Principal Component Analysis & Kernel Trick

Bartłomiej Szałach

11 listopada 2017

1 Wstęp

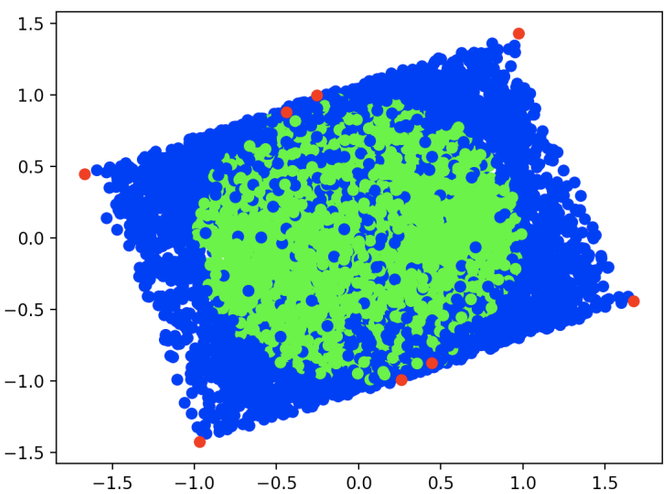
Celem zadania jest obserwacja działania metody analizy głównych składowych. Pierwsze zadanie dotyczy wykorzystania metody PCA do wykonania wizualizacji punktów znajdujących się w hiperześcianie pokolorwanych zgodnie z ich położeniem. W drugim zadaniu przygotowano dwa zbiory danych w programie graficznym, które następnie potraktowano PCA, naniesiono wektory reprezentujące główne składowe, a następnie zastosowano *kernel trick*. Do zadań użyto języka Python z bibliotekami: *sklearn, numpy, matplotlib, PIL (pillow).*

2 Punkty hiperześcianu

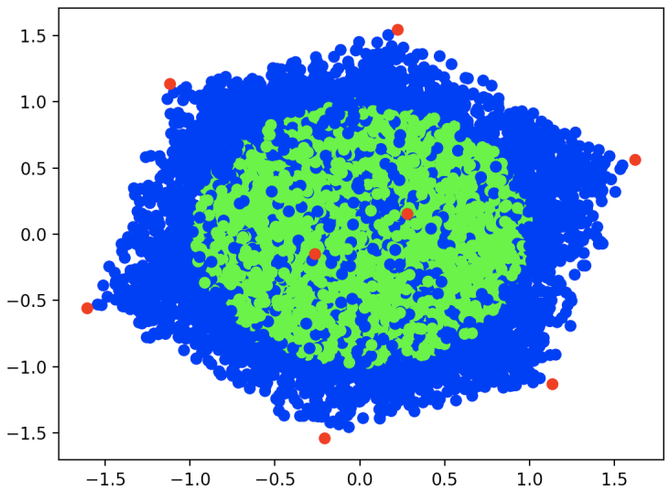
2.1 Wstęp

Hiperkulę o promieniu R wpisano w hiperześcian o długości boku 2R. Pokolorowano narożniki sześcianu na czerwono, punkty w jego wnętrzu (ale nie we wnętrzu kuli) na niebiesko, a punkty z kuli na zielono. Wykorzystano metodę PCA by wykonać wizualizację (rzut na płaszczyznę 2D) tejże sytuacji dla 3, 4, 5, 7 i 13 wymiarów.

