PCA Kernel Trick

11 listopada 2017

Grzegorz Borkowski

---

1. WSTĘP

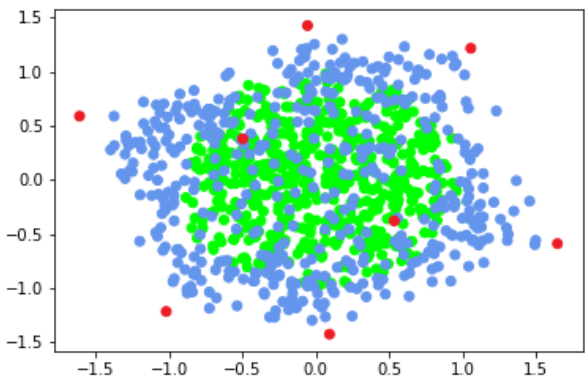
Zadanie zostało zaimplementowane w Pythonie 3.6.1
Do raportu został dołączony Jupyter Notebook z kodem źródłowym programu.

2. ZADANIE

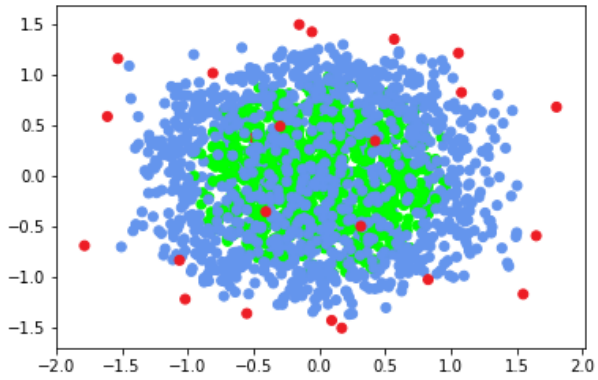
2.1 Treść zadania

A: Wróćmy do hiperkuli wpisanej w hipersześcian z pierwszego zadania. Pokolorujmy narożniki sześcianu na czerwono, punkty w jego wnętrzu (ale nie we wnętrzu kuli) na niebiesko, a punkty z kuli na zielono. Wykorzystać metodę PCA by wykonać wizualizację (rzut na płaszczyznę 2D) tejże sytuacji dla 3, 4, 5, 7 i 13 wymiarów.

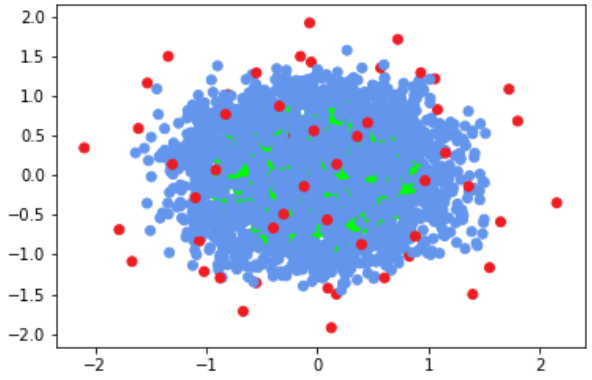
B: Wygenerujmy 2 zbiory danych wyglądające +- tak, jak na rysunkach a i b (punkty w różnych kolorach należą do różnych klas). Potraktujmy je klasycznym PCA. Przygotujmy diagramy prezentujące rozkład punktów w nowo znalezionej przestrzeni. Nanieśmy również na pierwotne rysunki strzałki prezentujące wektory będące znalezionymi "principal components" (gdzie długość wektora jest proporcjonalna do wariancji przez niego wyjaśnianej). Spoiler - efekty będą dość mizerne. Można próbować z tym walczyć stosując tzw. kernel trick - zamiast klasycznego iloczynu skalarnego wykorzystać inną, odpowiednio spreparowaną funkcję. Zrzutujmy więc oba zbiory w nową przestrzeń, ale tym razem wykorzystując kernel PCA z kernelami "cosine" (tu warto sprawdzić jaki wpływ na wynik będzie miało wcześniejsze wyśrodkowanie danych lub jego brak) i "rbf" (radial basis function - tu należy sprawdzić różne wartości parametru gamma wpływającego na pracę kernela) oraz przygotujmy diagramy prezentujące efekty.



