Szymon Nowak-Trzos 411246

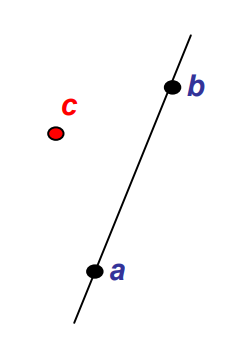
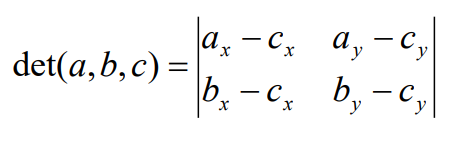
Informatyka rok II, gr.2 środa nieparzysta 13:00-14:30

Algorytmy Geometryczne – Ćwiczenie 1 Sprawozdanie

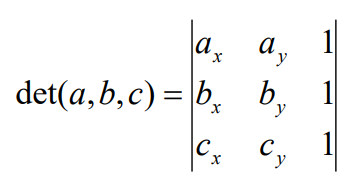
1. Wstęp
   1. Cel Ćwiczenia

Celem ćwiczenia była implementacja algorytmu sprawdzającego położenie punktów w stosunku do prostej przeprowadzonej przez dwa punkty oraz przeprowadzenie analizy wyników obliczeń na kilku zbiorach danych.

* 1. Teoria

Aby określić, czy punkt **c** leży po prawej, po lewej stronie prostej **ab** czy na niej należało określić wyznacznik którejkolwiek z poniższych macierzy:

**1)**



**2)**

Wyznacznik równy 0 wskazuje, że punkt **c** jest współliniowy z **a** i **b**. W przeciwnym wypadku znak wyznacznika rozróżnia punkty po lewej i po prawej stronie **ab.**

* 1. Użyte Narzędzia

Ćwiczenie zostało wykonane dzięki zmodyfikowanemu narzędziu graficznemu załączonego na stronie UPEL. Do niezbędnych obliczeń został wykorzystany język programowania Python oraz biblioteka numpy. Ponadto wykresy zostały wygenerowano przy pomocy biblioteki matplotlib, a czas został wyznaczony poprzez bibliotekę time. Obliczenia, których wyniki zostały uwzględnione w tabelach, oraz wszystkie wykresy zostały wygenerowane na procesorze Intel Core i5-1135G7 2.40GHz

1. Szczegóły wykonywania ćwiczenia
   1. Zbiory Danych

Wygenerowane zostały cztery zbiory danych:

1. 10^5 losowych punktów o współrzędnych z przedziału [-1000, 1000]
2. 10^5 losowych punktów o współrzędnych z przedziału [-10^14, 10^14],
3. 1000 losowych punktów leżących na okręgu o środku (0,0) i promieniu R=100,
4. 1000 losowych punktów o współrzędnych z przedziału [-1000, 1000] leżących na prostej wyznaczonej przez wektor ([-1, 0], [1, 0.1]).

Do wygenerowania zbiorów A i B wykorzystane zostały funkcje random.uniform z biblioteki random języka Python. Zbiory C i D uzyskane zostały przez parametryzację odpowiednio okręgu o zadanym środku i promieniu oraz linii przechodzącej przez dwa punkty.

|  |  |
| --- | --- |
| Rysunek 1. | Rysunek 2. |
| Rysunek 3. | Rysunek 4. |

* 1. Wyznaczniki

Do określenia położenia punktów w każdym zbiorze danych zostały wykorzystane cztery różne algorytmy liczenia wyznaczników:

* „wlasny\_3x3” – własna implementacja obliczania wyznacznika macierzy 3x3,
* „wlasny\_2x2” – własna implementacja obliczania wyznacznika macierzy 2x2,
* „np\_3x3” – funkcja numpy.linalg.det() do obliczania wyznacznika macierzy 3x3,
* „np\_2x2” – funkcja numpy.linalg.det() obliczania wyznacznika macierzy 2x2.
  1. Tolerancja – **ε**