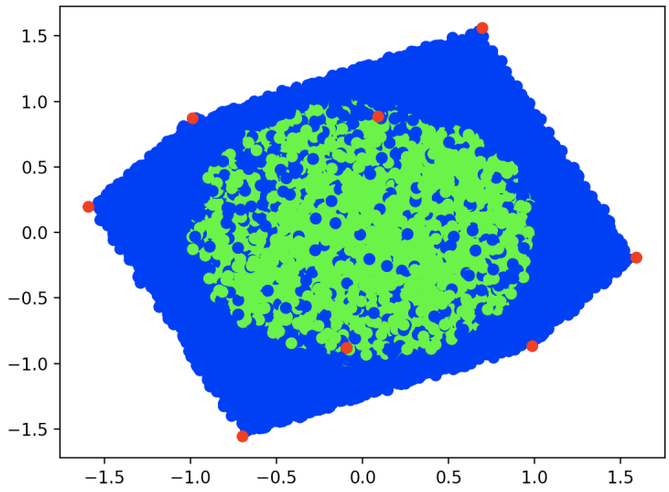
2.2 Wyniki badań



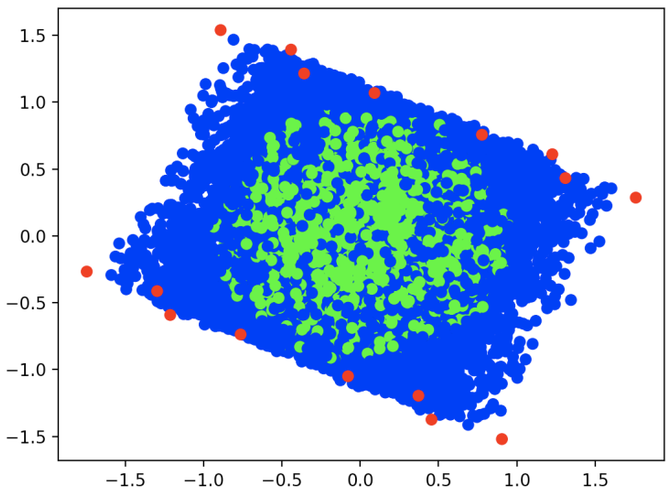
Rysunek 1: Punkty z 3 wymiarów (10k punktów)



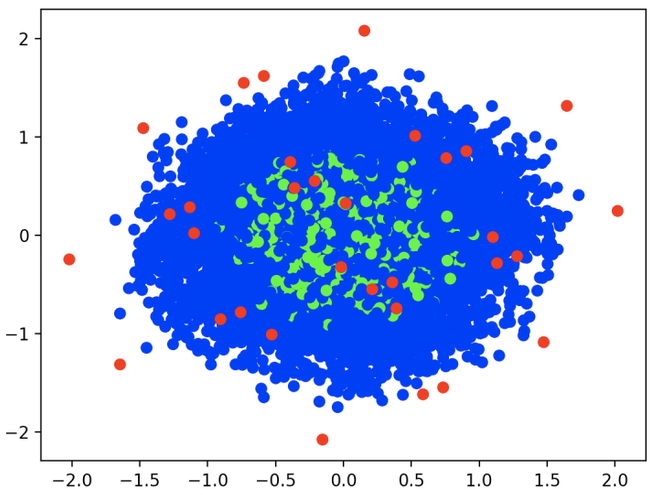
Rysunek 2: Punkty z 3 wymiarów (10k punktów)



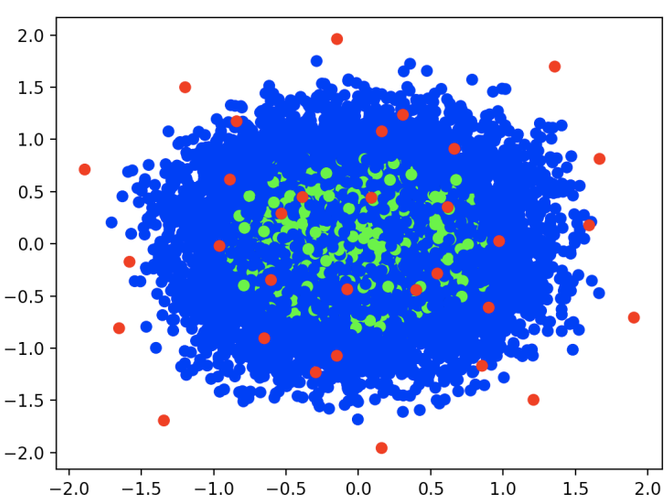
Rysunek 3: Punkty z 3 wymiarów (100k punktów)