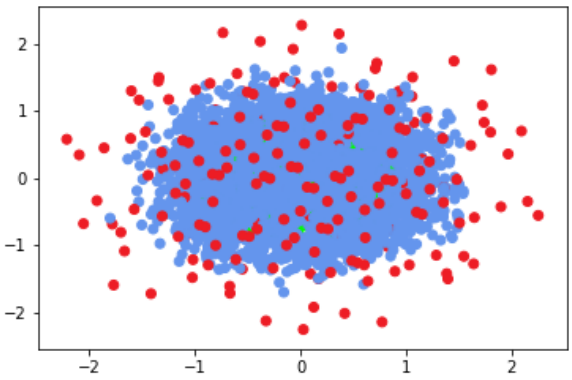
Rysunek 1. Wymiar=3, Rzut elementów zbioru na płaszczyznę 2D



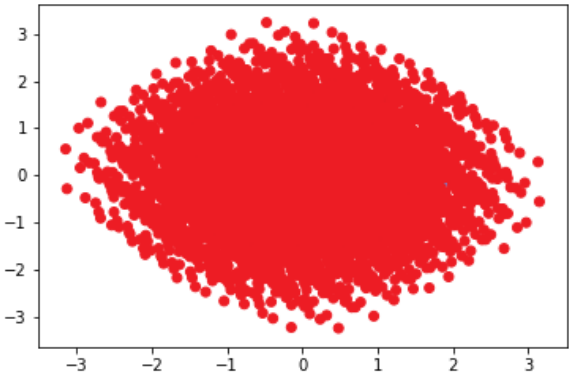
Rysunek 2. Wymiar=4, Rzut elementów zbioru na płaszczyznę 2D



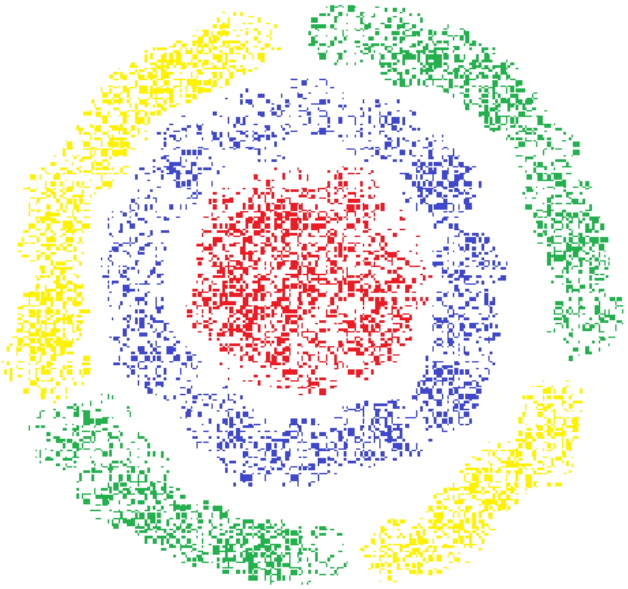
Rysunek 3. Wymiar=5, Rzut elementów zbioru na płaszczyznę 2D



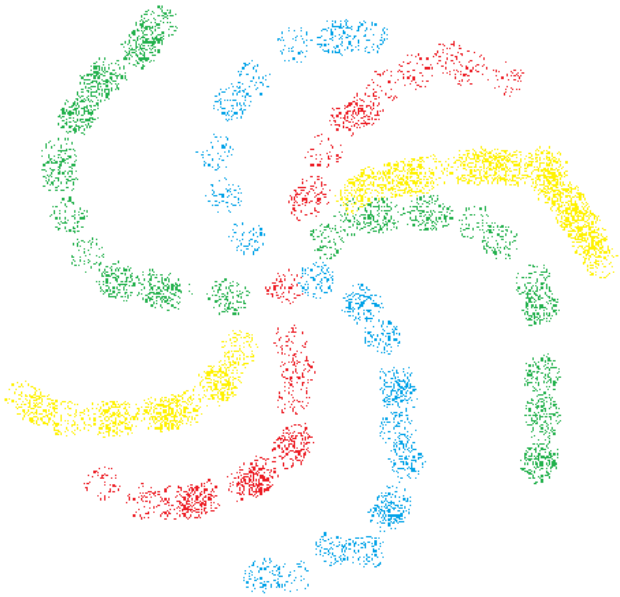
Rysunek 4. Wymiar=7, Rzut elementów zbioru na płaszczyznę 2D



