METODY REDUKCJI WYMIARU: LLE, ISOMAPS, CONFORMAL EIGENMAPS

Małgorzata Salawa, Jakub Synowiec, Piotr Wójcik

14 listopada 2016

Akademia Górniczo-Hutnicza

REDUKCJA WYMIARU

Jak wykryć niskowymiarowe struktury w wielowymiarowym zbiorze danych?

Dane wejściowe: $\vec{x_i} \in \Re^D, i = 1, ..., n$

Dane wyjściowe: $\vec{y_i} \in \Re^d, d \ll D$

Zasada przekształcania $\vec{x}$ w $\vec{y}$:

- punkty położone względnie blisko, pozostają blisko
- punkty położone względnie daleko, pozostają daleko

Uwaga: **Nie dysponujemy etykietami**, tj. nie posiadamy zmiennej wyjściowej.

REDUKCJA WYMIARU

Struktury liniowe -> podprzestrzeń liniowa

Struktury nieliniowe -> rozmaitość (zachowuje właściwości przestrzeni euklidesowych wyłącznie dla lokalnych obszarów)

METODY REDUKCJI WYMIARÓW

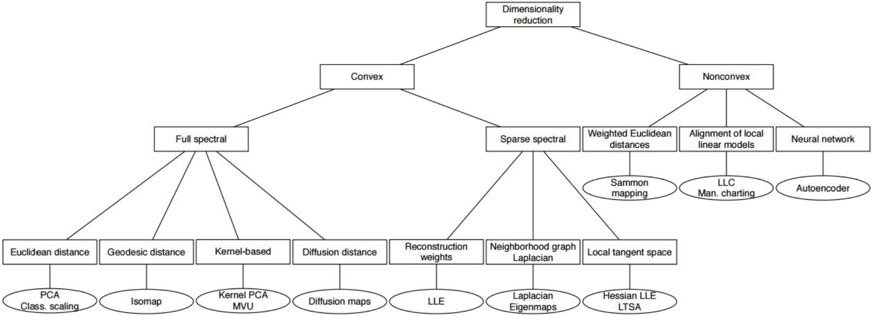


Figure 1: Taxonomy of dimensionality reduction techniques.

METODY REDUKCJI WYMIARÓW

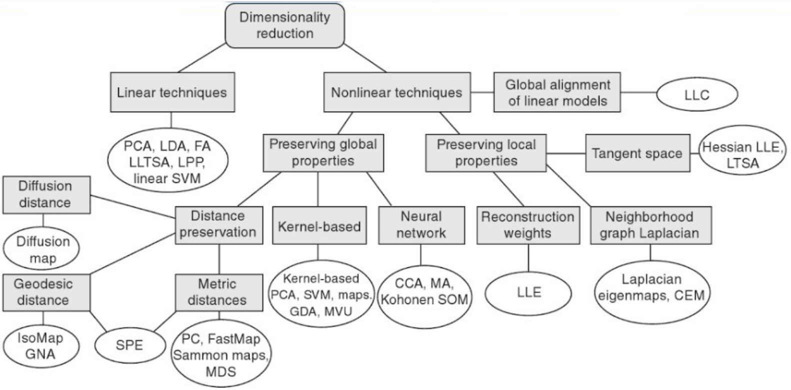
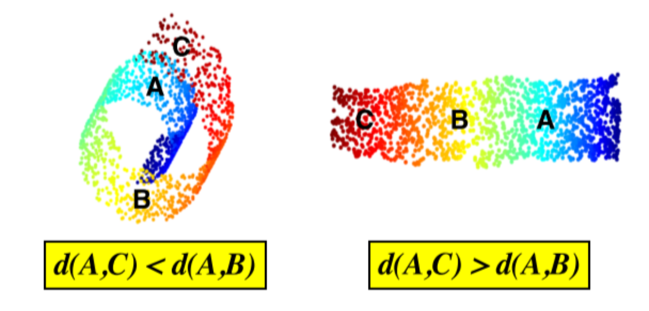


Figure 15.1 A functional taxonomy of advanced dimensionality reduction techniques. GNA = geodesic nullspace analysis; SOM = self-organizing map.

DLACZEGO NIELINIOWOŚĆ?



JAK?