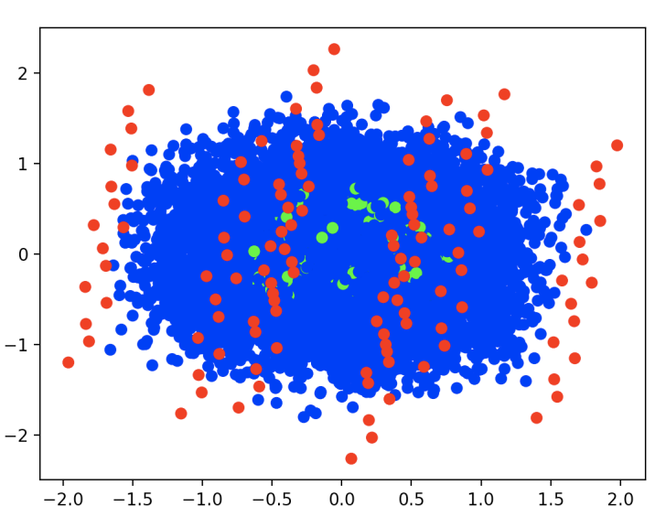
Rysunek 4: Punkty z 4 wymiarów



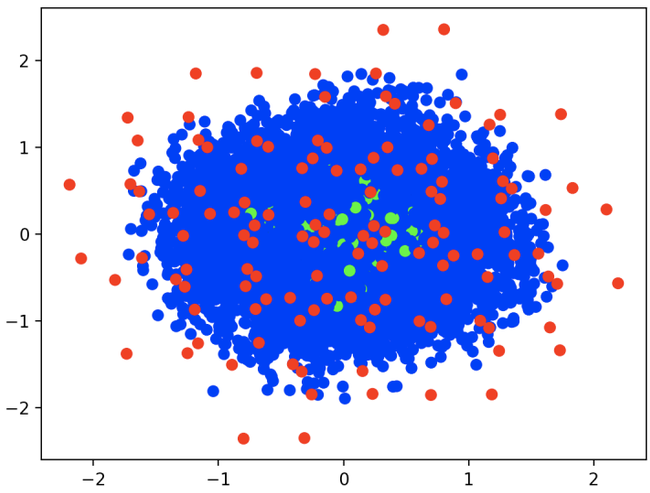
Rysunek 5: Punkty z 5 wymiarów



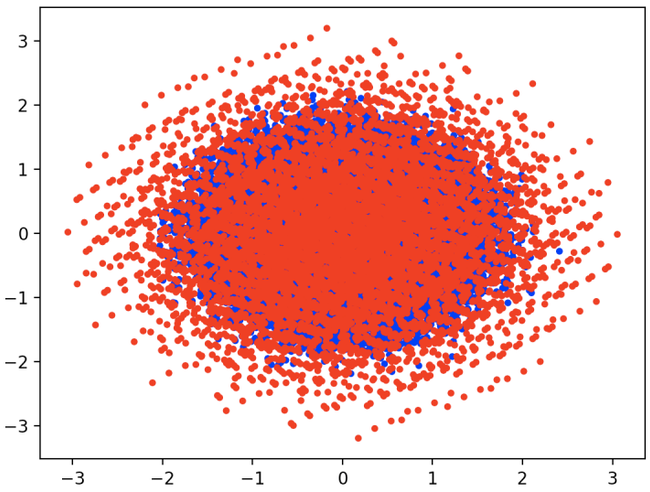
Rysunek 6: Punkty z 5 wymiarów



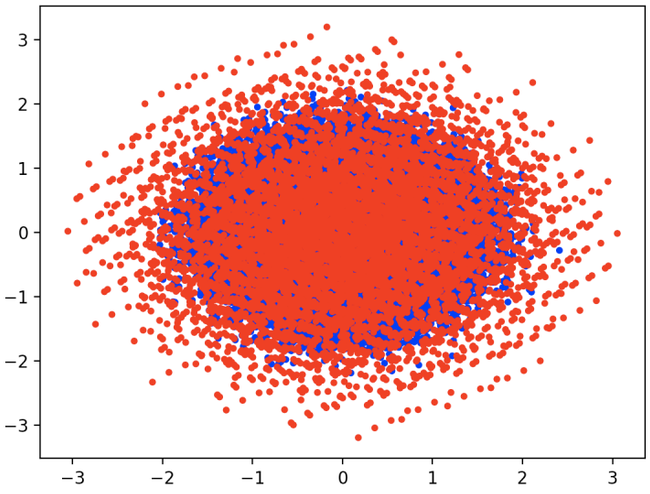
Rysunek 7: Punkty z 7 wymiarów (10k punktów)



Rysunek 8: Punkty z 7 wymiarów (10k punktów)



Rysunek 9: Punkty z 13 wymiarów (100k punktów)



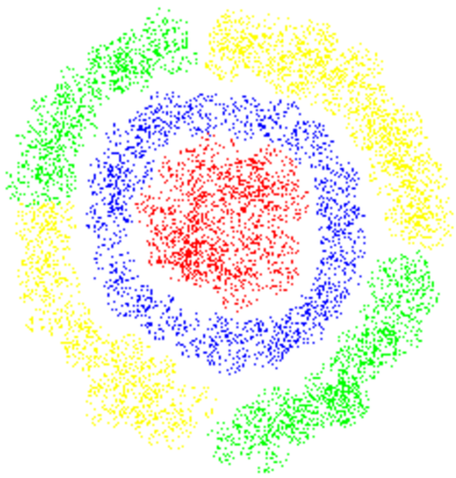
Rysunek 10: Punkty z 13 wymiarów (500k punktów)

3 Kernel PCA