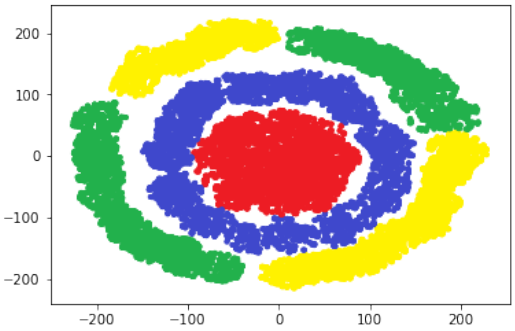
Rysunek 5. Wymiar=13, Rzut elementów zbioru na płaszczyznę 2D



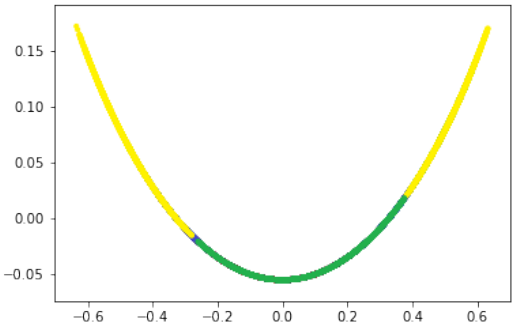
Rysunek 6. Wizualizacja zbioru wejściowego "A"



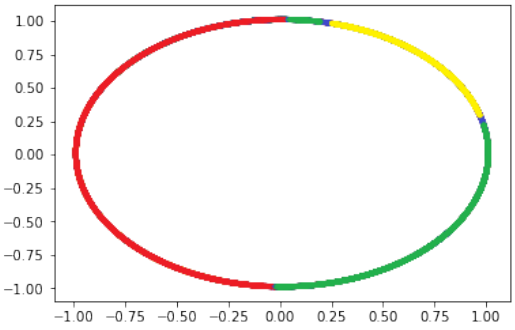
Rysunek 7. Wizualizacja zbioru wejściowego "B"



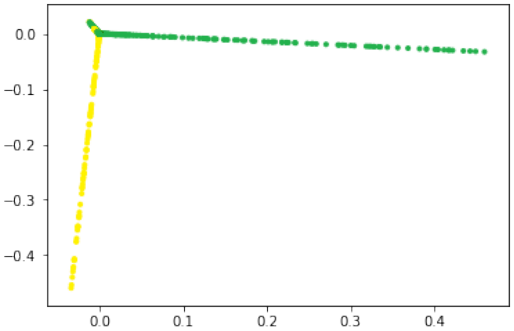
Rysunek 8. "Klasyczne" PCA dla zbioru A



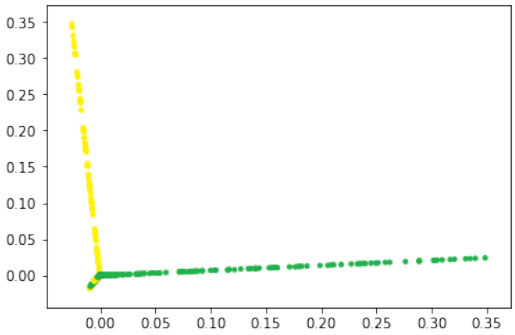
Rysunek 9. Kernel PCA dla zbioru A, kernel = Cosine



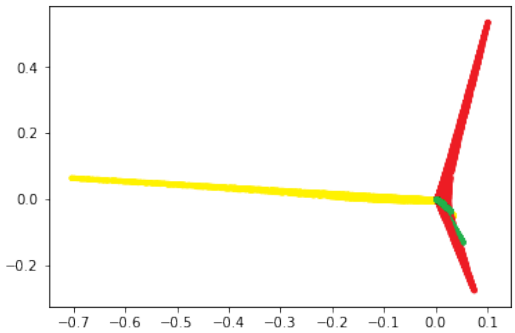
Rysunek 10. Kernel PCA dla zbioru A, kernel = Cosine, punkty zostały przeskalowane i przesunięte tak, że środkiem jest punkt (0,0)



