Dimensionality reduction

Outline

- From distances to points :
  - MultiDimensional Scaling (MDS)
  - FastMap
- Dimensionality Reductions or data projections
- Random projections
- Principal Component Analysis (PCA)

Multi-Dimensional Scaling (MDS)

- So far we assumed that we know both data points **X** and distance matrix **D** between these points
- What if the original points **X** are not known but only distance matrix **D** is known?
- Can we reconstruct **X** or some approximation of **X**?

Problem

- Given distance matrix **D** between **n** points
- Find a **k**-dimensional representation of every $\mathbf{x_i}$ point **i**
- So that $\mathbf{d(x_i,x_j)}$ is as close as possible to **D(i,j)**

Why do we want to do that?

How can we do that? (Algorithm)

High-level view of the MDS algorithm

- Randomly initialize the positions of **n** points in a **k**-dimensional space
- Compute pairwise distances **D'** for this placement
- Compare **D'** to **D**
- Move points to better adjust their pairwise distances (make **D'** closer to **D**)
- Repeat until **D'** is close to **D**

The MDS algorithm

- ***Input:*** **n x n** distance matrix **D**
- Random **n** points in the **k**-dimensional space **(x_1,…,x_n)**
- **stop = false**
- **while not stop**
  - **totalerror = 0.0**
  - For every **i,j** compute
    - **D'(i,j)=d(x_i,x_j)**
    - **error = (D(i,j)-D'(i,j))/D(i,j)**
    - **totalerror +=error**
    - **For** every dimension **m**: **x_{im} = (x_{im}-x_{jm})/D'(i,j)*error**
  - **If totalerror** small enough, **stop = true**

Questions about MDS

- Running time of the MDS algorithm
  - **$O(n^2 I)$,** where **I** is the number of iterations of the algorithm

- MDS does not guarantee that metric property is maintained in **d'**

- Faster? Guarantee of metric property?

Problem (revisited)

- Given distance matrix **D** between **n** points
- Find a **k**-dimensional representation of every x_i point **i**
- So that:
  - $d(x_i,x_j)$ is as close as possible to **D(i,j)**
  - $d(x_i,x_j)$ is a metric
  - Algorithm works in time ***linear*** in **n**

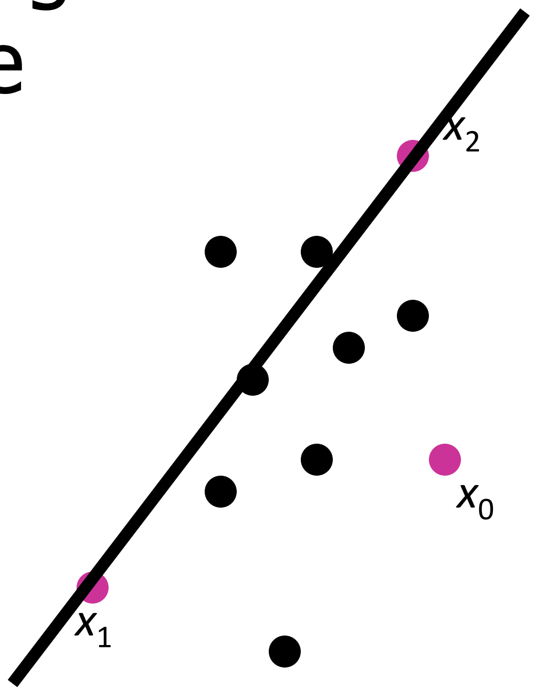
FastMap

- Select two **pivot points** $\mathbf{x}_a$ and $\mathbf{x}_b$ that are far apart.
- Compute a **pseudo-projection** of the remaining points along the “line” $\mathbf{x}_a\mathbf{x}_b$
- **“Project”** the points to a subspace orthogonal to “line” $\mathbf{x}_a\mathbf{x}_b$ and **recurse**.

Selecting the Pivot Points

The pivot points should lie along the principal axes, and hence should be far apart.