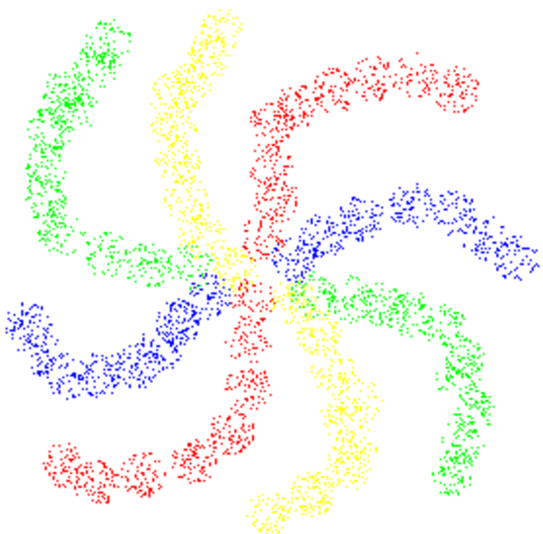
3.1 Wstęp

Przygotowano dwa zbiory danych zaprezentowane na poniższych rysunkach. Następnie, potraktowano je metodą PCA, która jednak nie daje satysfakcjonujących rezultatów (wyniki różnią się od wejściowych obserwacji bardzo nieznacznie). Aby ominąć to zjawisko stosujemy tzw. *kernel trick*, który pozwala nam "podnieść" wymiar danych, czyniąc je liniowo niezależnymi. Kernel trick używa jednego z wybranych kerneli, w tym doświadczeniu zostaną użyte kernele: **cosine** oraz **rbf** (radial basis function).

