POLITECHNIKA GDAŃSKA

Monitorowanie i Diagnostyka w Systemach Sterowania

Wydział Elektrotechniki i Automatyki
Katedra Elektrotechniki, Systemów Sterowania i Informatyki
Dr hab. inż. Michał Grochowski

Monitorowanie i Diagnostyka w Systemach Sterowania

na studiach II stopnia specjalności: Systemy Sterowania i Podejmowania Decyzji

Nieliniowe PCA – kernel PCA

na podstawie:

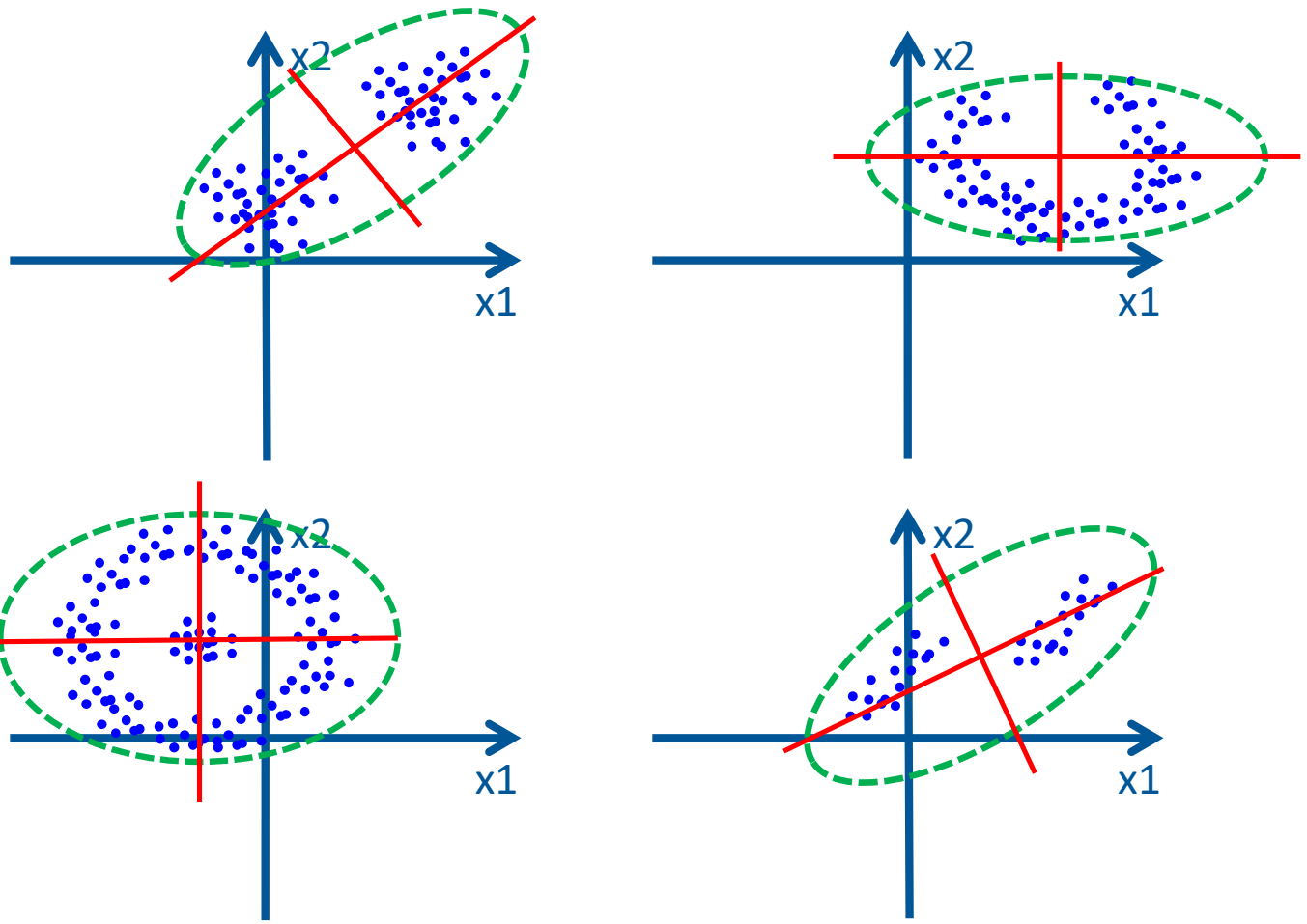
Nowicki A. *Detekcja i lokalizacja uszkodzeń przy użyciu metody Kernel PCA w aplikacji do systemów dystrybucji wody pitnej*. Praca magisterska. PG 2010 – Promotor: dr inż. M. Grochowski;

Nowicki i Grochowski. *Kernel PCA In Application to Leakage Detection In Drinking Water Distribution System*. Lecture Notes in Computer Science. ICCCI 2011, Part I, LNCS 6922, pp.497-506. Prezentacja na ICCCI 2011 - Gdynia;

Schölkopf, B., Smola, A., & Müller, K.-R. (1996). Nonlinear Component Analysis as a Kernel Eigenvalue Problem. Max-Planck-Institut fur biologische Bülthoff, Tübingen, Germany.

Opracował: dr inż. Michał Grochowski

Liniowe vs nieliniowe PCA



Czy w takich przypadkach PCA daje poprawne (wiarygodne) rezultaty ???

Liniowe vs nieliniowe PCA

Przykład:
Lokalny model sieci wodociągowej

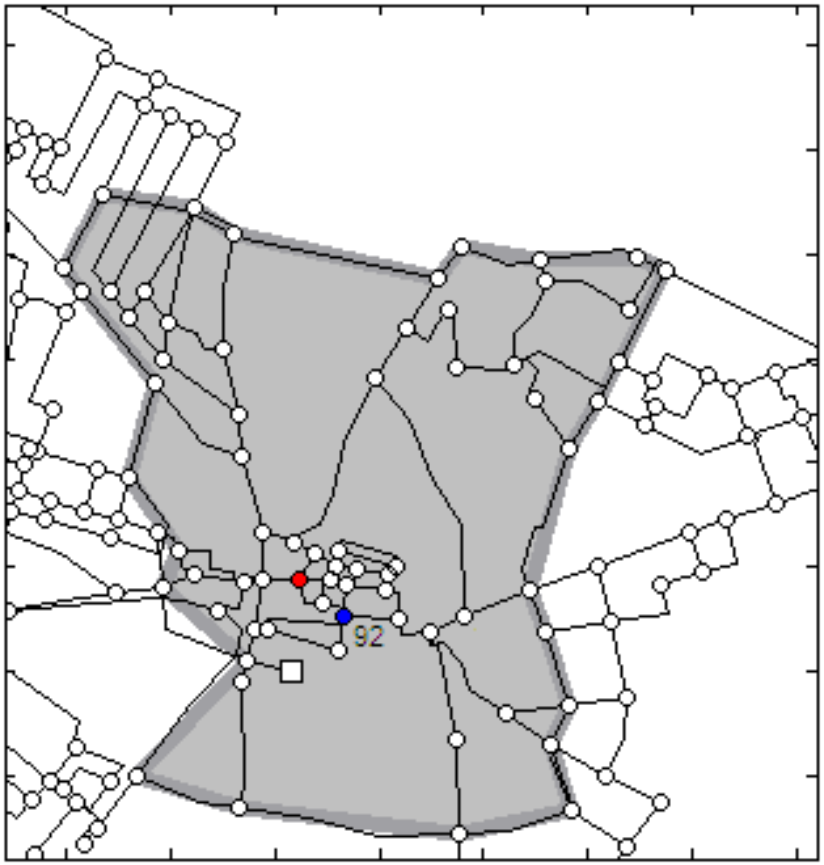
Symulacja wycieku w pobliżu węzła monitorowanego

- ciśnienie:

  P_{92}

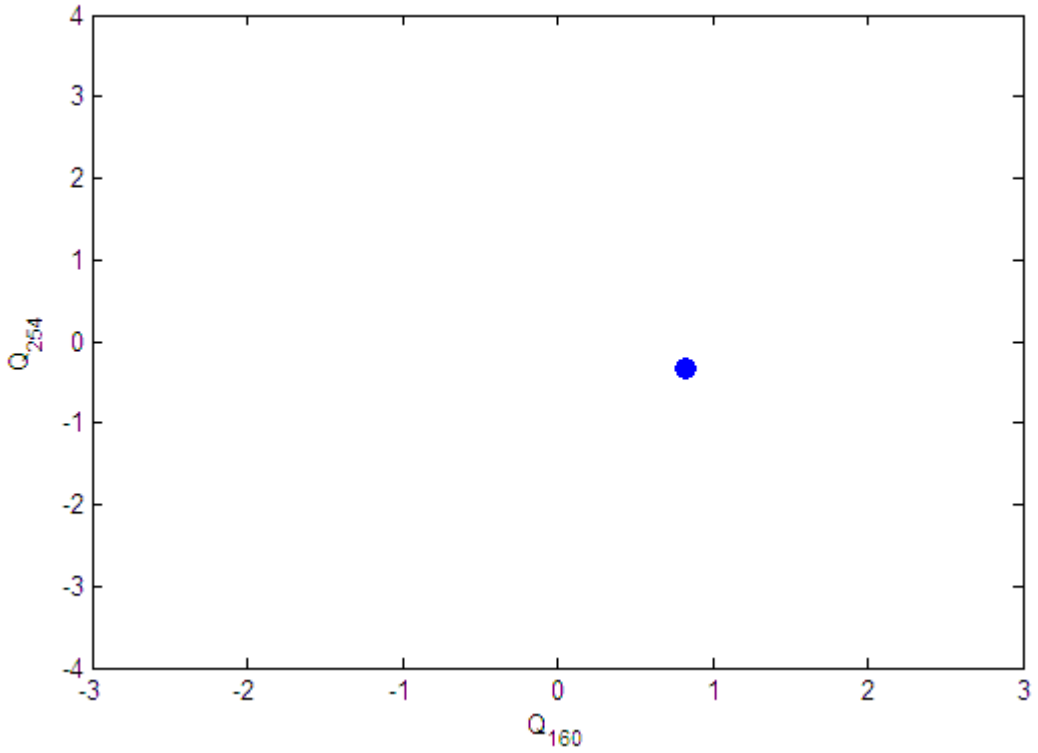
- przepływy

  Q_{144} , Q_{145} , Q_{161}



Liniowe vs nieliniowe PCA

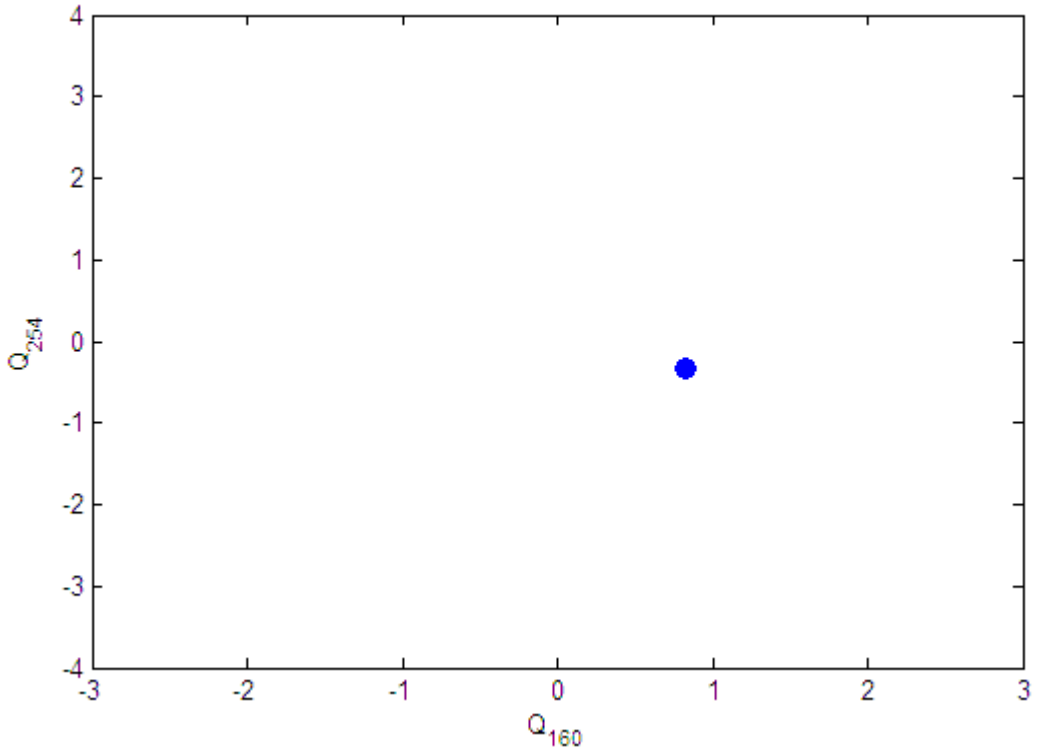
Przykład:
Lokalny model sieci wodociągowej