- Select any point $\mathbf{x_0}$
- Let $\mathbf{x_1}$ be the furthest from $\mathbf{x_0}$
- Let $\mathbf{x_2}$ be the furthest from $\mathbf{x_1}$
- Return $(\mathbf{x_1}, \mathbf{x_2})$



Pseudo-Projections

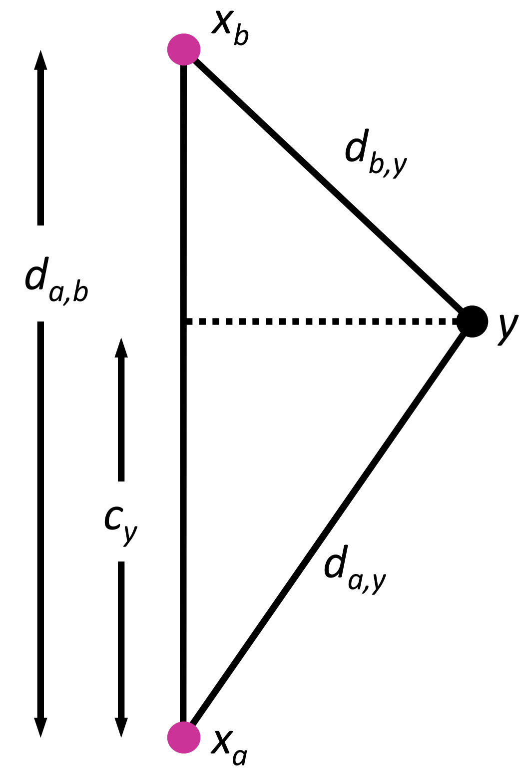
Given pivots (**x_a , x_b**), for any third point *y*, we use the **law of cosines** to determine the relation of **y** along **$x_a x_b$**

$$d_{by}^2 = d_{ay}^2 + d_{ab}^2 - 2c_y d_{ab}$$

The **pseudo-projection** for **y** is

$$c_y = \frac{d_{ay}^2 + d_{ab}^2 - d_{by}^2}{2d_{ab}}$$

This is first coordinate.

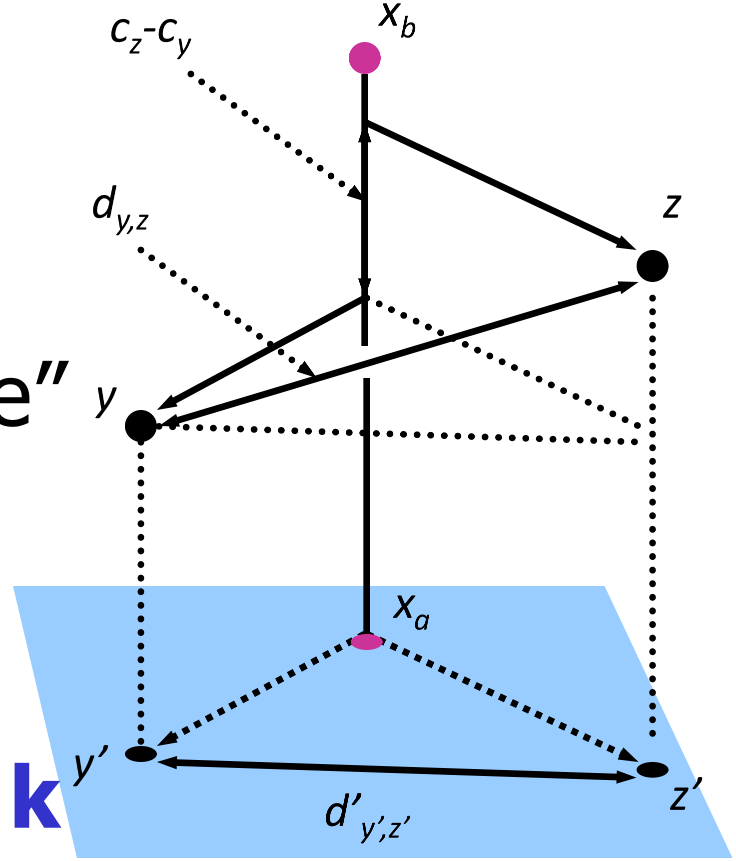


"Project to orthogonal plane"

Given distances along **$x_a x_b$** compute distances within the "orthogonal hyperplane"

$$d'(y',z') = \sqrt{d^2(y,z) - (c_z - c_y)^2}$$

Recurse using **d '(.,.)**, until **k** features chosen.