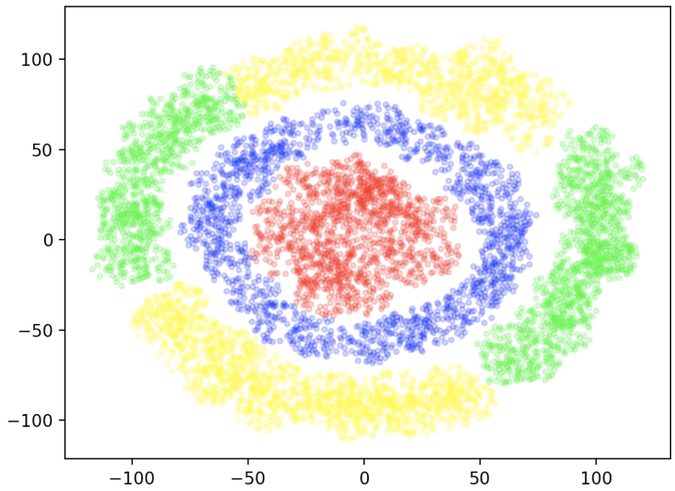
3.2 Wyniki badań



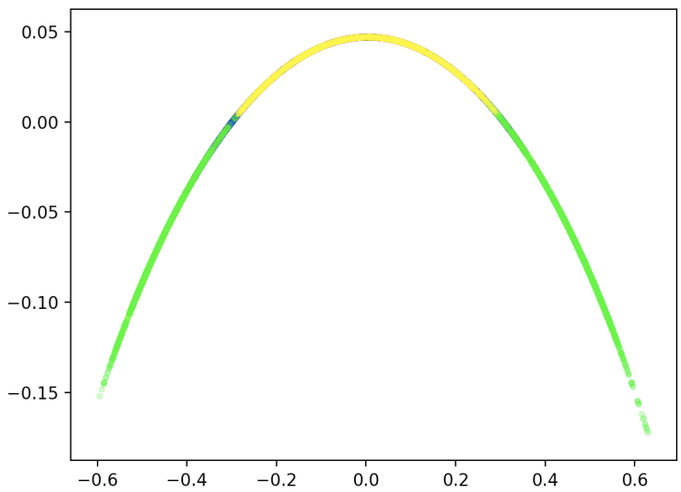
Rysunek 11: Zbiór A



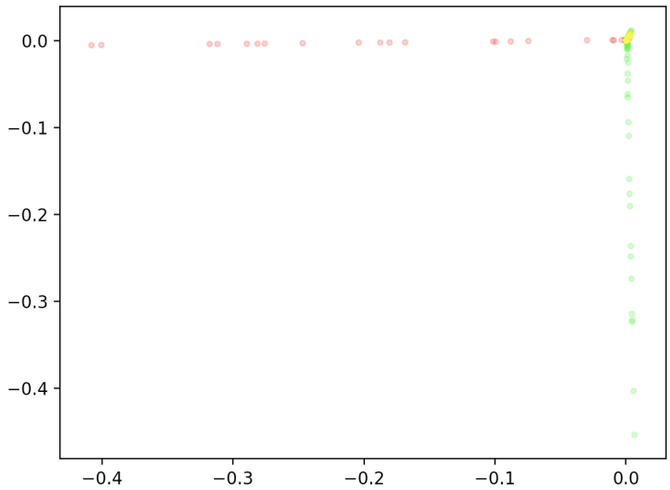
Rysunek 12: Zbiór B



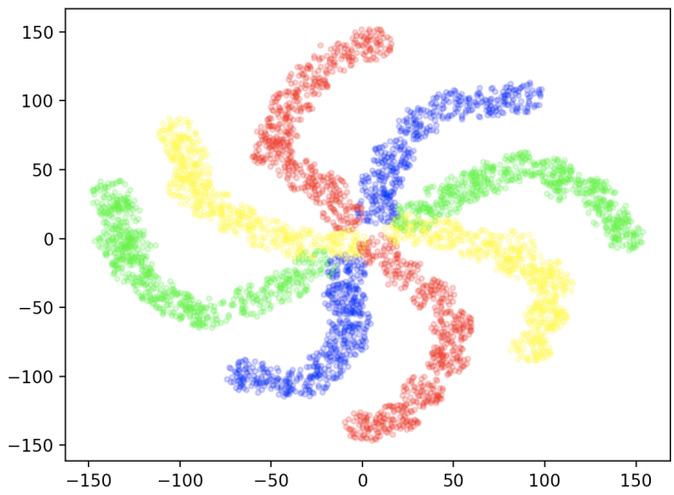
Rysunek 13: Projekcja zbioru A przy pomocy PCA