dzień 1 dzień 2 dzień 3 dzień 4 dzień 5 dzień 6 wyciek

Liniowe vs nieliniowe PCA

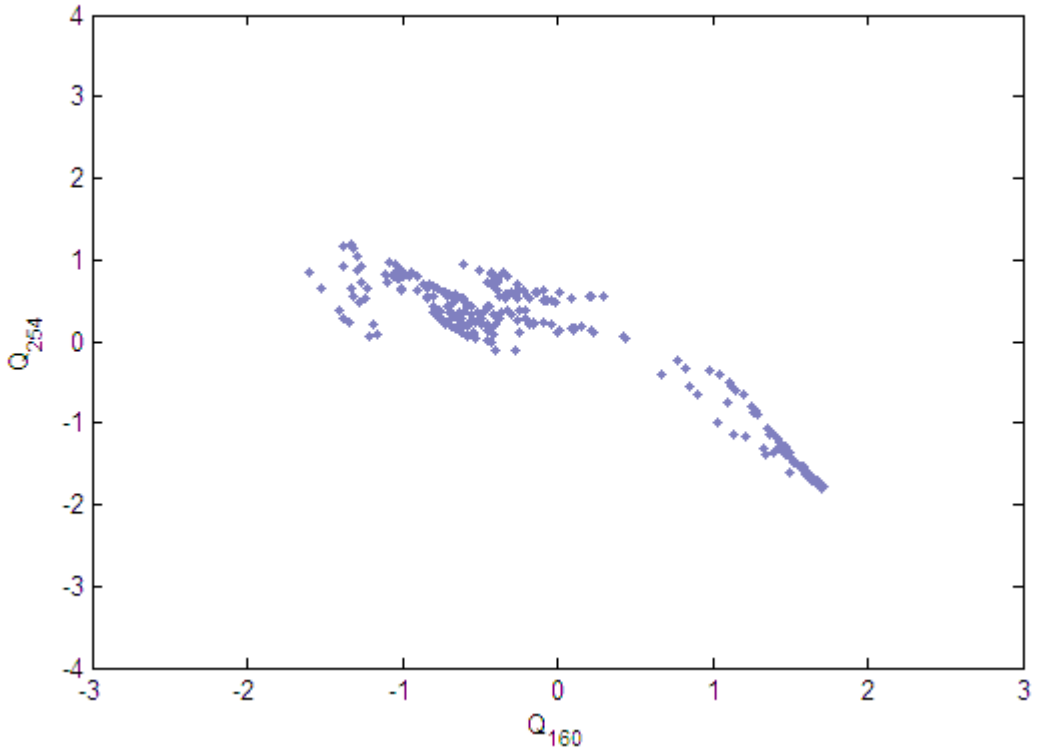
Przykład:
Lokalny model sieci wodociągowej



dzień 1 dzień 2 dzień 3 dzień 4 dzień 5 dzień 6 wyciek

Liniowe vs nieliniowe PCA

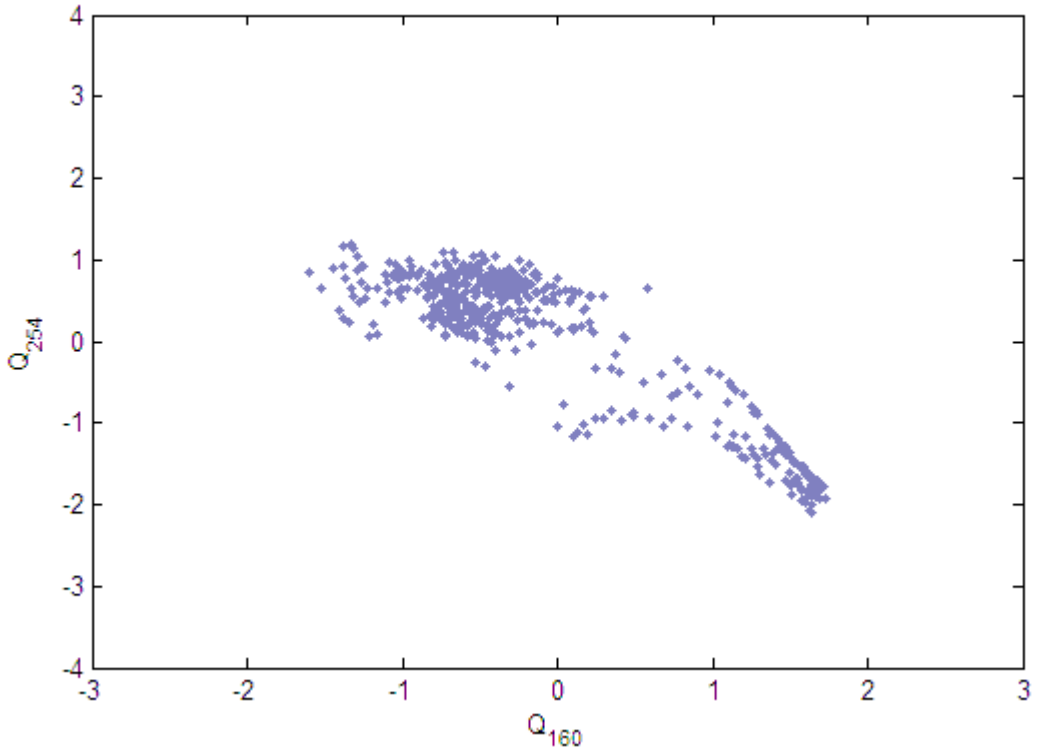
Przykład:
Lokalny model sieci wodociągowej



dzień 1 dzień 2 dzień 3 dzień 4 dzień 5 dzień 6 wyciek

Liniowe vs nieliniowe PCA

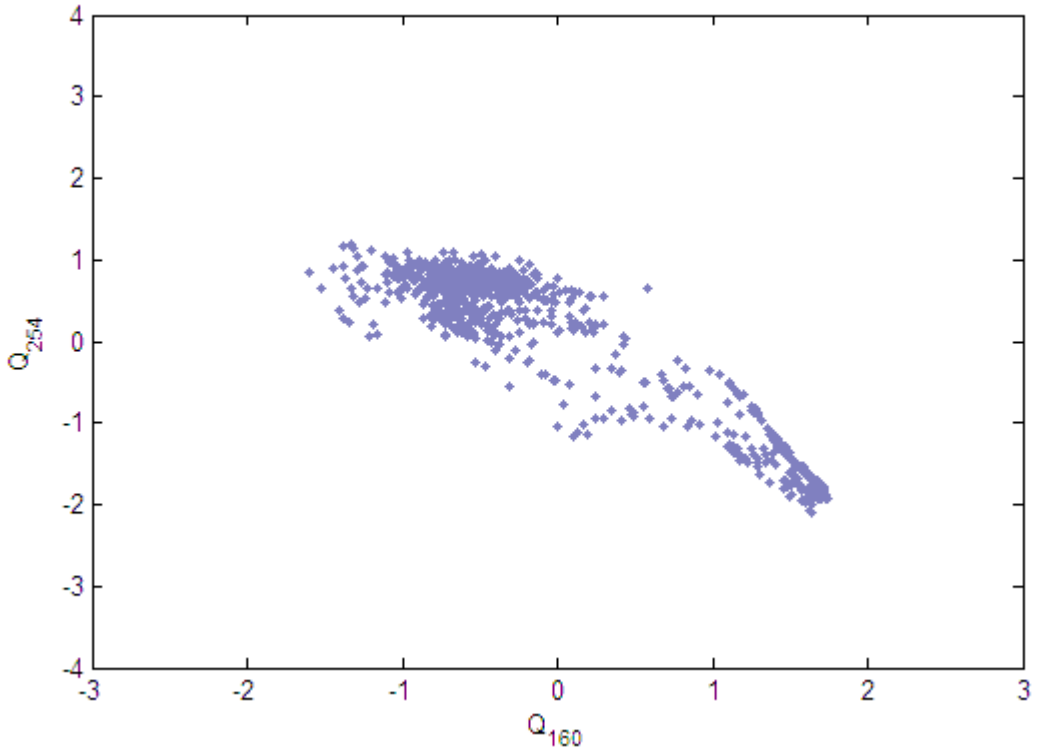
Przykład:
Lokalny model sieci wodociągowej



dzień 1 dzień 2 dzień 3 dzień 4 dzień 5 dzień 6 wyciek

Liniowe vs nieliniowe PCA

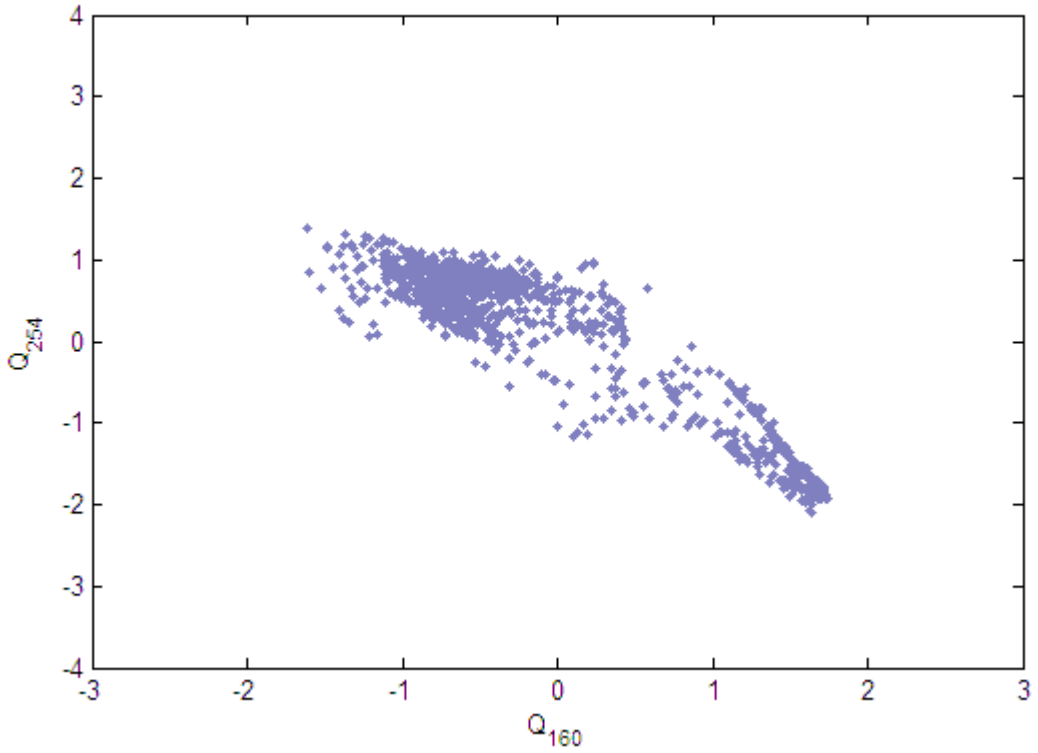
Przykład:
Lokalny model sieci wodociągowej