- Analiza macierzy danych (PCA, MDS, …)
- Analiza grafów (LLE, Isomaps, Conformal Eigenmaps, …)
- …

LLE

LOCALLY LINEAR EMBEDDING - LLE

Metoda składa się z trzech kroków:

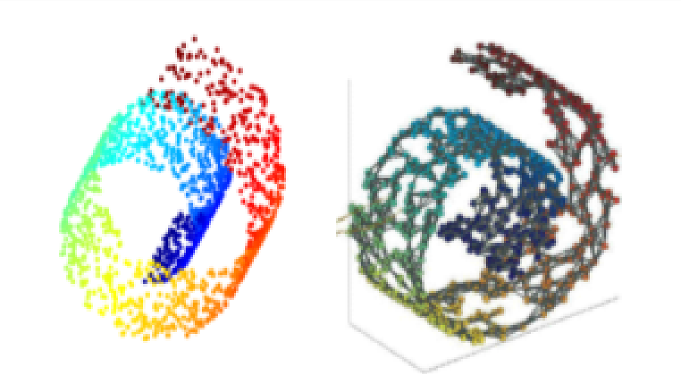
1. Wyszukujemy najbliższych sąsiadów (tworzymy graf sąsiedztwa)
2. Tworzymy macierz wag
3. Linearyzujemy model (przybliżamy modelu nieliniowy za pomocą liniowego)

1. WYSZUKANIE NAJBLIŻSZYCH SĄSIADÓW

Budujemy macierz sąsiedztwa dla grafu:

- Wierzchołki grafu reprezentują wektory wejściowe
- Nieskierowane krawędzie łączą punkty, które zawierają siebie nawzajem w swoim sąsiedztwie
- Różne metody wyboru sąsiedztwa:
  - k najbliższych punktów
  - punkty w odległości $d < \epsilon$
  - dodatkowe, sztywno narzucone wymagania

1. WYSZUKANIE NAJBLIŻSZYCH SĄSIADÓW



Złożoność obliczeniowa algorytmu budującego graf skaluje się do $O(n^2 D)$.

1. WYSZUKANIE NAJBLIŻSZYCH SĄSIADÓW

Po zbudowaniu grafu powinny być spełnione dwa warunki:

1. Graf jest spójny *(niespójność może mocno zaburzyć względne położenie spójnych części grafu przy redukcji wymiarów)*
2. Sąsiedzi wektorów danych wejściowych odwzorowują wierzchołki w przestrzeni rozmaitości *(zaburzenie może spowodować powstanie krawędzi między niechcianymi punktami)*

2. UTWORZENIE MACIERZY WAG

Tworzymy macierz wag poprzez minimalizację błędu:

$$\Phi(W) = \sum_i \left| \vec{x_i} - \sum_j W_{ij} \vec{x_j} \right|^2$$

Gdzie:

- $\sum_j W_{ij} = 1$
- $W_{ij} \neq 0$ tylko dla ustalonych wcześniej sąsiadów

Uwaga 1: Macierz wag nie zmienia się przy operacjach takich jak przesunięcie, obrót i przeskalowanie.

Uwaga 2: Można odtworzyć z dobrą dokładnością przestrzeń danych przy pomocy niewielkiej ilości punktów oraz macierzy wag.

3. LINEARYZACJA

Rozwiązujemy tutaj **Sparse Eigenvalue Problem**.

Odtwarzamy przestrzeń danych w mniejszym wymiarze:

- Minimalizujemy błędy, czyli funkcję:

$$\Psi(y) = \sum_i \left| \vec{y_i} - \sum_j W_{ij} \vec{y_j} \right|^2$$

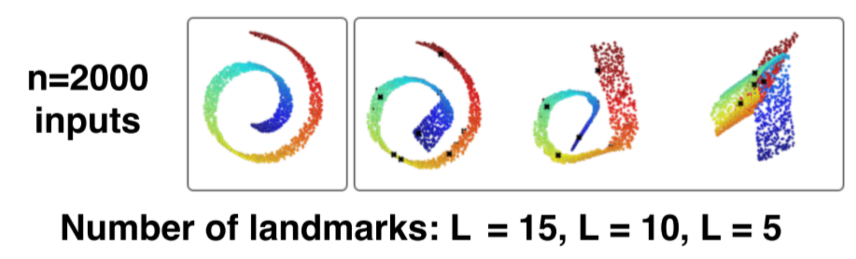
- Ustawiamy środek współrzędnych w środku sumy wektorów, czyli:

$$\sum_i \vec{y_i} = \vec{0}$$

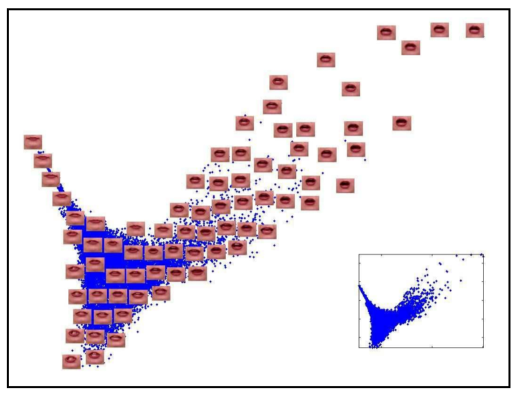
- Wymuszamy, aby macierz kowariancji spełniała równanie:

$$\frac{1}{N} \sum_i \vec{y_i} \vec{y_i}^T = I_d$$

3. LINEARYZACJA



LLE - REZULTAT

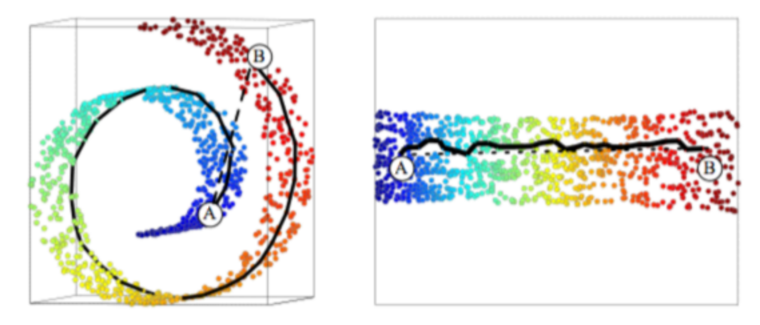


$n = 15960$ zdjęć, $k = 24$ najbliższych sąsiadów
$D = 65664$ piksele (wymiary wejściowe)
$d = 2$ wymiary wyjściowe

Isomaps

ISOMAPS

Podstawowa idea:
Zachowanie odległości (w sensie teorii grafów) w podrozmaitości.



ALGORYTM ISOMAPS

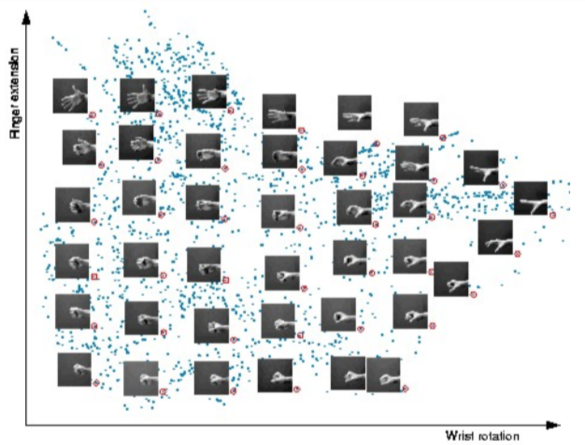
1. Budujemy **graf sąsiedztwa** (tak jak w 1. kroku LLE), dodatkowo ważymy go na podstawie odległości między punktami
2. Korzystając z grafu tworzymy **macierz sąsiedztwa** w taki sposób, że M_{ij} zawiera długość **najkrótszej ścieżki** pomiędzy punktami i i j
3. Na tak spreparowanym zbiorze danych wykonujemy redukcję **MDS** *(Skalowanie wielowymiarowe dąży do rozmieszczenia punktów tak, aby obiekty podobne do siebie znajdowały się bliżej)*

Uwaga: Sposób zdefiniowania odległości (jako ścieżka w grafie, a nie odległość Euklidesowa między punktami) jest w tej metodzie **źródłem nieliniowości**.

