Note-08

Manifold Learning (continued)

Why manifold learning?

goal: reduce the dimensionality of the data but the structurer of the data is preserved

MDS

"PCA on distances(dissimilarities)"

$$C = (I - \frac{1}{N}1^T 1) \tag{1}$$

$$SVD(\frac{1}{2}C^{-T}D^2C) \Rightarrow U^T\Sigma^{\frac{1}{2}} = X \tag{2}$$

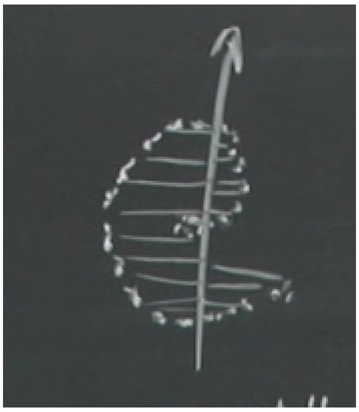
which Σ = zero out singular values for demined

Another item to relax from the PCA formulation:

Can we perform a local projection instead of a global one, Example case (not nicely solvable with PCA)

eg:

"swiss roll"



MDS also has difficulties here!

because the globally applied distances can not distinguish local structures on the manifold

"ISOMAP"(isometric Feature Mapping)

Perform MDS, but on graph distances instead of some global metric like the Euclidean distance.

Graph distance:

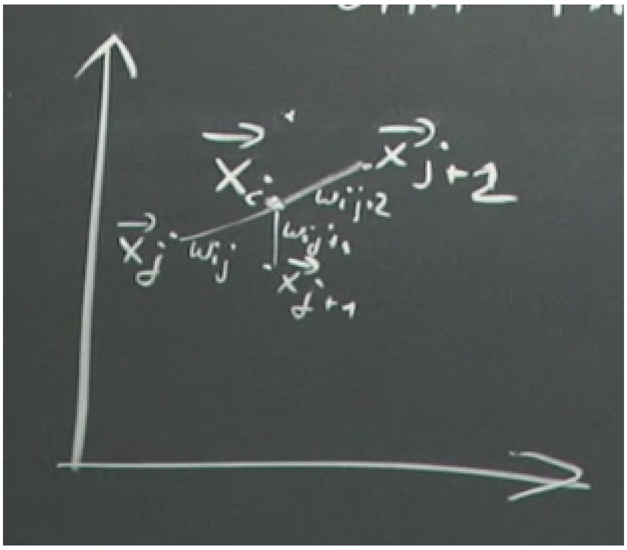
- Define a local neighborhood, compute Euclid distance to these neighbors ⇒ Spare adjacency matrix

- Compute all-pairs-shortest paths to obtain all missing distances
  ⇒ this is also called a "geodesic" distance

Locally Linear Embedding (LLE)

Idea

- Represent every sample as a linear combination of its neighbors
- Search for a lower dimensional representation with the same / similar linear combination of neighbors



- 1) Search for weight w_{ij}:

$$min \Sigma ||x_i - \Sigma_{j \in N(x_i)} w_{ij} x_j||_2^2 \tag{3}$$

subject to $\Sigma_{j \in N(x_i)} w_{ij} = 1$

- 2) In the lower dimensional space:

Solve for $x^{‘}$ in

$$min \Sigma ||x_i - \Sigma_{j \in N(x_i)} w_{ij} x_j||_2^2 \tag{4}$$

subject to $\frac{1}{N} \Sigma_i x_i x_i^T = I$
covariance matrix is a identity
$\Sigma x_i = 0$

Note: For the linear constraint $\Sigma_{j \in N(x_i)} w_{ij} = 1$, we need a linear programming solver.

⇒ Let`s consider the earlier problem with a quadratic constraint $\Sigma_{j \in N(i) w_{ij}^2} = 1$

⇒ computer the derivative, solve a linear system

Let`s work on the solution of the first part of the problem.
The objective function is invariant to translations:

$$\Sigma_{i=1}^N ||x_i - \Sigma_{j \in N(x_i)}||_2^2 = \Sigma_{i=1}^N ||x_i + t - \Sigma_{j \in N(x_i - t)}||_2^2 \tag{5}$$

for an arbitrary but fixed i, we set $t = -x_i$

$$\Rightarrow ||x_i - x_i - \Sigma_j w_{ij}(x_j - x_i)||_2^2 = ||x_i - x_i - \Sigma_j w_{ij}(x_j - x_i)|| \tag{6}$$

$$||w_{i1}(x_1 - x_i) + w_{i2}(x_2 - x_1) + ... ||_2^2 = ||M_i * w_i|| \tag{7}$$

$$\Rightarrow minimize(M_i w_i)^T (M_i w_i) + \lambda * (1 - w_i^T w_i) \tag{8}$$

Note: $M_{ij} = 0$ if $x_j \not\in x_i$

$$\frac{\partial}{\partial w_i}(w_i^T M_i^T M_i w_i + \lambda(1 - w_i^T w_i) = 2M_i^T M_i w_i - \lambda 2 w_i \not= 0 \tag{9}$$

$M_i^T M_i * w_i = \lambda w_i$

Eigenvalue / Eigenvector problem $\Leftrightarrow$ take eigenvector that belongs to smallest non=zero eigenvalue

2 step: solve second object function with the calculated weights w_i for x_i (solution is similar)

Laplacian Eigenmaps

- Build an adjacency graph form the feature neighborhood
- Compute affinities between neighbored noted(weight)
- Perform eigen decomposition of the Graph Laplacian of the weights
- Low-dimension embedding