dzień 1 dzień 2 **dzień 3** dzień 4 dzień 5 dzień 6 wyciek

Liniowe vs nieliniowe PCA

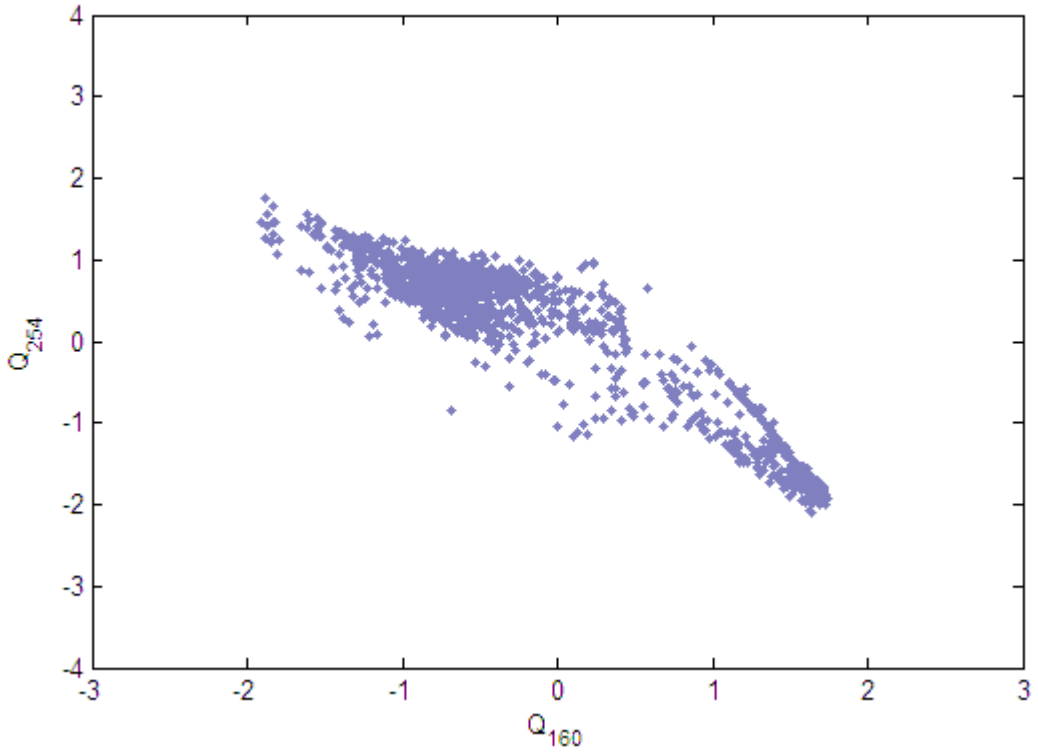
Przykład:
Lokalny model sieci wodociągowej



dzień 1 dzień 2 dzień 3 **dzień 4** dzień 5 dzień 6 wyciek

Liniowe vs nieliniowe PCA

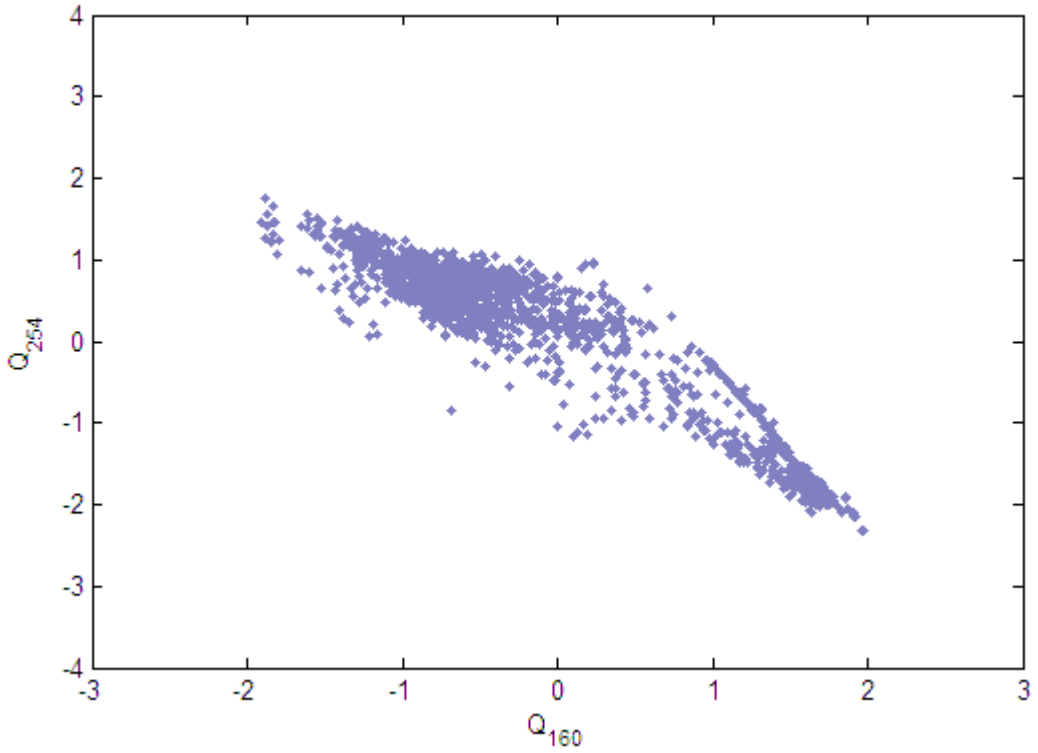
Przykład:
Lokalny model sieci wodociągowej



dzień 1 dzień 2 dzień 3 dzień 4 **dzień 5** dzień 6 wyciek

Liniowe vs nieliniowe PCA

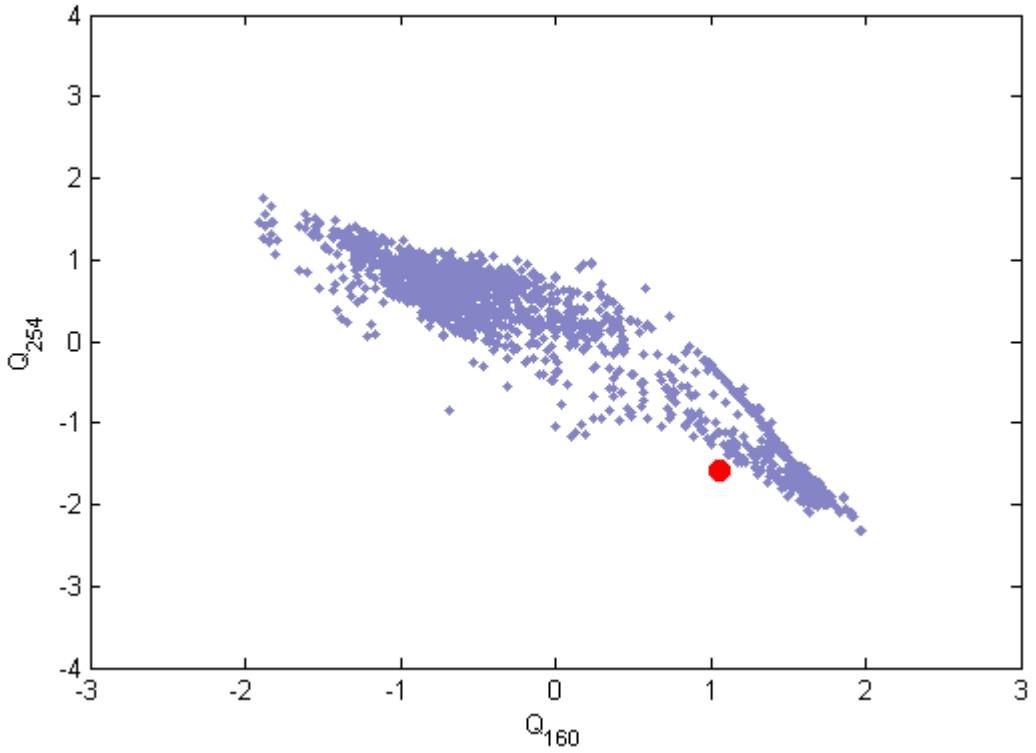
Przykład:
Lokalny model sieci wodociągowej



dzień 1 dzień 2 dzień 3 dzień 4 dzień 5 **dzień 6** wyciek

Liniowe vs nieliniowe PCA

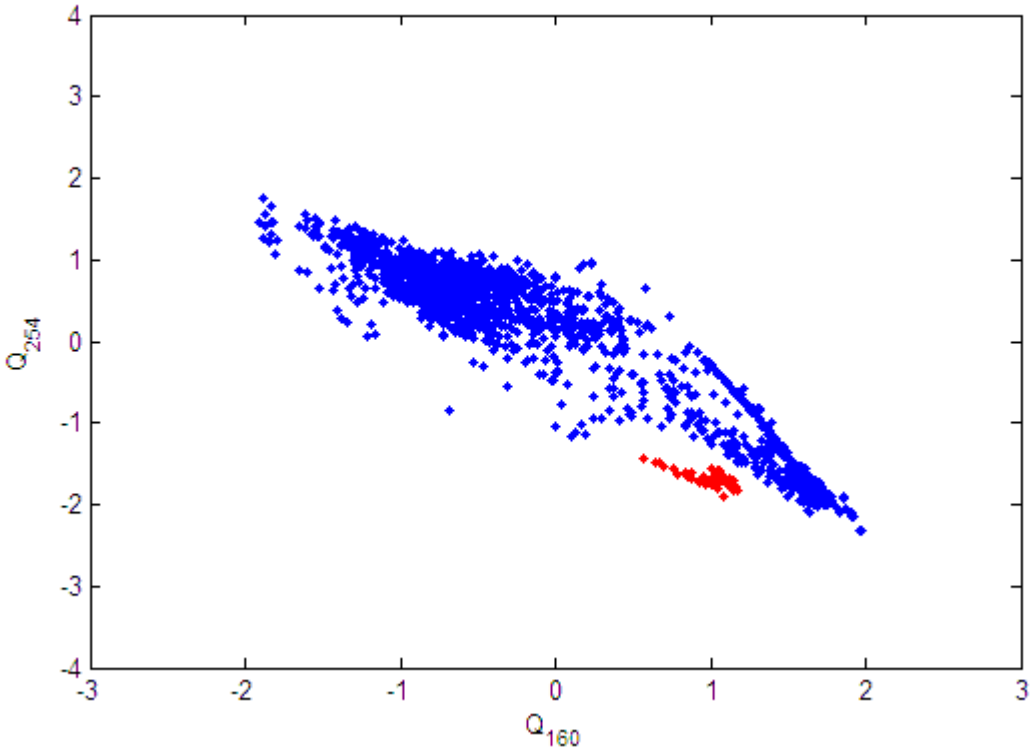
Przykład:
Lokalny model sieci wodociągowej



dzień 1 dzień 2 dzień 3 dzień 4 dzień 5 dzień 6 wyciek

Liniowe vs nieliniowe PCA

Przykład:
Lokalny model sieci wodociągowej

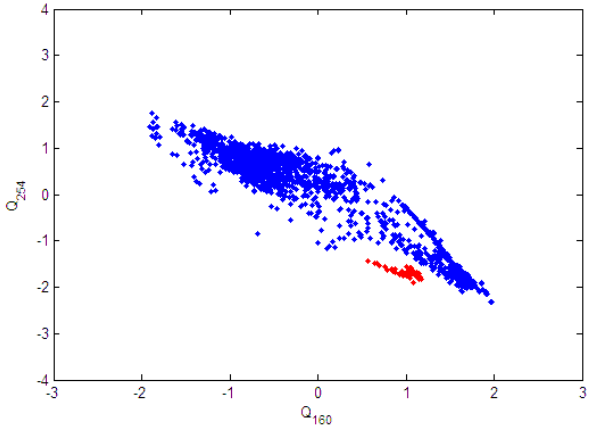


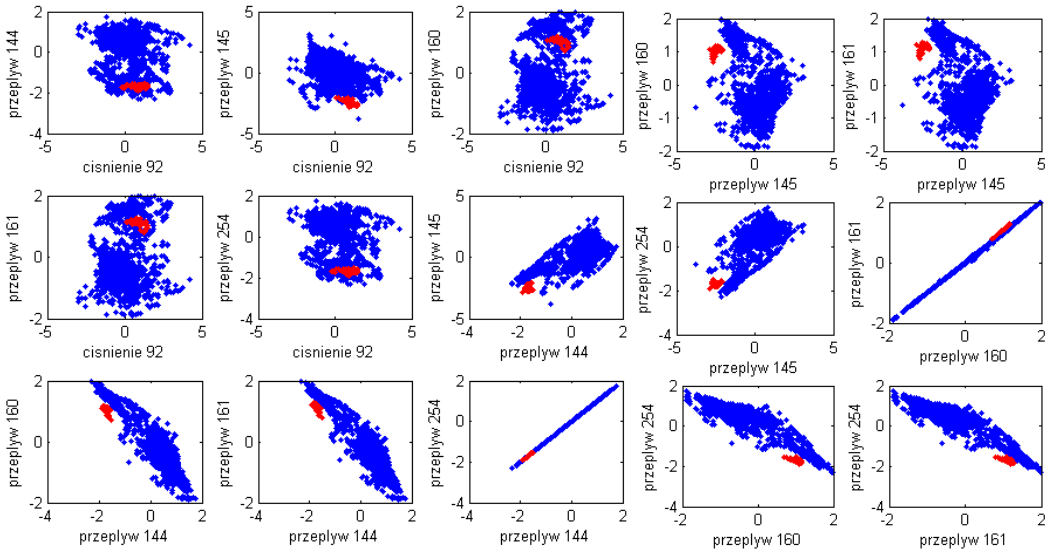
dzień 1 dzień 2 dzień 3 dzień 4 dzień 5 dzień 6 wyciek