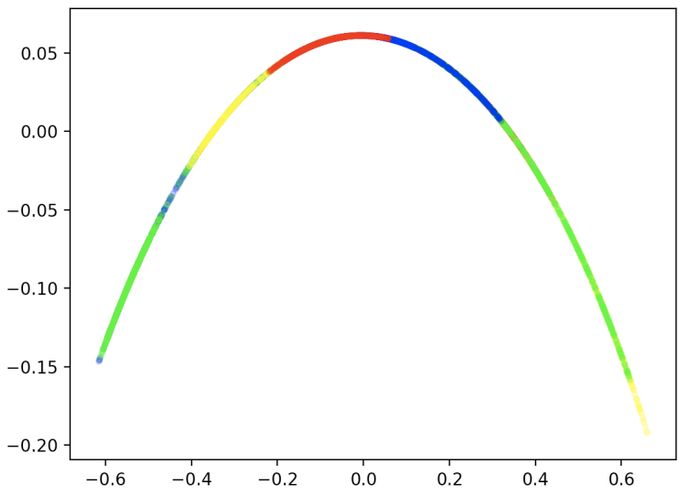
Rysunek 14: Projekcja zbioru A przy pomocy KPCA z kernelem cosine



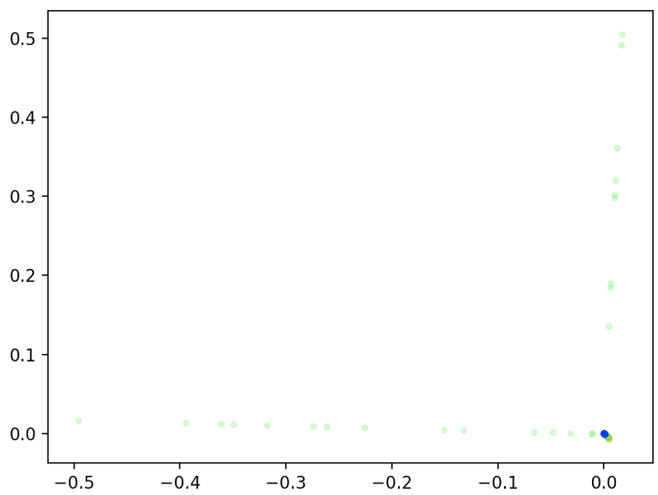
Rysunek 15: Projekcja zbioru A przy pomocy KPCA z kernelem rbf (gamma=10)



Rysunek 16: Projekcja zbioru B przy pomocy PCA



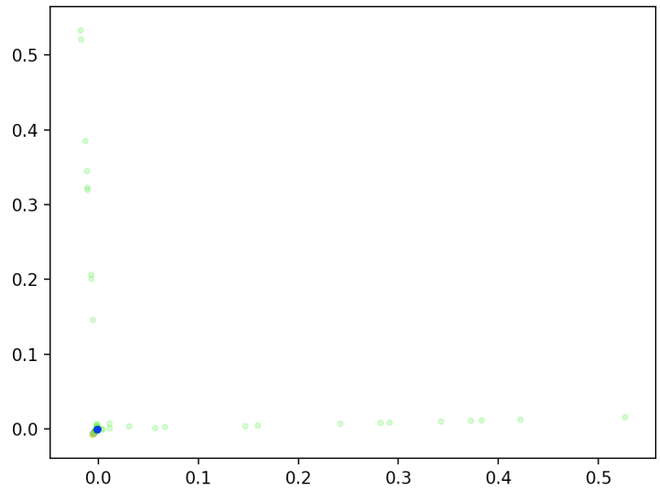
Rysunek 17: Projekcja zbioru B przy pomocy KPCA z kernelem cosine