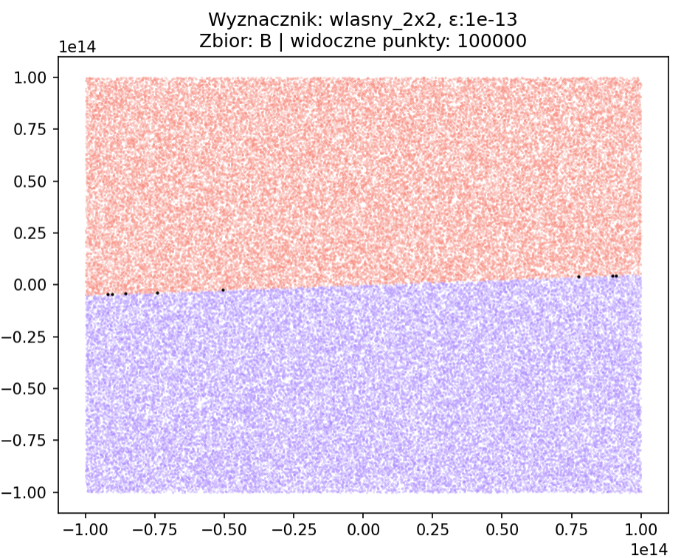
Ze względu na niedoskonałość precyzji obliczeń w komputerze zastosowano kilka różnych granic tolerancji **ε**:

* **ε1** = 10^(-13)
* **ε2** = 10^(-14)
* **ε3** = 10^(-15)
* **ε4** = 10^(-16)
  1. Wykresy

Na wykresach kolorem czerwonym (#f9897b) zostały przedstawione punkty po prawej stronie prostej **ab**, kolorem fioletowym (#ad8cff) punkty po lewej stronie zaś kolorem czarnym punkty zakwalifikowane jako współliniowe do prostej.

Wszystkie wykresy mają podany tytuł w formie:

„Wyznacznik: \_\_\_\_\_, ε: \_\_\_\_\_  
Zbior: \_\_ | widoczne punkty: \_\_\_\_\_”



Rysunek 5. Przykładowy wykres podziału punktów w zbiorze

B przy użyciu wyznacznika „wlasny\_2x2” o tolerancji 10^-13

1. Analiza Danych
   1. Zbiory A i C

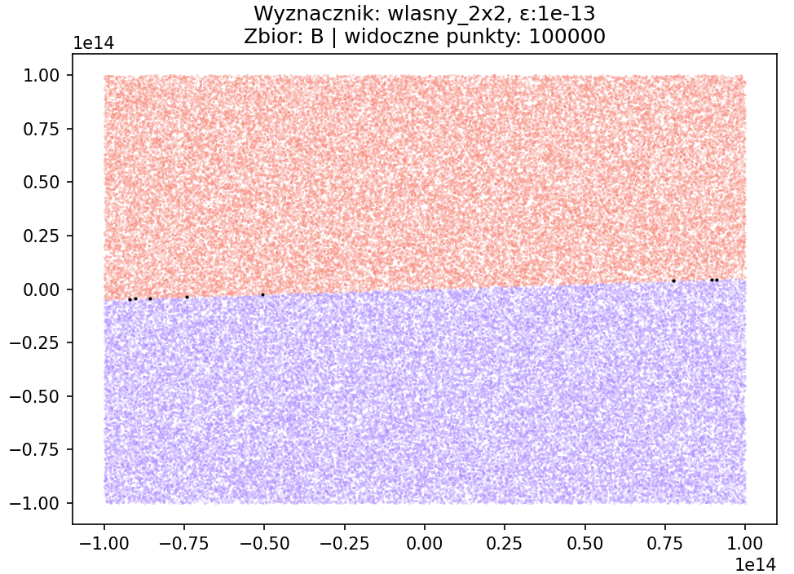
W zbiorach A i C nie wykryto żadnych współliniowych punktów, co pokrywa się z przewidywanym wynikiem, gdyż prawdopodobieństwo natrafienia na taki punkt w tych zbiorach jest znikome. Nie ma również zauważalnych różnic pomiędzy klasyfikacjami przeprowadzanymi przez różne wyznaczniki z różnymi tolerancjami, zatem w dalszej części pracy skupię się na zbiorach B i D.

|  |  |
| --- | --- |
| Rysunek 6. Klasyfikacja punktów w zbiorze A | Rysunek 7. Klasyfikacja punktów w zbiorze C |
| Rysunek 8. Przykładowy wykres podziału punktów w zbiorze A | Rysunek 9. Przykładowy wykres podziału punktów w zbiorze C |

* 1. Zbiory B i D

Już w zbiorze B możemy zauważyć różnice w klasyfikacji punktów pomiędzy wyznacznikami.

