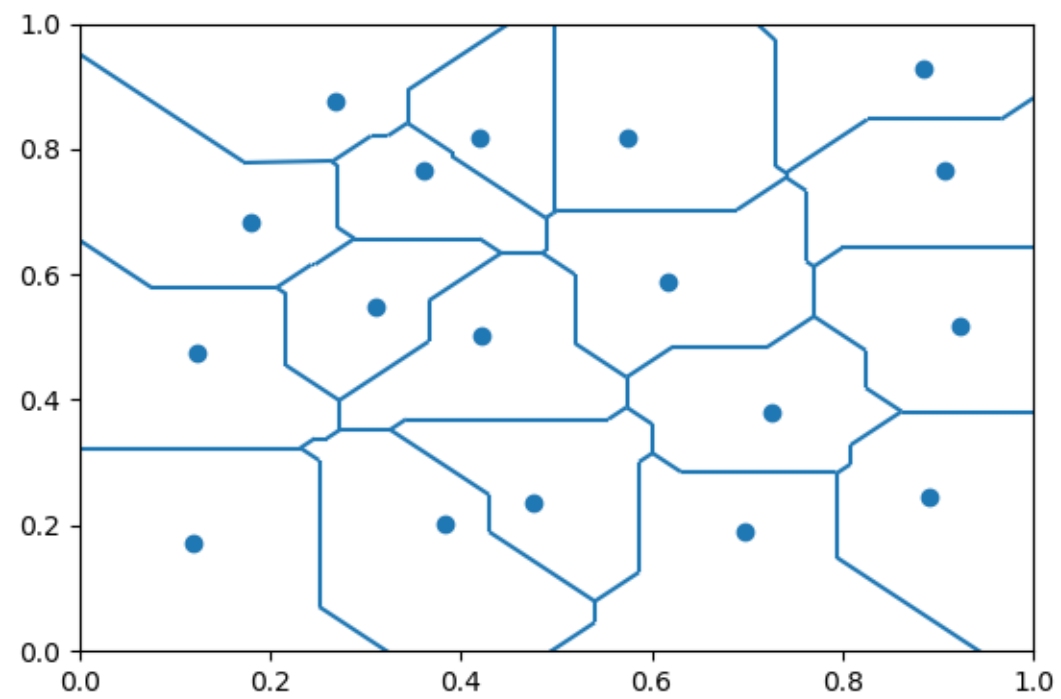
Previous

Next



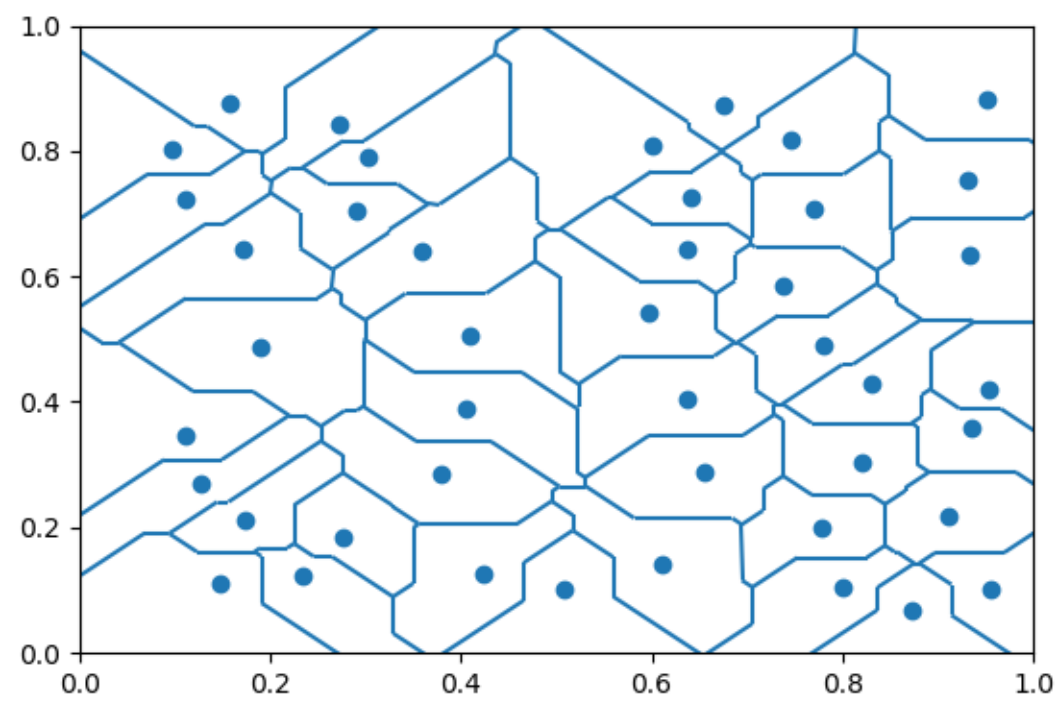
Previous

Next



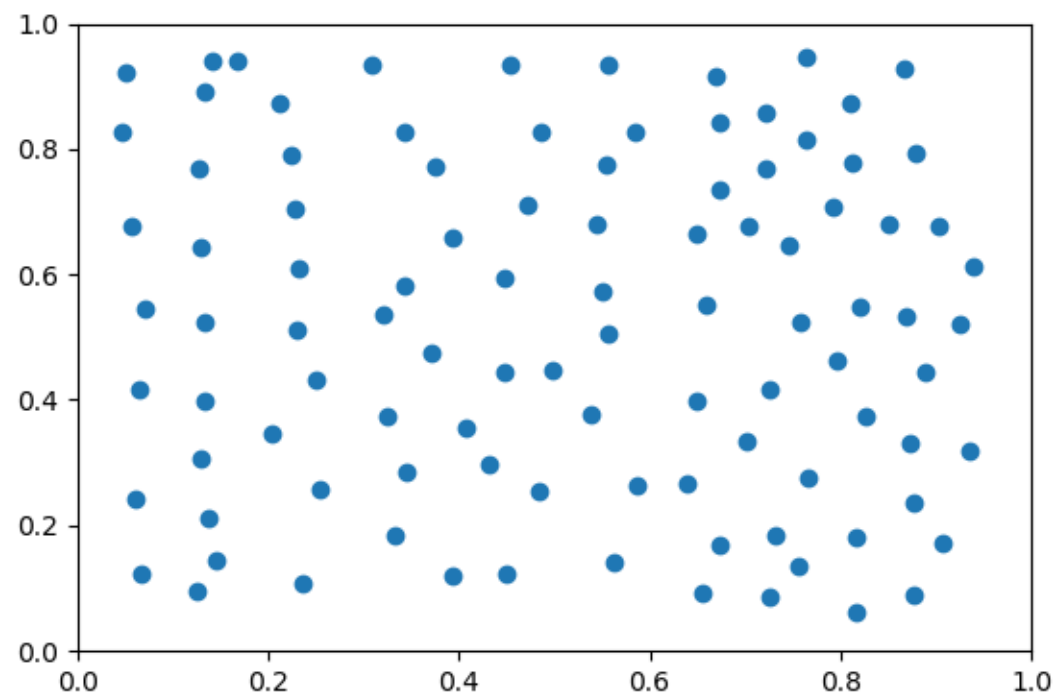
Previous

Next

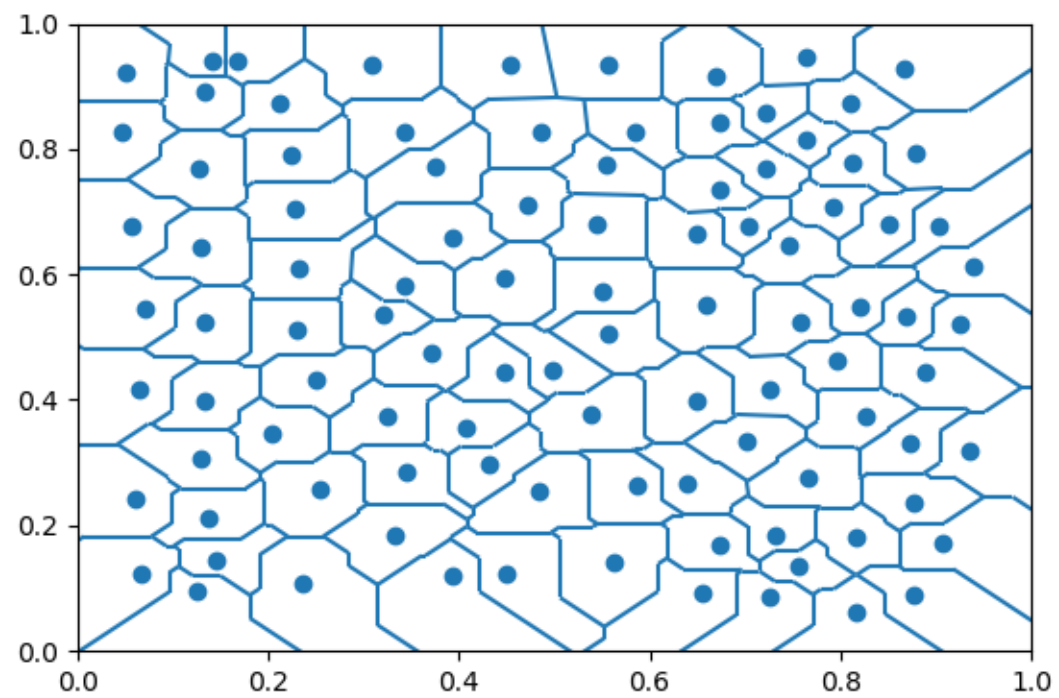


Previous

Next







Previous

Next

# Bibliografia

- <https://github.com/MSingh3012> oraz <https://github.com/Zhylkaaa>  
RBTre
- <https://stackoverflow.com/users/1212596/paul-draper>  
Lines\_Intersection
- Model wizualizacji z zajęć

Koniec