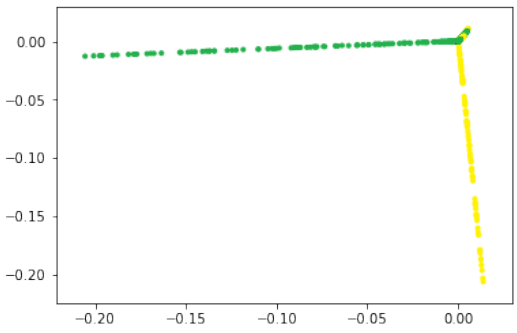
Rysunek 11. Kernel PCA dla zbioru A, kernel = rbf



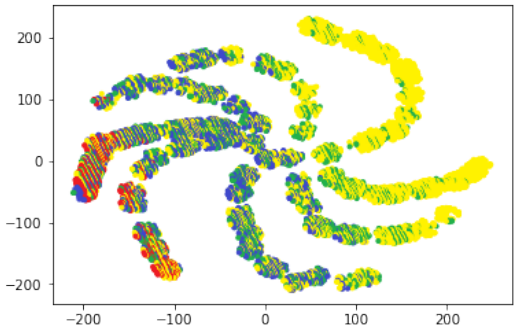
Rysunek 12. Kernel PCA dla zbioru A, kernel = rbf, gamma = 1



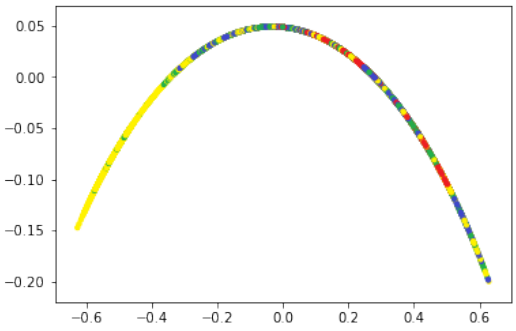
Rysunek 13. Kernel PCA dla zbioru A, kernel = rbf, gamma = 0.01



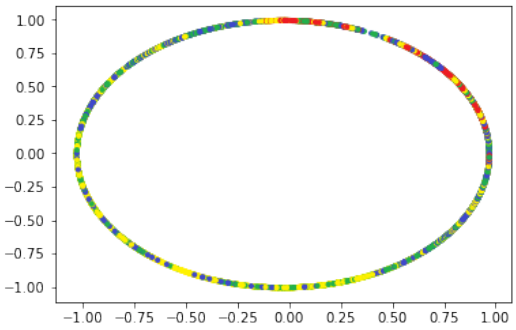
Rysunek 14. Kernel PCA dla zbioru A, kernel = rbf, gamma = 10



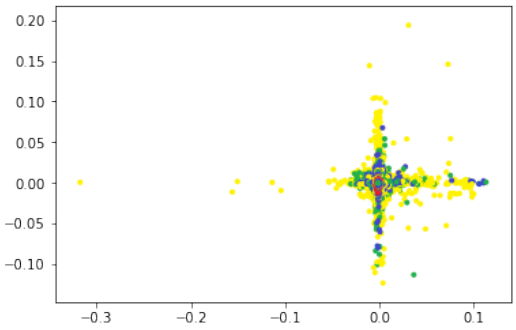
Rysunek 15. "Klasyczne" PCA dla zbioru B



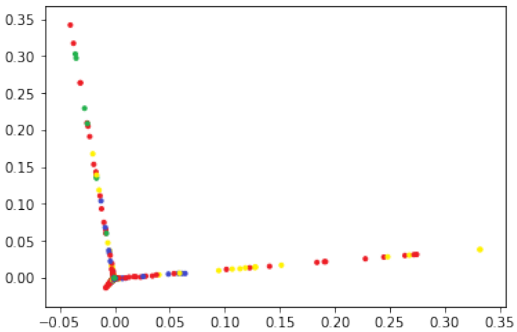
Rysunek 16. Kernel PCA dla zbioru B, kernel = Cosine