The FastMap algorithm

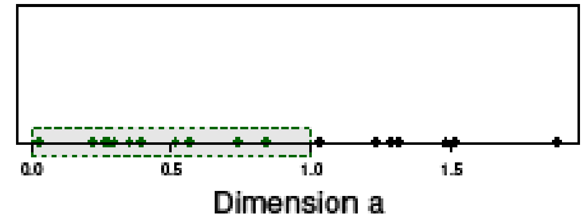
- **D:** distance function, **Y: nxk** data points
- **f=0** //global variable
- FastMap(**k,D**)
  - If **k<=0** return
  - **(x_a,x_b)**← chooseDistantObjects(**D**)
  - If(**D(x_a,x_b)==0),** set **Y[i,f]=0** for every **i** and return
  - **Y[i,f] = [D(a,i)^2+D(a,b)^2-D(b,i)^2]/(2D(a,b))**
  - **D'(i,j) // new distance function on the projection**
  - **f++**
  - FastMap(**k-1,D'**)

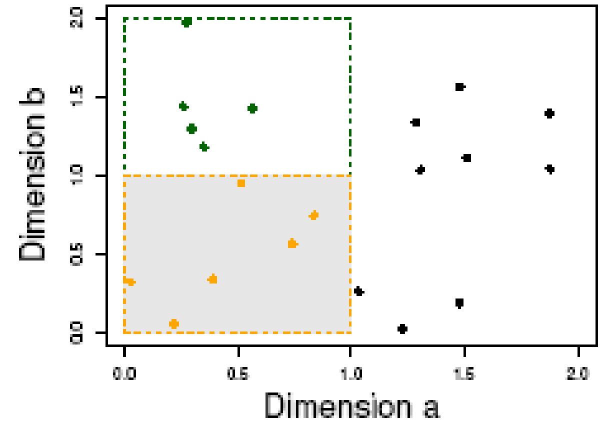
FastMap algorithm

- Running time
  - Linear number of distance computations

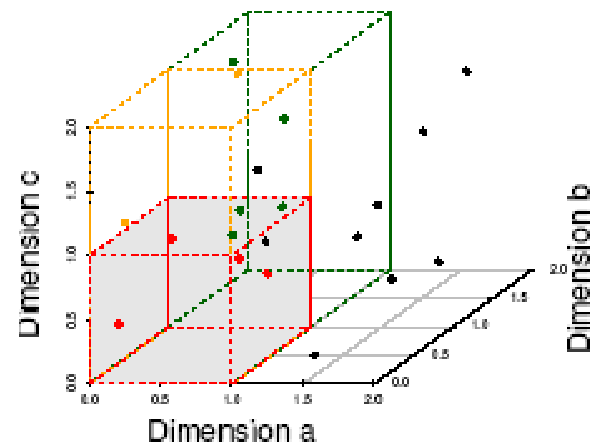
The Curse of Dimensionality

- Data in only one dimension is relatively packed
- Adding a dimension "stretches" the points across that dimension, making them further apart
- Adding more dimensions will make the points further apart—high dimensional data is extremely sparse
- Distance measure becomes meaningless





(b) 6 Objects in One Unit Bin



(c) 4 Objects in One Unit Bin

(graphs from Parsons et al. KDD Explorations 2004)

The curse of dimensionality

- The efficiency of many algorithms depends on the number of dimensions **d**

  – Distance/similarity computations are at least linear to the number of dimensions

  – Index structures fail as the dimensionality of the data increases

Goals

- Reduce dimensionality of the data
- Maintain the meaningfulness of the data

Dimensionality reduction

- Dataset **X** consisting of **n** points in a **d**-dimensional space
- Data point $\mathbf{x_i \in R^d}$ (**d**-dimensional real vector):