Liniowe vs nieliniowe PCA

Przykład:
Lokalny model sieci wodociągowej

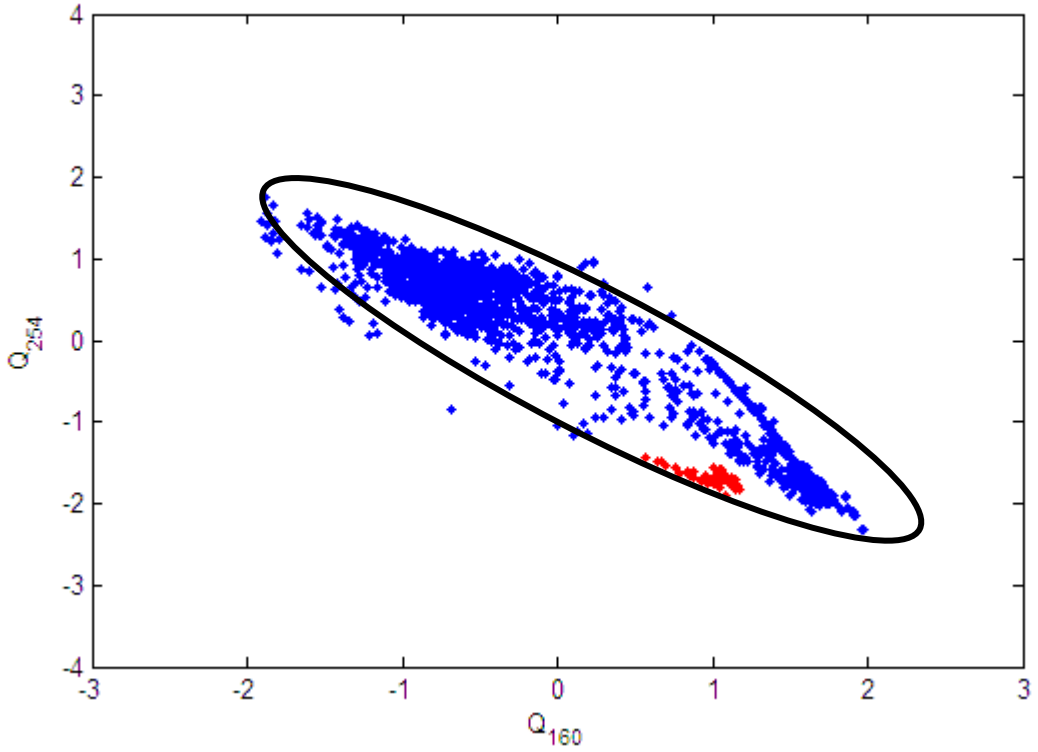
- Przypadek uproszczony:
  2 pomiary
- Przypadek rzeczywisty
  6 pomiarów





Liniowe vs nieliniowe PCA

- **Modele statystyczne:** **PCA**
  VQPCA – Vector Quantization PCA
  kPCA – kernel PCA



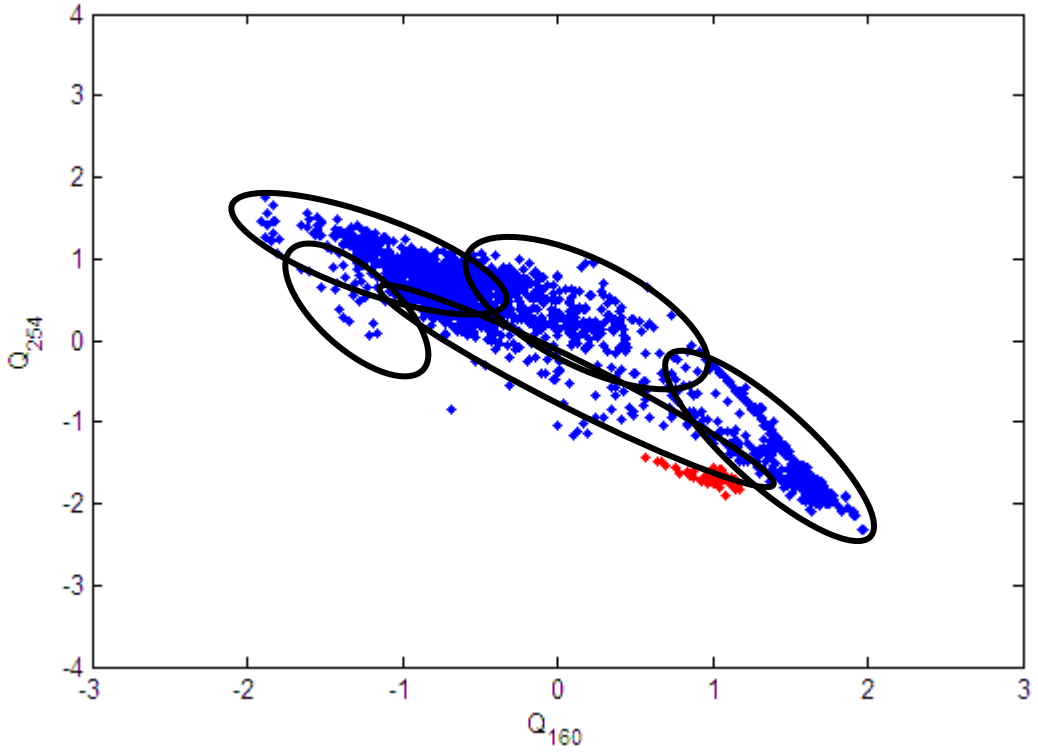
PCA

VQPCA

KPCA

Liniowe vs nieliniowe PCA

- **Modele statystyczne:** **PCA**
  VQPCA – Vector Quantization PCA
  kPCA – kernel PCA



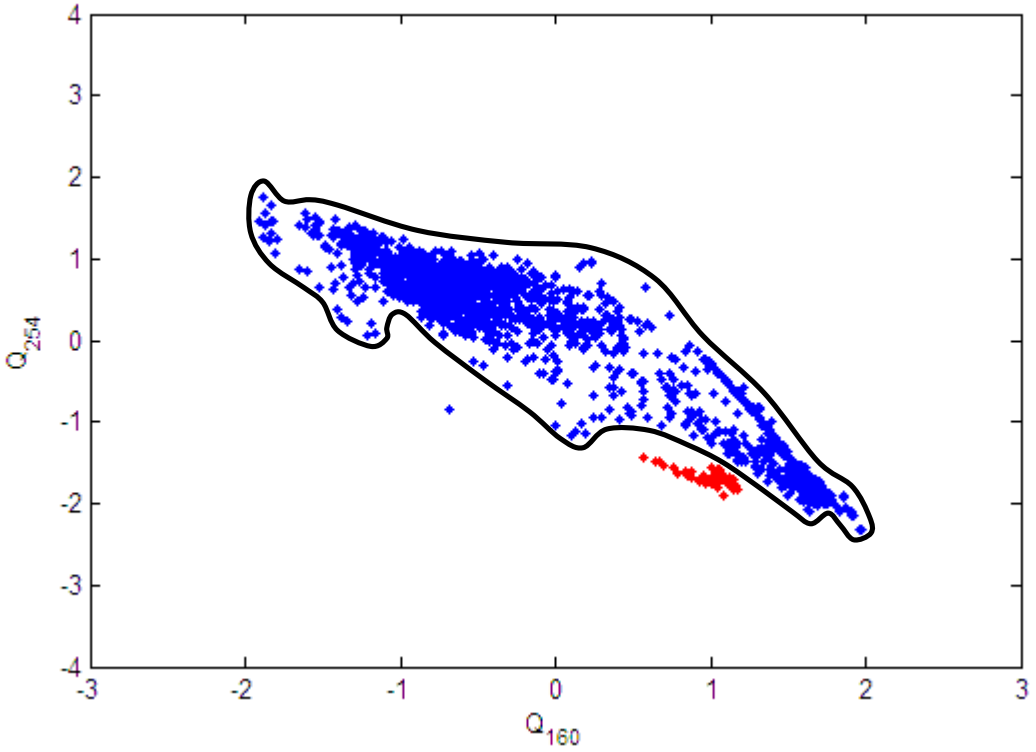
PCA

VQPCA

KPCA

Liniowe vs nieliniowe PCA

- **Modele statystyczne:** **PCA**
  VQPCA – Vector Quantization PCA
  kPCA – kernel PCA



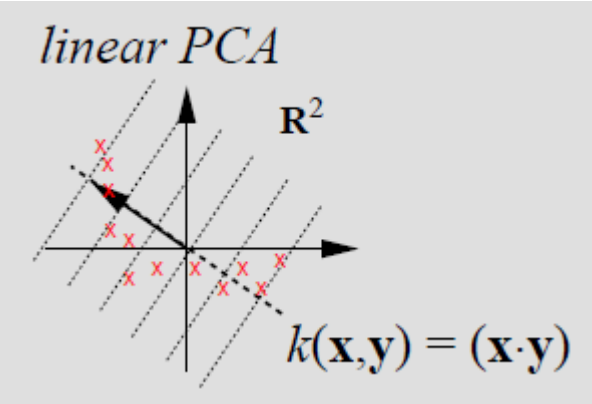
PCA

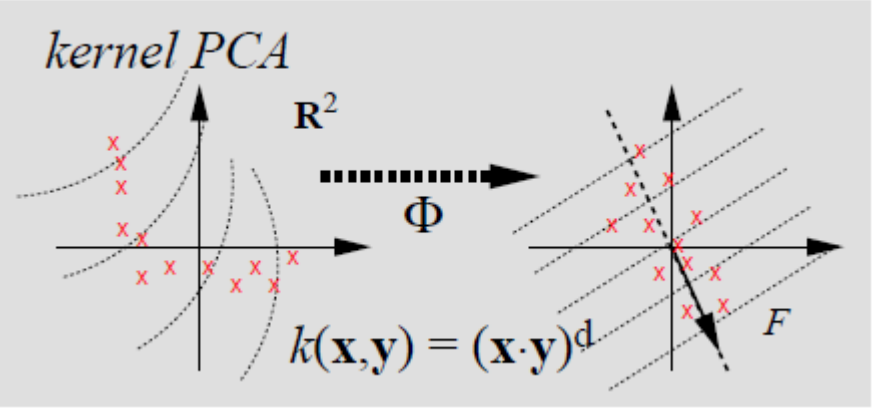
VQPCA

KPCA

Liniowe vs nieliniowe PCA

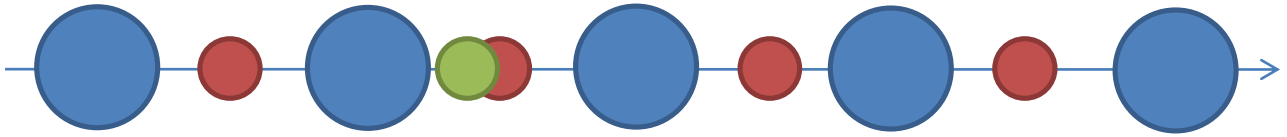
Metoda KPCA polega na odwzorowaniu *X* w rozszerzoną przestrzeń cech poprzez pewne nieliniowe odwzorowanie