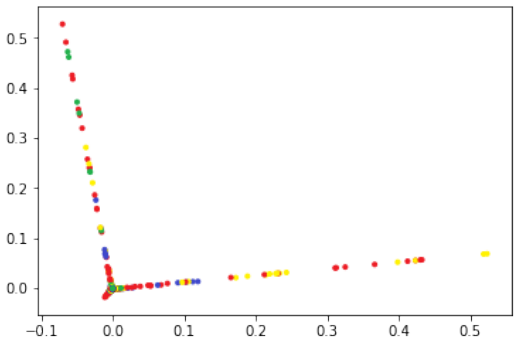
Rysunek 17. Kernel PCA dla zbioru B, kernel = Cosine, przeskalowane i przesunięte do (0,0)



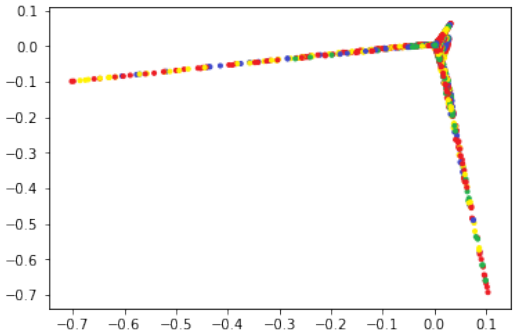
Rysunek 18. Kernel PCA dla zbioru B, kernel = rbf, gamma = 100



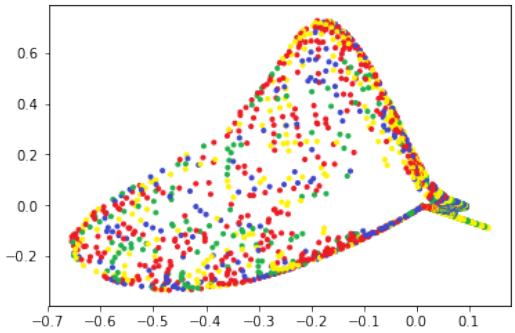
Rysunek 19. Kernel PCA dla zbioru B, kernel = rbf, gamma = 10



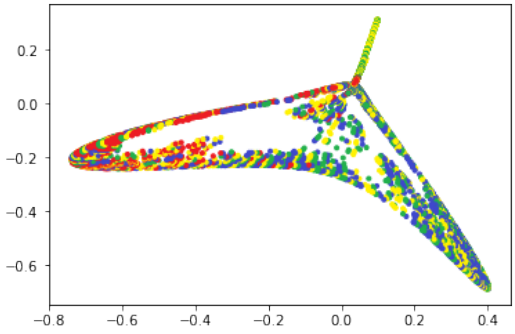
Rysunek 20. Kernel PCA dla zbioru B, kernel = rbf, gamma = 1



Rysunek 21. Kernel PCA dla zbioru B, kernel = rbf, gamma = 0.1



Rysunek 22. Kernel PCA dla zbioru B, kernel = rbf, gamma = 0.01



Rysunek 23. Kernel PCA dla zbioru B, kernel = rbf, gamma = 0.001

3. WNIOSKI

(1) Wraz ze wzrostem wymiarów coraz więcej punktów leży poza hipersferą
(2) PCA pozwala nam na stworzenie dwuwymiarowego wykresu, który dla pewnych zbiorów danych może mniej więcej pokazać nam naturę danych określonych w większych wymiarach, tak np. dla liczby wymiarów n=13, możemy zauważyć, że zdecydowana większość punktów to wierzchołki hiperześcianu, a wraz, ze wzrostem wymiaru coraz mniejszy procent punktów leży wewnątrz kuli
(3) Klasyczne PCA nie nadaje się zawsze do wizualizacji pewnych zbiorów danych. Jeżeli dane będą określone pewnym manifoldem, musimy zastosować Kernel PCA, aby zwizualizować dane w mniejszej liczbie wymiarów
(4) Eksperyment podkreśla istotę, aby dane dla kernel PCA z kernelem cosinus miały odpowiednią strukturę (tzn. najlepiej gdy są z przedziału [-1, 1])
(5) Dla kernel = rbf, wybór parametru gamma ma ogromne znaczenie dla poprawnej wizualizacji danych