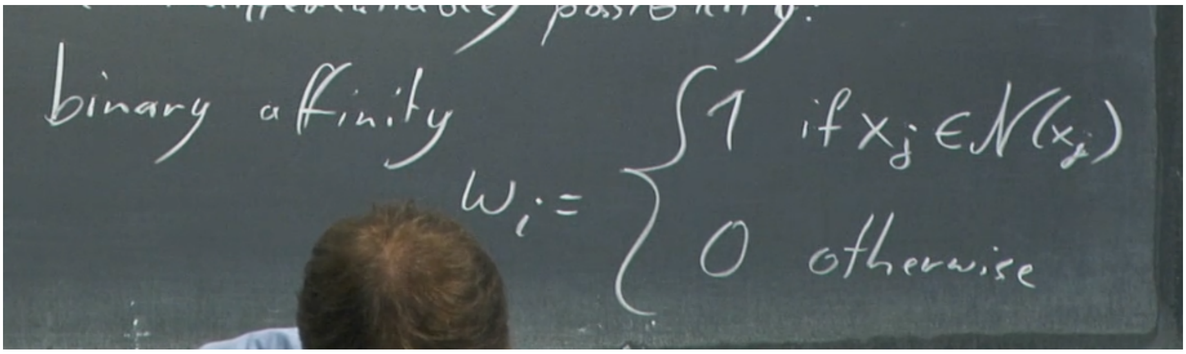
Step 1:

- Everything with in field (Euclidean?) distance d is a neighbor
- The k nearest samples are neighbors

Step 2: Pick any function that decreases with increasing distance 1 specific proposal heat kernel

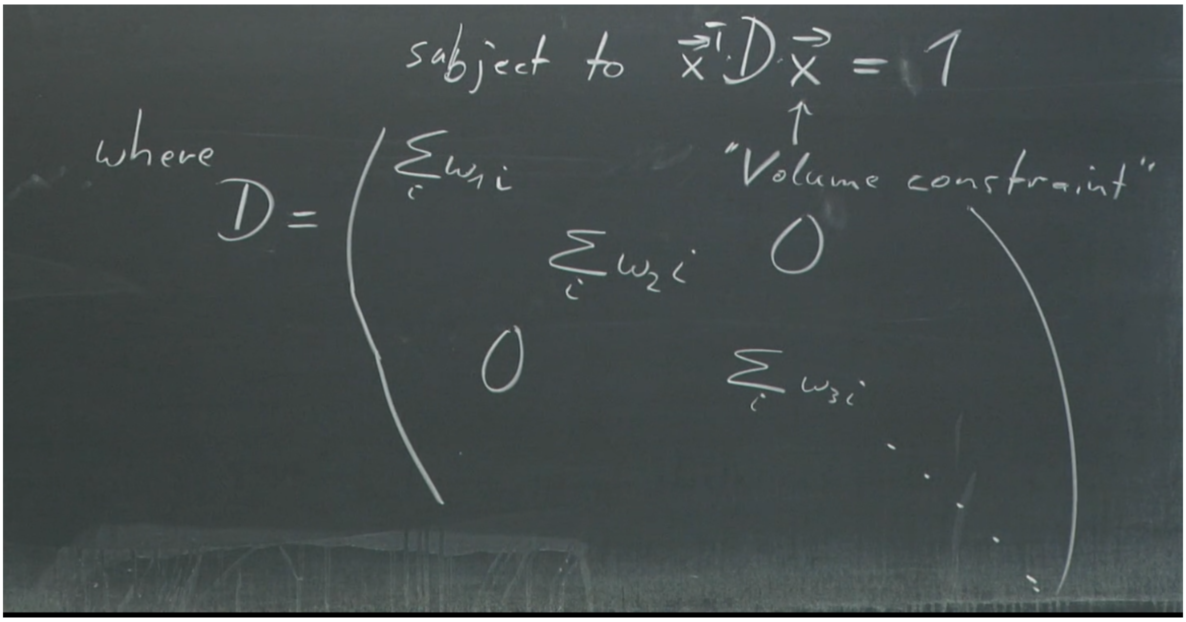
$$w_i = e^{\frac{-||x_i - x_j||_2^2}{t}} \tag{10}$$

another (non-differentiable possibility):



Step 3: theoretical derivation

minimize $\Sigma_{i=1}^N \Sigma_{j=1}^N (x_i^{`} - x_j^{`})^2 w_{ij}$



Rewrite the objective function:

$$\Sigma_{ij}(x_i^{‘} - x_j^{‘})^2 w_{ij} = \Sigma_{ij}(x_i^{‘2} - x_j^{‘2} - 2x_j^{‘}x_j^{‘})w_{ij} \tag{11}$$

$$= \Sigma_{ij}(x_i^{‘2} w_{ij} + \Sigma_{ij}(x_j^{‘2} w_{ij} - 2\Sigma_{ij} x_i^{‘} x_j^{‘} w_{ij} \tag{12}$$

$$= 2\Sigma_{ij} x_i^{‘2} w_{ij} - 2\Sigma_{ij} x_i^{‘} x_j^{‘} w_{ij} \tag{13}$$

$$= 2(x^{‘T} D x^{‘} - x^{‘} W_{x_i^{‘}}) \tag{14}$$

$$= 2x^{T^{‘}}(D - W)x \tag{15}$$

Minimize $x^{‘T} L x^{‘}$ subject to $x^{‘T} D x^{‘} =^{‘}$

$$\Rightarrow \frac{\partial}{\partial x}(x^{‘T} L x^{‘} + \lambda(1 - x^{‘T} D x^{‘}) \tag{16}$$

$$= 2Lx^{‘} - \lambda D x^{‘} \not= 0 \tag{17}$$

$$\Rightarrow D^{-1} L x^{‘} = \lambda x^{‘} \tag{18}$$