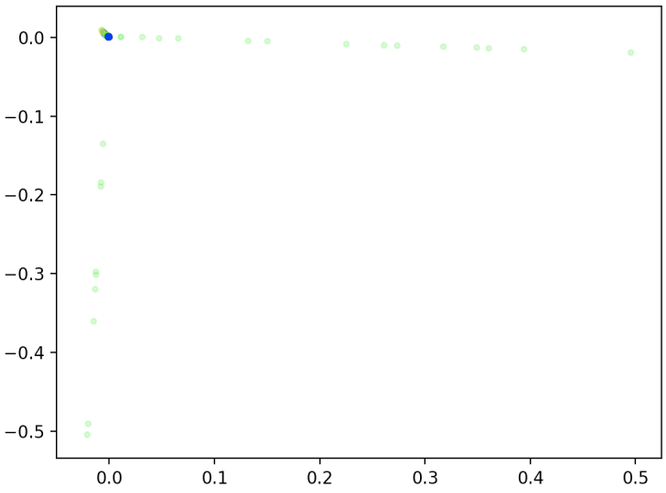
Rysunek 18: Projekcja zbioru B przy pomocy KPCA z kernelem rbf (gamma=10)

4 Wpływ parametru gamma

Metodą RBF można sterować zmieniając paramter gamma. Radialna funkcja bazowa jest funkcją rzeczywistą, której wartość zależy zwykle wyłącznie od odległości od określonego punktu. Wartość paramteru gamma określa jak bardzo oddalone punkty mają wpływ na granicę decyzyjną przy klasyfikacji. W tym ćwiczeniu radialna funkcja bazowa jest używana jako kernel trick i nie będzie omawiana. Zauważalne zmiany zachowań w zależności od parametru to na przykład rozmieszczenie punktów na osi pionowej. Dla małej wartości mamy punkty powyżej osi OY, natomiast dla dużej, wartości są poniżej OY. We wszystkich przypadkach punkty są bardzo blisko osi współrzędnych.



Rysunek 19: Projekcja zbioru B przy pomocy KPCA z kernelem rbf (gamma=3)



Rysunek 20: Projekcja zbioru B przy pomocy KPCA z kernelem rbf (gamma=30)

5 Wnioski i podsumowanie

5.1 Punkty hiperześcianu

- Po zastosowaniu PCA do punktów z przestrzeni n-wymiarowej i narysowaniu otrzymanych punktów w dwóch wymiarach, na odpowiednich wykresach zobaczyć można n-wymiarowe hiperześciany. Jest to szczególnie dobrze widoczne dla trzech wymiarów. Na rysunkach widoczny jest sześcian o wierzchołkach zaznaczonych przez czerwone punkty.

- Transformacja PCA obraca i skaluje dane, natomiast w dwóch wymiarach przypomina ona robienie zdjęcia aparatem. Dzięki temu możemy zaobserwować jak przykładowo może wyglądać tesserakt.

- Ilość wierzchołków rośnie wykładniczo i jest dana wzorem $v = 2^d$. Ponieważ w zastosowanym programie punkty były generowane wewnątrz hipersześcianu, a potem dodawane wierzchołki to dla 13 wymiarów i 10000 punktów w rzeczywistości dostajemy 10000 punktów + 8192 punkty. Z tego powodu, dla większych wymiarów potrzebne było zwiększenie ilości punktów (jak np. 100000 i 50000 dla dim = 13). Mimo tego, jak widać na rysunku nr 10 - wierzchołki i tak przykrywają resztę punktów.

- Wraz ze wzrostem wymiaru coraz więcej punktów leży poza hiperkulą.

5.2 Kernel PCA

- Dla wygenerowanych zbiorów zwykła metoda PCA nie przynosi oczekiwanych rezultatów. Należy użyć jednej z metod kernel trick.

- W metodzie Kernel PCA duże znaczenie odgrywa rodzaj używanego kernela - dostajemy zupełnie inne punkty dla *cosine* i *rbf*.

- Dla kernela *cosine* rezultaty można porównać do wykresu funkcji $f(x) = cos(x)$.