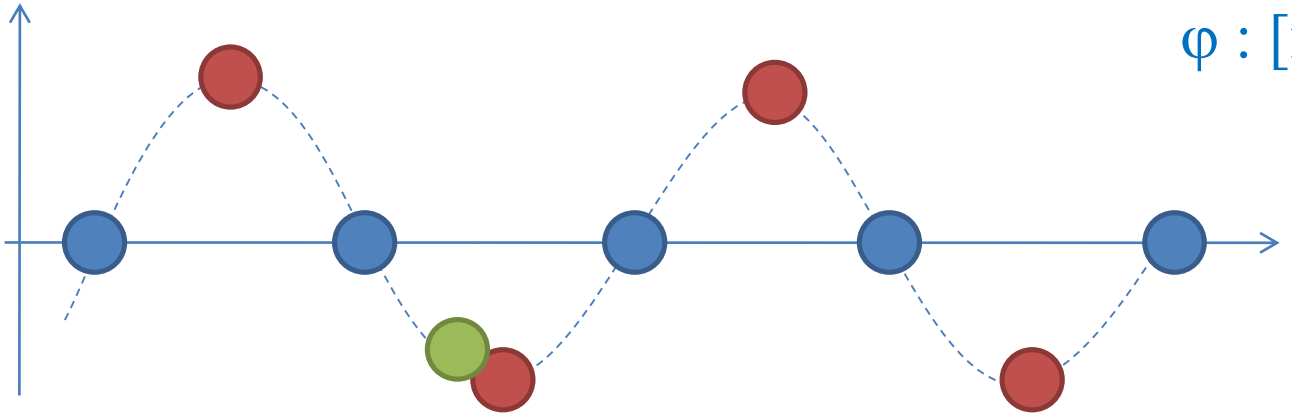
źródło: Schölkopf, B., Smola, A., & Müller, K.-R. (1996)

kernel PCA (kPCA) – opis intuicyjny metody

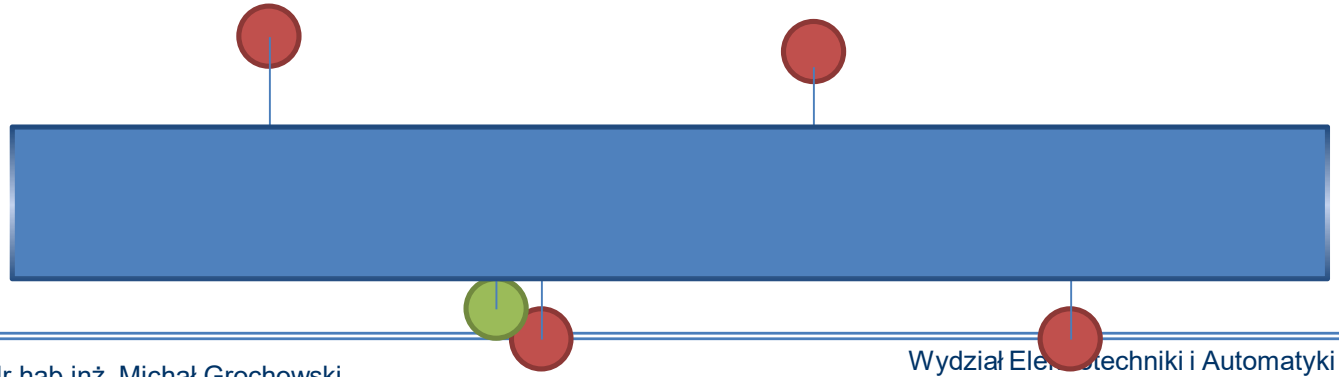


R^M Przestrzeń wejść

$\varphi : [x] \rightarrow [x, \sin(x)]$



F Przestrzeń cech



F_R Zredukowana przestrzeń cech

kernel PCA (kPCA) – opis metody

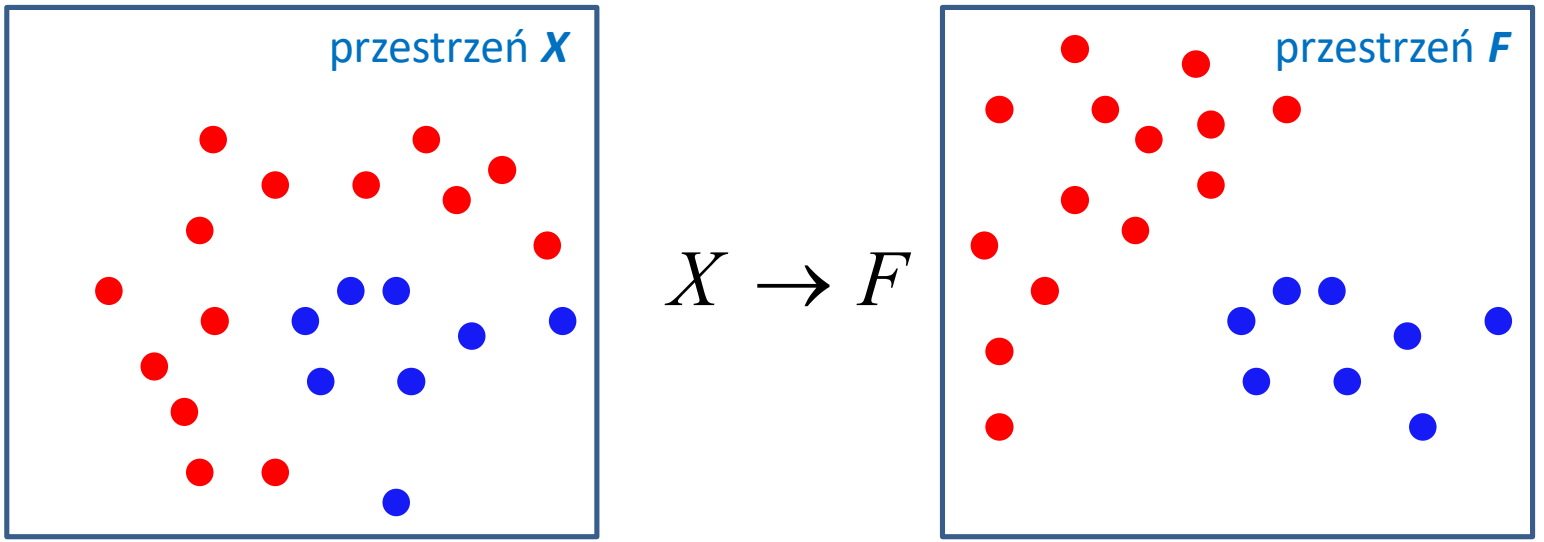
Niech:

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{1,m} & \cdots & x_{m,n} \end{bmatrix}$$

będzie zbiorem obserwacji pozostających w nieliniowej zależności.

Metoda KPCA polega na odwzorowaniu *X* w rozszerzoną przestrzeń cech poprzez pewne nieliniowe odwzorowanie

$$\Phi(\cdot) : \Re^n \rightarrow F$$

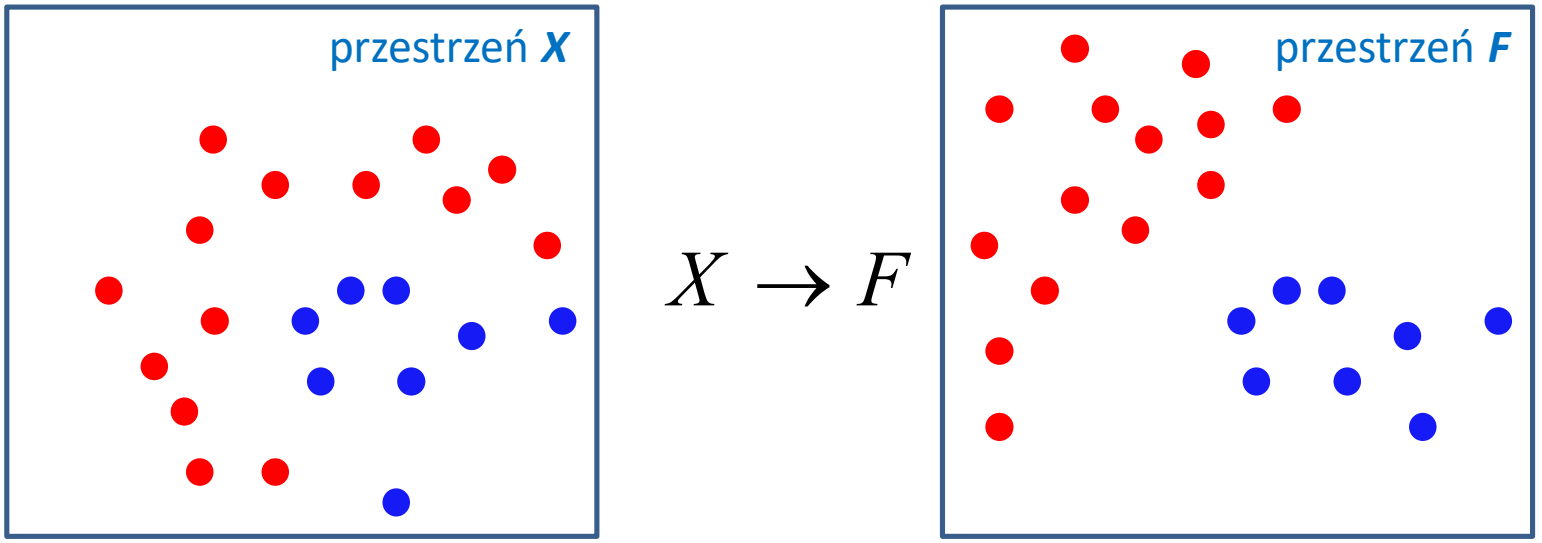


kernel PCA (kPCA) – opis metody

Wymiar przestrzeni *F* może być dowolnie duży (zwykle większy niż przestrzeń obserwacji), w ogólności również nieskończony.

Przy odpowiednio dobranym Φ, za pomocą klasycznej metody PCA daje się wyznaczyć liniowe zależności pomiędzy odwzorowanymi w *F* obserwacjami $\Phi(x_i)$.

$$\Phi(\cdot) : \Re^n \rightarrow F$$



kernel PCA (kPCA) – opis metody

Metoda PCA opiera się o wektory własne macierzy kowariancji wyznaczonej dla wektorów obserwacji *x*. Sytuacja jest oczywista jeśli znamy *x*.

W przypadku KPCA wyznaczenie $\Phi(x_i)$ w postaci jawnej, o ile jest możliwe w przypadku skończenie wymiarowej przestrzeni cech *F* (aczkolwiek bardzo kosztowne obliczeniowo dla przestrzeni o większej liczbie wymiarów), to w przypadku nieskończenie wymiarowej przestrzeni jest nierealizowalne.

Rozwiązaniem jest zastosowanie tzw. sztuczki jądra (ang. *kernel trick*) zaprezentowanej po raz pierwszy w pracy (M.A Aizerman, 1964).

kernel PCA (kPCA) – opis metody

kernel trick – „sztuczka jądra”

Metoda polega na przekształceniu algorytmu PCA w taki sposób, aby wektor Φ(x) występował tylko i wyłącznie w obrębie iloczynu skalarnego.
Wtedy iloczyn ten można zastąpić tzw. reprezentacją (funkcją) jądra $k(x_i, x_j)$:

$$\boldsymbol{K}_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j) = \langle \boldsymbol{\Phi}(\boldsymbol{x}_i), \boldsymbol{\Phi}(\boldsymbol{x}_j) \rangle$$

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{1,m} & \cdots & x_{m,n} \end{bmatrix} \qquad \boldsymbol{\Phi}: \mathcal{R}^n \rightarrow \mathcal{F} \qquad K = \begin{bmatrix} k_{1,1} & \cdots & k_{1,m} \\ \vdots & k_{i,j} & \vdots \\ k_{m,1} & \cdots & k_{m,m} \end{bmatrix}$$

Oznacza to, że możemy pracować na wektorach Φ(x) bez konieczności wykonywania odwzorowania Φ(•).

kernel PCA (kPCA) – opis metody

kernel trick – „sztuczka jądra”

Załóżmy że mamy dwa wektory w przestrzeni obserwacji X:

$$x, y \in \Re^n$$

Niech:

$$\Phi(x), \Phi(y) \in F$$

są efektem transformacji tych wektorów poprzez operator Φ(•) do przestrzeni F.