Redukcja MDS:

1. Podnosimy macierz odległości **P** do kwadratu (n x n)
2. Tworzymy macierz pomocniczą $J = I - \frac{1}{n}[1]$ ([1] jest wielkości (n x n))
3. Obliczamy macierz B jako $B = -\frac{1}{2}JP^2J$
4. Wyznaczamy d największych wartości własnych i odpowiadające im wektory własne
5. Macierz rozwiazan $Y = ED$, gdzie E to macierz wektorów własnych, a D to macierz, na której przekątnej znajdują się pierwiastki z wartości własnych

ISOMAPS - REZULTAT

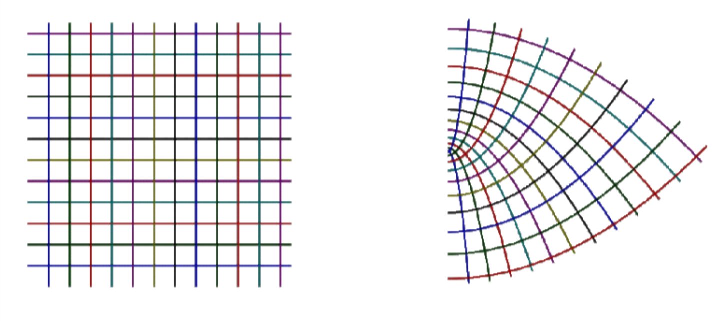


$n = 2000$, $k = 6$ najbliższych sąsiadów, $D = 64^2$

Conformal Eigenmaps

CONFORMAL EIGENMAPS

Podstawowa idea:
Zachowanie zarówno kątów, jak i odległości.



CONFORMAL EIGENMAPS

Rozważmy trojkąt (x_h, x_i, x_j) w przestrzeni D-wymiarowej oraz (y_h, y_i, y_j) w przestrzeni d-wymiarowej, gdzie $d < D$.

Odwzorowanie między tymi przestrzeniami na tych trójkątach jest odwzorowaniem równokątnym (conformal), jeśli:

$$\frac{\|y_h - y_i\|^2}{\|x_h - x_i\|^2} = \frac{\left\|y_i - y_j\right\|^2}{\left\|x_i - x_j\right\|^2} = \frac{\left\|y_h - y_j\right\|^2}{\left\|x_h - x_j\right\|^2}$$

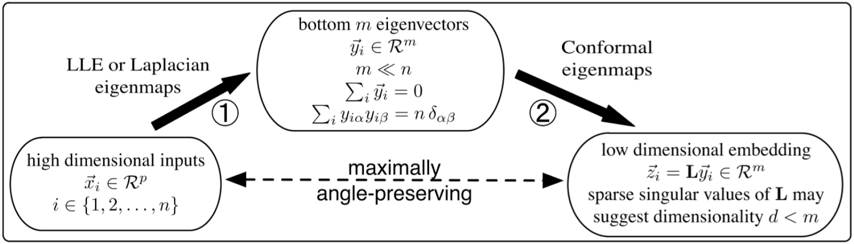
CONFORMAL EIGENMAPS

Bazując na tej obserwacji, definiujemy miarę C_h, która oszacowuje równokątność dla zmiennej x_h w grafie sąsiedztwa:

$$C_h = \sum_{ij} \eta_{hi}\eta_{ij}(\left\|y_i - y_j\right\|^2 - s_h \left\|x_i - x_j\right\|^2)^2$$

Gdzie:
η_{ij} - przyjmuje 1 jeśli x_i i x_j są połączone w grafie sąsiedztwa
s_h - reprezentuje przekształcenia w transformacji z wysokowymiarowej przestrzeni X do jej niskowymiarowego odpowiednika Y.

SCHEMAT ALGORYTMU REDUKCJI WYMIARÓW



SCHEMAT ALGORYTMU REDUKCJI WYMIARÓW

Etap 1. był już omówiony przy okazji LLE.

Etap 2.
Posiadamy już wektory y_i i chcemy dokonać transformacji

$$\vec{z_i} = \mathsf{L}\vec{y_i}$$

Aby wyeliminować rozwiązanie trywialne $L = 0$, dokładany jest dodatkowy warunek:

$$trace(\mathsf{L}^T\mathsf{L}) = 1$$

Gdzie:
$trace(M)$ - ślad macierzy M (suma wartości na przekątnej głównej)

Problem sprowadza się do znalezienia macierzy L oraz minimalizacji funkcji C_h.