  $\mathbf{x_i = [x_{i1}, x_{i2}, \ldots, x_{id}]}$
- Dimensionality reduction methods:
  - **Feature selection:** choose a subset of the features
  - **Feature extraction:** create new features by combining new ones

Dimensionality reduction

- Dimensionality reduction methods:
  - **Feature selection:** choose a subset of the features
  - **Feature extraction:** create new features by combining new ones
- Both methods map vector $\mathbf{x_i \epsilon R^d}$, to vector $\mathbf{y_i \ \epsilon \ R^k, (k << d)}$

- $\mathbf{F : R^d \rightarrow R^k}$

Linear dimensionality reduction

- Function **F** is a ***linear*** projection
- $\mathbf{y_i = A\, x_i}$
- $\mathbf{Y = A\, X}$
- **Goal:** **Y** is as ***close*** to **X** as possible

Closeness: Pairwise distances

- **Johnson-Lindenstrauss lemma:** Given $\varepsilon>0$, and an integer n, let k be a positive integer such that $k \geq k_0 = O(\varepsilon^{-2} \log n)$. For every set X of n points in R^d there exists $F: R^d \rightarrow R^k$ such that for all $x_i, x_j \in X$

$$(1-\varepsilon)||x_i - x_j||^2 \leq ||F(x_i) - F(x_j)||^2 \leq (1+\varepsilon)||x_i - x_j||^2$$

What is the intuitive interpretation of this statement?

JL Lemma: Intuition

- Vectors $x_i \epsilon R^d$, are projected onto a **k**-dimensional space (**k<<d**): $y_i = R\, x_i$
- If $||x_i||=1$ for all **i**, then,

  $||x_i - x_j||^2$ is approximated by $(d/k)||x_i - x_j||^2$
- **Intuition:**
  - The expected squared norm of a projection of a unit vector onto a random subspace through the origin is **k/d**
  - The probability that it deviates from expectation is very small

JL Lemma: More intuition

- **$x=(x_1,\dots,x_d)$**, **d** independent Gaussian N(0,1) random variables; **$y = 1/|x|(x_1,\dots,x_d)$**
- **z :** projection of **y** into first **k** coordinates
  - **$L = |z|^2$, $\mu = E[L] = k/d$**
- **$Pr(L \geq (1+\varepsilon)\mu) \leq 1/n^2$** and **$Pr(L \leq (1-\varepsilon)\mu) \leq 1/n^2$**
- **$f(y) = sqrt(d/k)z$**
- What is the probability that for pair **(y,y')**: **$|f(y)-f(y')|^2/(|y-y'|)$** ***does not*** lie in range **$[(1-\varepsilon),(1+\varepsilon)]$**?
- What is the probability that some pair suffers?

Finding random projections

- Vectors $\mathbf{x_i} \epsilon \mathbf{R^d}$, are projected onto a $\mathbf{k}$-dimensional space ($\mathbf{k<<d}$)
- Random projections can be represented by linear transformation matrix $\mathbf{R}$
- $\mathbf{y_i = R\ x_i}$

- What is the matrix $\mathbf{R}$?

Finding random projections

- Vectors $\mathbf{x_i \epsilon R^d}$, are projected onto a **k**-dimensional space (**k<<d**)
- Random projections can be represented by linear transformation matrix **R**
- $\mathbf{y_i = R\ x_i}$

- What is the matrix **R**?

Finding matrix **R**

- Elements **R(i,j)** can be Gaussian distributed
- Achlioptas* has shown that the Gaussian distribution can be replaced by

$$R(i,j) = \begin{cases} +1 \text{ with prob } \frac{1}{6} \\ 0 \text{ with prob } \frac{2}{3} \\ -1 \text{ with prob } \frac{1}{6} \end{cases}$$

- All zero mean, unit variance distributions for **R(i,j)** would give a mapping that satisfies the ***JL*** lemma

- **Why is Achlioptas result useful?**