W przestrzeni F chcemy policzyć (szukamy):

$$\Phi(x)' \cdot \Phi(y)$$

Zdefiniujmy funkcję jądra (kernel):

$$k(x, y) = (\Phi(x), \Phi(y))$$

kernel PCA (kPCA) – opis metody

kernel trick – „sztuczka jądra”

x = [x1; x2]

y = [y1; y2]

Jak wygląda przestrzeń *F* ??

Φ(x)’ Φ(y) – Wynik jest skalarem!

Prosty przykład

Niech x=(x1; x2), natomiast jako nieliniową transformację Φ(·) wybieramy wielomian 2go stopnia.

$$z = \Phi(x) = \left(1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\right)$$

kernel PCA (kPCA) – opis metody

$$z = \Phi(x) = \left(1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\right)$$

kernel trick – „sztuczka jądra”

Iloczyn skalarny x,y w przestrzeni F (kernel):

$$k(x,y) = z_1 \cdot z_2 = \Phi(x) \cdot \Phi(y) = \left(1 + x_1 y_1 + x_2 y_2 + x_1^2 y_1^2 + x_2^2 y_2^2 + x_1 y_1 x_2 y_2\right)$$

Trik !!!

Czy możemy policzyć *k(x,y)* bez transformacji wektorów *x* oraz *y* ?

Wybierzmy jądro:

$$k(x,y) = \left(1 + x' y\right)^2$$

$$k(x,y) = \left(1 + x' y\right)^2 = \left(1 + x_1 y_1 + x_2 y_2\right)^2$$
$$= \left(1 + x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 y_1 x_2 y_2\right)$$

kernel PCA (kPCA) – opis metody

kernel trick – „sztuczka jądra”

$$k(x,y) = \left(1 + x_1^2 y_1^2 + x_2^2 y_2^2 + 2x_1 y_1 + 2x_2 y_2 + 2x_1 y_1 x_2 y_2\right)$$

Otrzymany wynik jest iloczynem skalarnym

… transformacji (przekształcenia):

$$\left(1, x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1 x_2\right)$$

$$\left(1, y_1^2, y_2^2, \sqrt{2}y_1, \sqrt{2}y_2, \sqrt{2}y_1 y_2\right)$$

więc wybrane *k(x,y)* może być jądrem (kernelem).

kernel PCA (kPCA) – opis metody

kernel trick – „sztuczka jądra”

Rozszerzmy nasze rozumowanie na wyższe wymiary

Niech:

$$x, y \in \Re^n \qquad \Phi(\cdot) : \Re^n \to F$$

Przestrzeń F jest m wymiarowa:

$$k(x, y) = \left(1 + x' y\right)^m$$

$$k(x, y) = \left(1 + x_1 y_1 + x_2 y_2 + \ldots + x_m y_m\right)^m$$

Można również wprowadzić współczynniki skalujące jądro *a* i *b*:

$$k(x, y) = \left(axy' + b\right)^m$$

kernel PCA (kPCA) – opis metody

kernel trick – „sztuczka jądra”

Metody wyznaczania jądra:

1. „konstruowanie – sprawdzanie”
2. Sprawdzanie warunków odpowiednich założeń matematycznych (tzw. warunki Mercera)
3. Liczenie na szczęście ☺ …

kernel PCA (kPCA) – opis metody

kernel trick – „sztuczka jądra”

Warunki Mercera:

Funkcja K(x,y) może stanowić funkcję jądra, jeżeli:

1. Jest symetryczna, np.: $k(x,y) = x*y=y*x$

2. Macierz:

$$K = \begin{bmatrix} k_{1,1} & \cdots & k_{1,m} \\ \vdots & k_{i,j} & \vdots \\ k_{m,1} & \cdots & k_{m,m} \end{bmatrix} \qquad k_{ij}=k(x_i,x_j)$$

jest półdodatnio określona dla dowolnych wektorów x, y.

kernel PCA (kPCA) – opis metody

kernel trick – „sztuczka jądra”

Wybór funkcji jądra jest istotny ponieważ determinuje odwzorowanie Φ(•).

Najczęściej spotykane funkcje jądra to:

- wielomianowa: $k(x,y) = (x^T y)^d$, przy $d = const$
- Gaussa: $k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\delta^2}\right)$, przy $\delta > 0$
- odwrotnej formy kwadratowej: $k(x,y) = \frac{1}{\sqrt{\|x-y\|^2+c}}$, przy $c = const, c > 0$
- sigmoidalna: $k(x,y) = tanh(\gamma x^T y + \Theta)$, przy $\gamma, \Theta = const$

Szczególnym przypadkiem funkcji wielomianowej jest funkcja wielomianowa pierwszego rzędu i odpowiada ona przekształceniu liniowemu: $\Phi(\cdot) : \Re^n \to \Re^n$

Taka funkcja jądra (*k(x,y) = x*y)* sprowadza KPCA do klasycznej metody PCA.

kernel PCA (kPCA) – opis metody

Wektory własne $\Phi(x_i)$ znajdują się zawsze w podprzestrzeni przestrzeni cech *F* rozpiętej przez wektory.

Tak więc wymiar macierzy kowariancji (a więc i ilość wektorów własnych) jest określony przez liczbę obserwacji, a nie ich wymiar !!!
Oznacza to, że w ogólności KPCA umożliwia wyznaczenie znacznie większej ilości składników głównych, niż PCA.

Przy dużej ilości obserwacji wyznaczenie wszystkich wektorów własnych może być kłopotliwe, ale istnieją metody aproksymacji jedynie istotnych wektorów własnych, tj. takich którym odpowiadają duże wartości własne (np. metoda potęgowa).

Budowa modelu kPCA - algorytm

- Podobnie jak przy PCA zapisujemy zbiór obserwacji w postaci macierzy X zawierającej N obserwacji, każda po M zmiennych;

Kolejne zmienne

$$X = \begin{bmatrix} x_{1,1} & \cdots & x_{1,n} \\ \vdots & \ddots & \vdots \\ x_{1,m} & \cdots & x_{m,n} \end{bmatrix}$$

Kolejne pomiary

każdy element $x_{i,j}$ stanowi *j-tą* zmienną *i-tej* obserwacji.

Traktując kompletną obserwację jako wektor x_i możemy zapisać:

$$X = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

n – ilość zmiennych pomiarowych;
m - ilość danych pomiarowych;

- Podobnie jak przy PCA normalizujemy dane;

Budowa modelu kPCA - algorytm

- Wyznaczamy macierz K (ang. *kernel matrix*), w której każdy element k_{ij} przyjmuje wartości odpowiadające iloczynowi skalarnemu obserwacji $\Phi(x_i)$ i $\Phi(x_j)$ odwzorowanych w przestrzeni cech *F*.

Do jej obliczenia wykorzystujemy wybraną funkcję jądra:

$$K = \begin{bmatrix} k_{1,1} & \cdots & k_{1,m} \\ \vdots & k_{i,j} & \vdots \\ k_{m,1} & \cdots & k_{m,m} \end{bmatrix} \qquad k_{ij}=k(x_i,x_j)$$

Macierz *K* stanowi jądro przekształcenia. Dla jądra będącego funkcją Gaussa, możemy zapisać:

$$k_{ij} = \exp\left(-\frac{\|x_i - x_j\|^2}{2\delta^2}\right)$$

Z czego wynika, że macierz jest symetryczna, a wszystkie elementy na przekątnej są sobie równe.
Wykorzystanie tych faktów umożliwia redukcję obliczeń.

Budowa modelu kPCA - algorytm

- Dokonujemy przesunięcia obserwacji $\Phi(x_i)$ w przestrzeni cech tak, aby ich środek znajdował się w początku układu współrzędnych.

Znormalizowane obserwacje oznaczamy jako: $\tilde{\Phi}(x_i)$

$$\tilde{\Phi}(x_i) = \Phi(x_i) - \Phi_0$$

gdzie:

$\tilde{\Phi}(x_i)$ - obraz obserwacji x_i w przestrzeni cech *F* uwzględniający przesunięcie środka zestawu danych do początku układu współrzędnych;

Φ_0 - środek zbioru N obserwacji

$$\Phi_0 = \frac{1}{m}\sum_{i=1}^{m}\Phi(x_i)$$

Budowa modelu kPCA - algorytm

Ponieważ nie znamy odwzorowanych punktów $\Phi(x_i)$ dokonujemy tego pośrednio przekształcając kolejno każdy element macierzy K:

$$\tilde{k}_{ij} = k_{ij} - \frac{1}{m}\sum_{r=1}^{m} k_{ir} - \frac{1}{m}\sum_{r=1}^{m} k_{rj} + \frac{1}{m^2}\sum_{r,s=1}^{m} k_{rs}$$

gdzie:

$\tilde{k}_{ij}$ - elementy macierzy $\tilde{K}$ zawierającej iloczyny skalarne przesunięte do początku układu współrzędnych w przestrzeni cech

$$\tilde{K} = \begin{bmatrix} \tilde{k}_{1,1} & \cdots & \tilde{k}_{1,M} \\ \vdots & \ddots & \vdots \\ \tilde{k}_{N,1} & \cdots & \tilde{k}_{N,M} \end{bmatrix}$$

Budowa modelu kPCA - algorytm

- Wyznaczamy wektory własne macierzy $\tilde{K}$ tak jak opisano w przypadku PCA lub korzystając z metod iteracyjnych. Wyznaczone wektory własne α pozostają w zależności względem wektorów własnych *V* macierzy korelacji przekształconych obserwacji Φ(x) jak to opisuje zależność:

$$v_i = \sum_{i=1}^{m} \alpha_i \, \Phi(x_i)$$

gdzie: α_i – i ty wektor własny macierzy $\tilde{K}$

- Wektor α należy tak przeskalować, aby spełniony był warunek:

$$\|v_i\| = 1 \Leftrightarrow \|\alpha_i\|^2 = \frac{1}{\lambda_i}$$

gdzie: λ_i – i ta wartość własna macierzy $\tilde{K}$

Budowa modelu kPCA - algorytm

- Analogicznie do PCA zgrupowane w macierz przeładowań *V* istotne wektory własne v_i umożliwiają wyznaczenie współrzędnych dowolnego punktu x_T względem składników głównych w przestrzeni cech.

$$x_T^{KPCA} = V\,\Phi(x_T)$$

Ponieważ nie znamy $\Phi(x_T)$ i nie możemy wyznaczyć v_i bez znajomości $\Phi(x_i)$, możemy ponownie skorzystać ze sztuczki jądra i otrzymamy zależność wykorzystującą iloczyn skalarny, który jesteśmy w stanie obliczyć:

$$V\,\Phi(x_T) = \sum_{i=1}^{L} \alpha_i \left(\Phi(x_i) \cdot \Phi(x_T)\right)$$

Budowa modelu kPCA - algorytm

Należy jeszcze uwzględnić przesunięcie do początku układy współrzędnych i ostatecznie dowolny punkt x_T można przedstawić w przeładowanej przestrzeni korzystając z równania:

$$x_T^{KPCA} = K'\alpha - \sum_{i=1}^{L} \alpha_i \left(\frac{1}{m} \sum_{j=1}^{m} k_j - \frac{1}{m^2} \sum_{r,s=1}^{m} K_{rs} \right) - \left(\frac{1}{m} \sum_{r=1}^{m} K_{ir} \right) \alpha$$

gdzie: K' – macierz iloczynów skalarnych;

$$K = \left[k_1^{'}, \ldots, k_L^{'} \right]$$

$$k_i^{'} = \left(\Phi(x_i) \cdot \Phi(x_T) \right) = k\left(\Phi(x_i), \Phi(x_T) \right)$$

Monitoring procesu

Jako miarę jakości wpasowania bieżących obserwacji w model wykorzystamy błąd rekonstrukcji *E* w przestrzeni cech *F* opisany równaniem

$$E(\tilde{\Phi}) = (\tilde{\Phi} \cdot \tilde{\Phi}) - (\mathrm{V}\tilde{\Phi} \cdot \mathrm{V}\tilde{\Phi})$$

gdzie: $\tilde{\Phi}$ – skrócony zapis:

$$\tilde{\Phi}(x_T) = \Phi(x_T) - \Phi_0$$

Interpretacją geometryczną bledu rekonstrukcji jest kwadrat odległości pomiędzy odwzorowaną w *F* obserwacją *x*, a jej rzutem na L- istotnych składników głównych.