CONFORMAL EIGENMAPS - FUNKCJA KOSZTU

Optymalne parametry skalowania mogą zostać obliczone jako funkcja macierzy $\mathbf{L}$ $=>$ funkcja kosztu zależy tylko od macierzy $\mathbf{L}$.

Niech $\mathbf{P} = \mathbf{L}^T\mathbf{L}$ i $p = vec(\mathbf{P})$ - wektor powstały w wyniku konkatenacji kolumn macierzy $\mathbf{P}$. Wtedy problem optymalizacji można zapisać jako:

$$\text{minimize } t \text{ such that:}$$
$$P \succeq 0,$$
$$trace(P) = 1,$$
$$\begin{pmatrix} \mathbf{I} & \mathbf{S}p \\ (\mathbf{S}p)^T & t \end{pmatrix} \succeq 0$$

Gdzie $\mathbf{S}$ jest zależna od $\{\vec{x_i}, \vec{y_i}\}_{i=1}^n$, ale niezależna od zmiennych optymalizacyjnych $\mathbf{P}$ ani t, a t to nieznana wartość skalująca.

CONFORMAL EIGENMAPS - ROZWIĄZANIE

Jest to instancja **problemu programowania półokreślonego** (SDP - Semidefinite Programming) nad nieznaną macierzą $\mathbf{P}$.

Po rozwiązaniu SDP, macierz $\mathbf{P}$ może być zdekomponowana do $\mathbf{L}^T\mathbf{L}$, skąd otrzymamy ostateczne rozwiązanie:

$$z_i = \mathbf{L}y_i \text{ dla każdego } i$$

CONFORMAL EIGENMAPS - ALTERNATYWNE PODEJŚCIE

Sformułowanie problemu minimalizacji zostaje zmodyfikowane (parametry s_i wyliczone metodą najmniejszych kwadratów), dodany zostaje warunek wykluczający trywialne rozwiązanie:

$$\begin{aligned} D(L) &= min_L(Vec(P)^T QVec(P)) + \lambda_1(trace(P) - 1)^2 \\ &= min_L(Vec(L^T L)^T QVec(L^T L)) + \lambda_1(trace(L^T L) - 1)^2 \end{aligned}$$

Otrzymujemy tutaj równanie 4. stopnia względem macierzy **L** - a więc trudne do rozwiązania.

CONFORMAL EIGENMAPS - ALTERNATYWNE PODEJŚCIE

Podstawowa idea:
Zmieniamy $P = L^T L$ na $P = B^T C$, (przy warunku $Vec(B) = Vec(C)$), dzięki czemu **redukujemy stopień równania** do 2 względem B i względem C.

Po przekształceniach otrzymujemy równania umożliwiające **iteracyjne przybliżanie** wartości macierzy B^T i C.

CONFORMAL EIGENMAPS - ALGORYTM

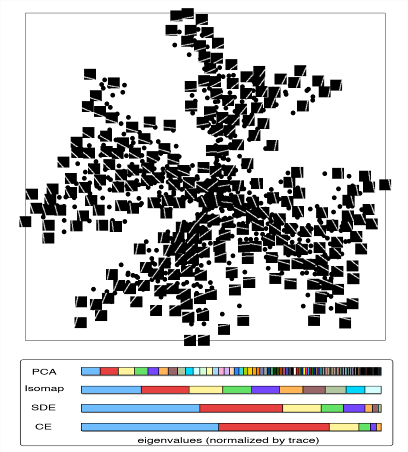
```
X = GetInitialHighDimensionalPoints()
Y = RunLLE(X)
Q = ComputeQMatrix(X,Y)
B,C = RandomMatrix()
While (ErrorFunction(B,C) > ErrorMargin)
      C = EstimateC (B, Q)
      B = EstimateB (C, Q)
EndWhile

L = B or L = C since B≈C

Z = LY to find the final embedding
```

Gdzie funkcje *EstimateB*() i *EstimateC*() bazują na skomplikowanych równaniach zależnych od macierzy C i B, a *CompyteQMatrix*() zależy tylko od $\{x_i, y_i\}_{i=1}^{n}$.

CONFORMAL EIGENMAPS - REZULTAT



$n = 2016,\ D = 24^2$

Pytania?

Praktyka

CONFORMAL EIGENMAPS - REZULTAT

https://github.com/meteran/reduction