Rysunek 6. Klasyfikacja punktów w zbiorze B

Pomimo dużych odstępów pomiędzy punktami (średnia minimalna odległość dwóch punktów od siebie w zbiorze B to 2.9\*10^6, co wyjaśnia również brak wpływu tolerancji na wynik klasyfikacji) wyznaczniki 2x2 kwalifikują niektóre punkty jako współliniowe do prostej **ab**. Jest to prawdopodobnie spowodowane niską precyzją operacji odejmowania dużych liczb od małych i odwrotnie.

Rysunek 11. Wykres podziału punktów

Najciekawsze wnioski można wyciągnąć z analizy zbioru D, w którym wszystkie punkty powinny być zaklasyfikowane jako współliniowe do prostej **ab.**

Rysunek 12. Klasyfikacja punktów w zbiorze D

Widzimy jednak duże odchyły od prawidłowej klasyfikacji punktów. Wyznaczniki 3x3 najdłużej utrzymują 100% dokładność, aczkolwiek już przy **ε3** = 10^(-15) prawdopodobieństwo dobrej kategoryzacji jest bardzo małe. Wyznacznik „wlasny\_2x2”, choć nie ma perfekcyjnej dokładności, zachowuje precyzję przez wszystkie **ε**, dobrze kwalifikując tym samym 70% punktów. Wyznaczniki biblioteczne wraz ze zmniejszaniem **ε** tracą zupełnie możliwość wykrycia współliniowych punktów.

|  |  |
| --- | --- |
| Rysunek 13.  1000 punktów współliniowych | Rysunek 14.  711 punktów współliniowych |
| Rysunek 15.  883 punktów współliniowych | Rysunek 16.  89 punktów współliniowych |

|  |  |  |
| --- | --- | --- |
| Lewa Strona | Współliniowe | Prawa Strona |
| Rysunek 16. | Rysunek 17. | Rysunek 18. |
| Rysunek 19. | Rysunek 20. | Rysunek 21. |
| Rysunek 22. | Rysunek 23. | Rysunek 24. |
| Rysunek 25. | Rysunek 26. | Rysunek 27. |

1. Pomiary czasowe funkcji

Czas poszczególnych funkcji został zmierzony 10 razy na zbiorze danych w postaci 10^6 punktów z przedziału [-1000, 1000], a następnie uśredniony aby dać wartości zamieszczone w tabeli poniżej.

Rysunek 16.

Tabela zmierzonych czasów

Możemy zauważyć, że samodzielne implementacje wyznaczników są znacząco szybsze od tych bibliotecznych. Może to wynikać ogólniejszej implementacji funkcji numpy.linalg.det(), która przyjmuje jako wejście macierze **n x n**.

1. Wnioski

Komputerowe sprawdzanie współliniowości punktów, choć nie trudne, nie jest jednak w pełni oczywiste. Nawet podczas tak prostego problemu małe różnice implementacyjne mogą prowadzić do ogromnych różnic w wynikach analizy. Wprawdzie na ¾ zbiorów danych nie było znaczącej różnicy, to na sytuacji granicznej wszystkie zaimplementowane algorytmy miały swoje wady. Samodzielnie zaimplementowany algorytm do obliczania wyznacznika 3x3 bardzo dokładnie wyznaczył współliniowość punktów, aczkolwiek większą dokładność przy mniejszych tolerancjach zapewniał algorytm „wlasny\_2x2”. Ponadto funkcje własne potrafią być nawet 15 razy szybsze od ich odpowiedników bibliotecznych.