Wyrażenie $\tilde{\Phi} \cdot \tilde{\Phi}$ określa sferyczne pole potencjału w przestrzeni cech i może być wyznaczone jako:

$$\tilde{\Phi} \cdot \tilde{\Phi} = k(x_T, x_T) - \frac{2}{L}\sum_{i=1}^{L} k(x_T, x_i) + \frac{1}{L^2}\sum_{i,j=1}^{L} K_{ij}$$

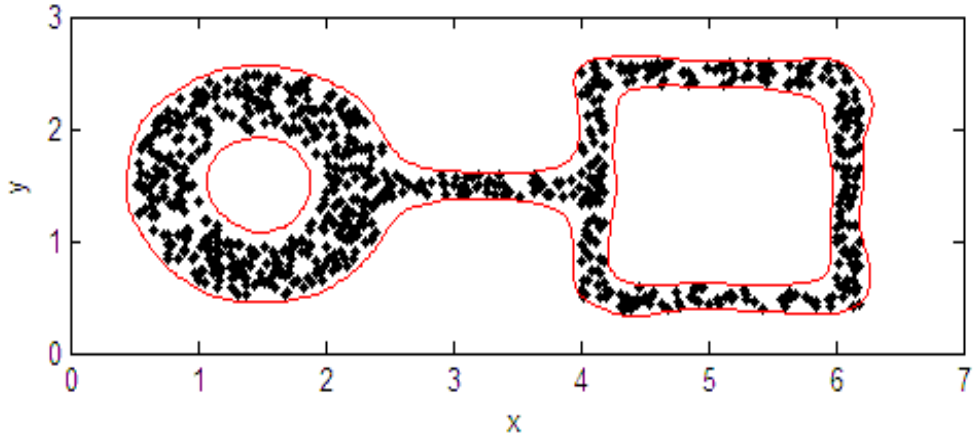
Monitoring procesu

Ponieważ nie znamy $\tilde{\Phi}$, wykorzystując poprzednie wzory możemy wyznaczyć wyrażenie na błąd rekonstrukcji w przestrzeni cech *F* w zależności od punktu pracy w przestrzeni wejść R^M:

$$E(x_T) = k(x_T, x_T) - \frac{2}{L}\sum_{i=1}^{L} k(x_T, x_i) + \frac{1}{L^2}\sum_{i,j=1}^{L} K_{ij} - x_T^{KPCA}\left(x_T^{KPCA}\right)^T$$

Jedna z hiperpłaszczyzn w przestrzeni cech *F*, o stałej wartości błędu rekonstrukcji może pełnić rolę granicy decyzyjnej determinującej dopuszczalność aktualnego stanu operacyjnego (Hoffmann, 2006).
Sedno sztuczki polega na tym, że taka hiperpłaszczyzna w przestrzeni cech, na skutek nieliniowego przekształcenia Φ(•), w przestrzeni wejść przybiera postać hiperpowierzchni o regularnych kształtach dopasowanych do danych treningowych.
W przypadku PCA błąd rekonstrukcji nie był ograniczony od góry, natomiast przy KPCA, na skutek nieliniowego odwzorowania Φ(•), błąd rekonstrukcji w przestrzeni wejść zbiega do pewnej granicznej wartości.

Monitoring procesu



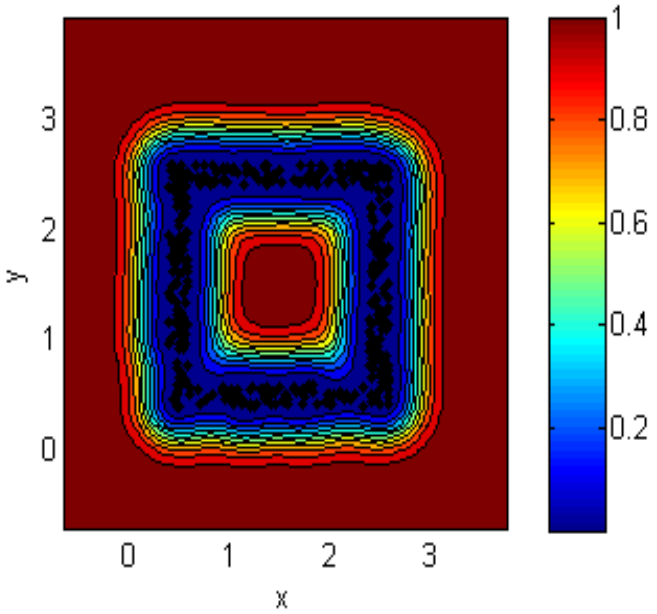
Przypadek zbioru dwuwymiarowych obserwacji. Przy odpowiednio dobranych parametrach metoda KPCA w połączeniu z błędem rekonstrukcji pełniącym rolę wskaźnika jakości procesu potrafią skutecznie opisać nawet złożone nieliniowe modele.
Punkty symbolizują obserwacje, krzywa jest izolinią stałego błędu rekonstrukcji.

Autor eksperymentu: Heiko Hoffman.

kernel PCA – analiza parametrów

Błąd rekonstrukcji

Do wnioskowania o bieżącym stanie systemu wykorzystujemy, błąd rekonstrukcji. Dla poprawnie stworzonego modelu błąd ten będzie bliski zeru w przestrzeni stanów dozwolonych, a wysoki dla pozostałych stanów; wybierając wartość błędu, która zapewnia zadawalający margines niepewności, określamy odpowiadającą jej izolinie, która pełnić będzie rolę granicy decyzyjnej.



Błąd rekonstrukcji - czarne punkty oznaczają obserwacje, kolorem zaznaczono obszary o równej wartości błędu rekonstrukcji.

kernel PCA – analiza parametrów

Funkcje jądra i parametry

Przy braku jakichkolwiek przesłanek co do zastosowania innej funkcji sugeruje się zastosowanie funkcji Gaussa.

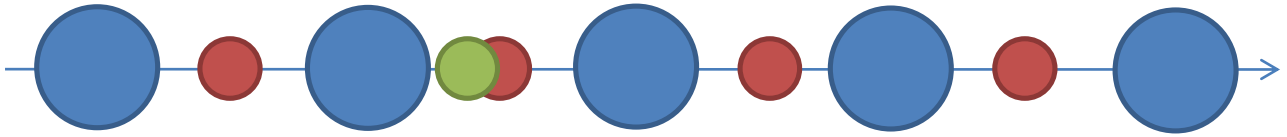
$$k(x,y) = \exp\left(-\frac{\|x-y\|^2}{2\delta^2}\right), \text{ przy } \delta > 0$$

Szerokość jądra δ pełni istotną rolę z punktu widzenia skuteczności metody i powinna być uważnie dobrana.

Zbyt duża wartość δ spowoduje, że eksponenta będzie się zachowywała prawie liniowo i wielowymiarowe odwzorowania zaczną tracić swoje nieliniowe właściwości.

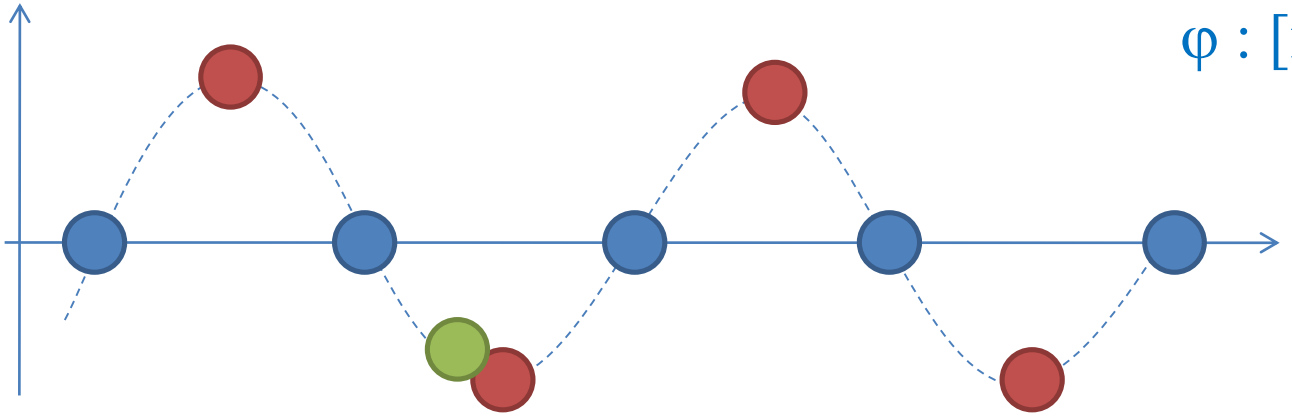
Z drugiej strony, za mała wartość doprowadzi do nieregularnej granicy decyzyjnej, zwiększając podatność na zakłócenia (Souza, 2010).

kernel PCA (kPCA) – opis intuicyjny metody

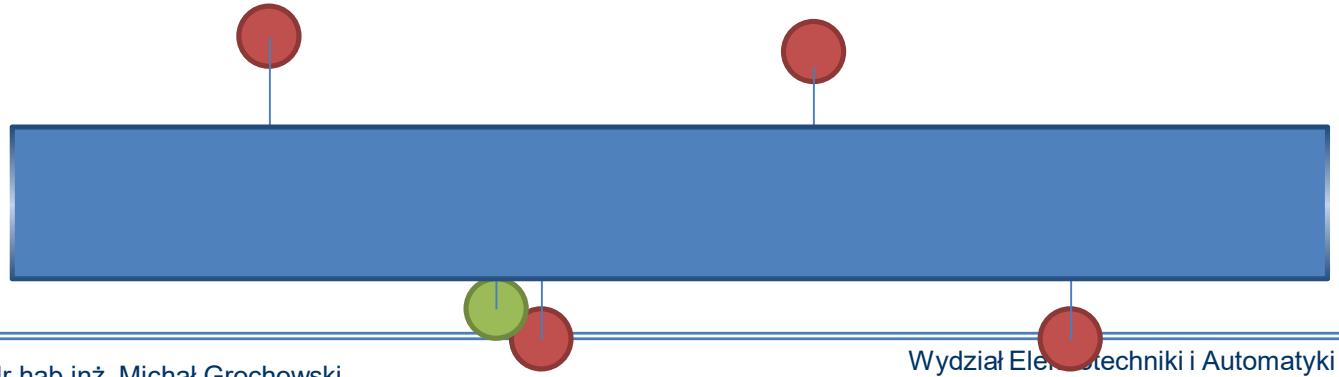


R^M Przestrzeń wejść

$\varphi : [x] \rightarrow [x, \sin(x)]$



F Przestrzeń cech



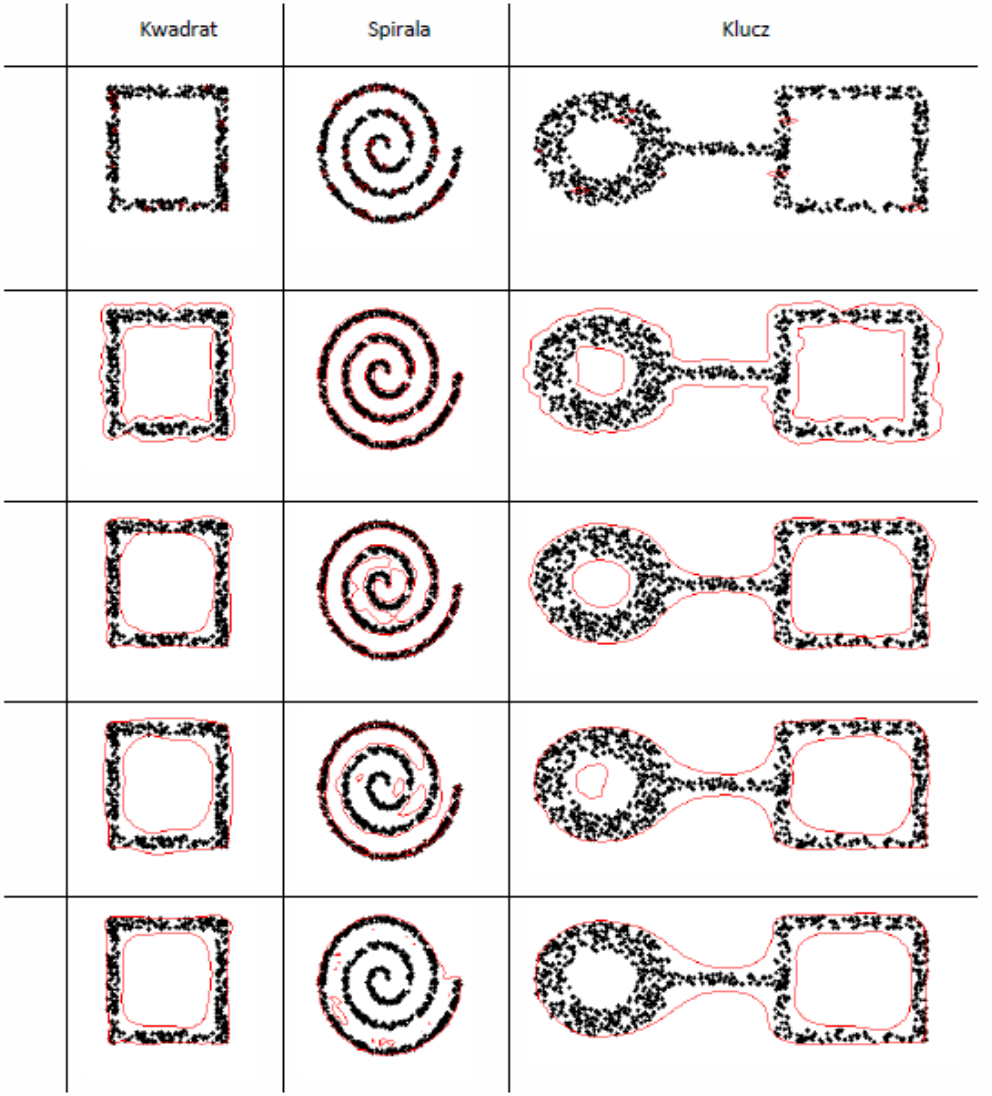
F_R Zredukowana przestrzeń cech

kernel PCA (kPCA) – opis intuicyjny metody

- ~~Odwzorowanie wejść $\Phi : X \rightarrow F$~~
- Funkcja jądra: miara podobieństwa między wektorami wejściowymi
- ~~Wyznaczenie podprzestrzeni $Fs \subset F$~~
- Wymiar Fs
- ~~Wyszukanie liniowych zależności~~
- PCA
- Wykorzystanie błędu rekonstrukcji jako granicy decyzyjnej
- Graniczna wartość błędu rekonstrukcji

kernel PCA – analiza parametrów

szerokość jądra δ



kernel PCA – analiza parametrów

Liczba składników głównych

Liczba istotnych składników głównych *L*, powinna być dobierana w zależności od złożoności modelu procesu reprezentowanego przez obserwacje.

Generalnie złożone procesy wymagają zwykle uwzględnienia większej liczby składników głównych, chociaż nie istnieją szczegółowe wytyczne w tej kwestii.

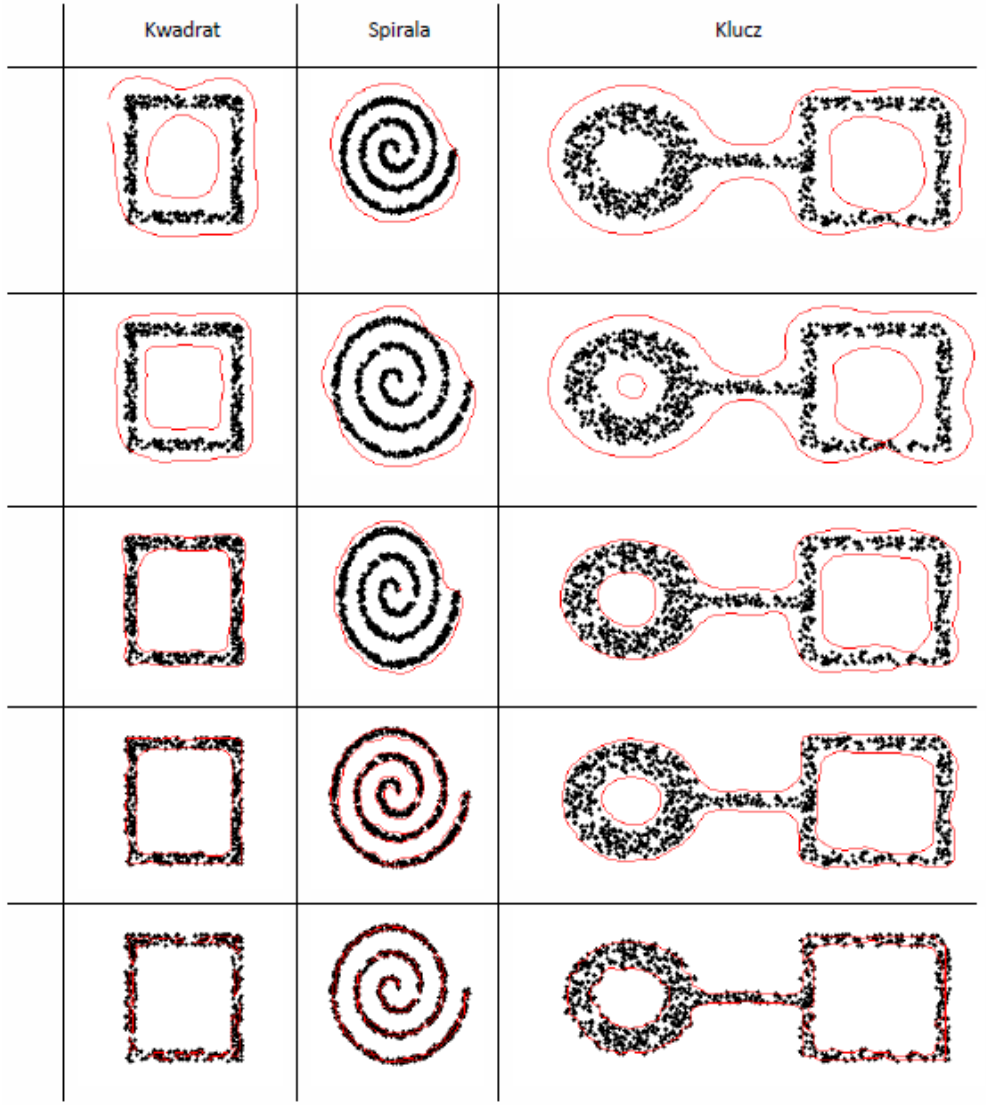
Zbyt mała ilość składników głównych prowadzi do znacznego uproszczenia modelu i w efekcie może doprowadzić do zbyt luźnej granicy decyzyjnej.

Natomiast zbyt duża liczba składników głównych prowadzi do zjawiska znanego w sieciach neuronowych jako przeuczenie. W efekcie system traci zdolność do generalizowania.

Ponieważ składniki główne znajdują się w podprzestrzeni wyznaczonej przez przekształcone wektory własne $\Phi(x_i)$, liczba istotnych składników głównych może być co najwyżej równa liczbie obserwacji x_i.

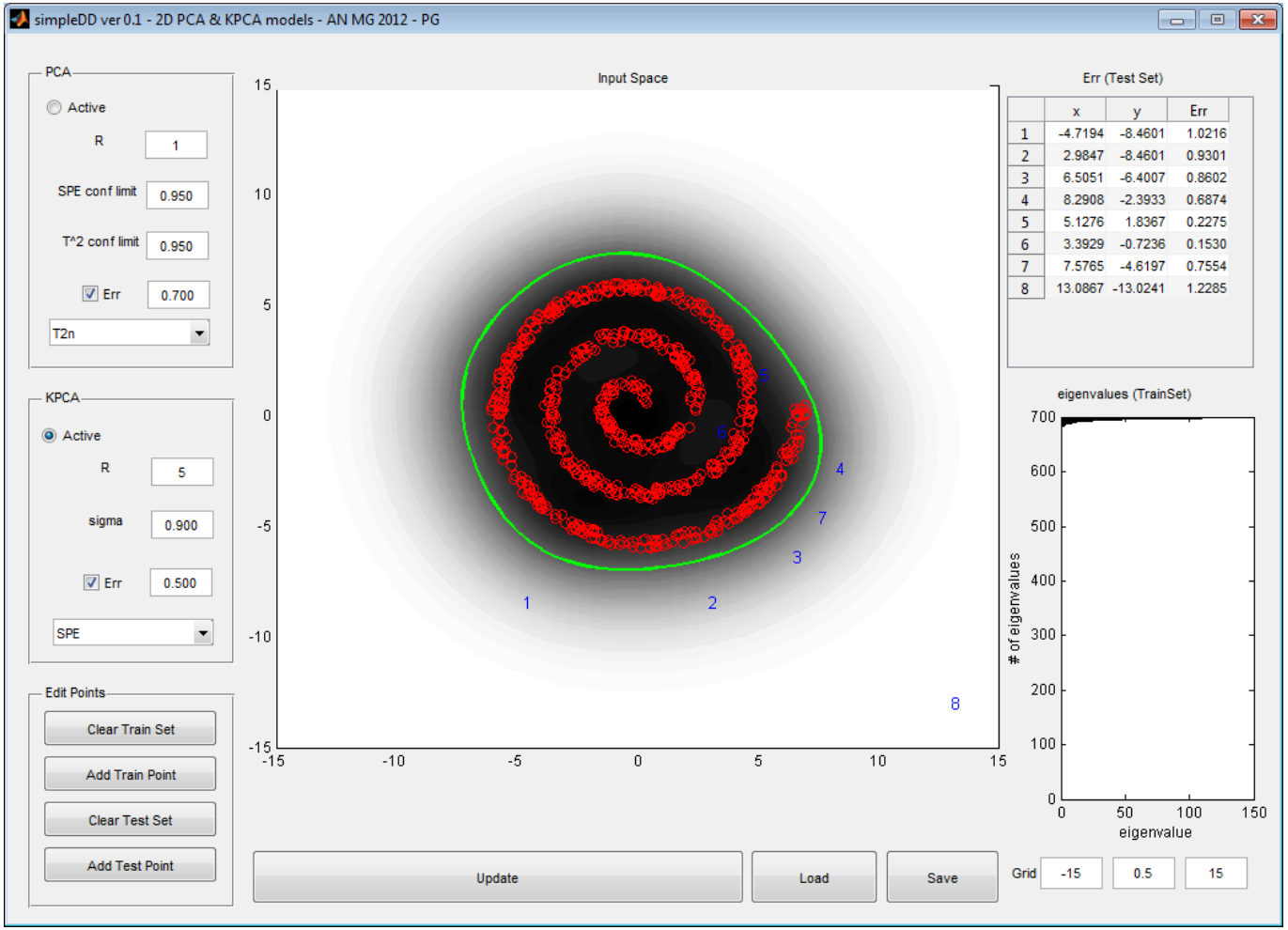
kernel PCA (kPCA) – analiza parametrów

Rozmiar przestrzeni F (ilość PCs)



kernel PCA (kPCA) – analiza parametrów

PCA vs kPCA - przykłady



Dziękuję za uwagę