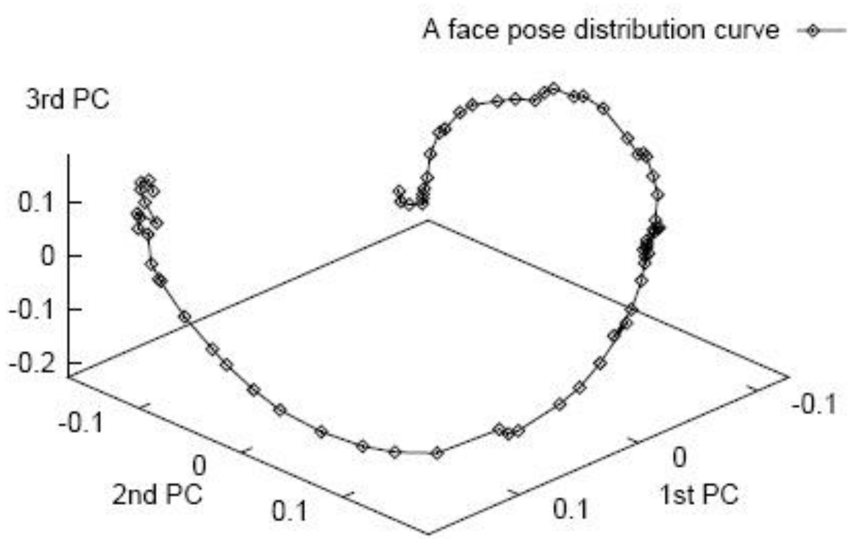
Nonlinear Methods

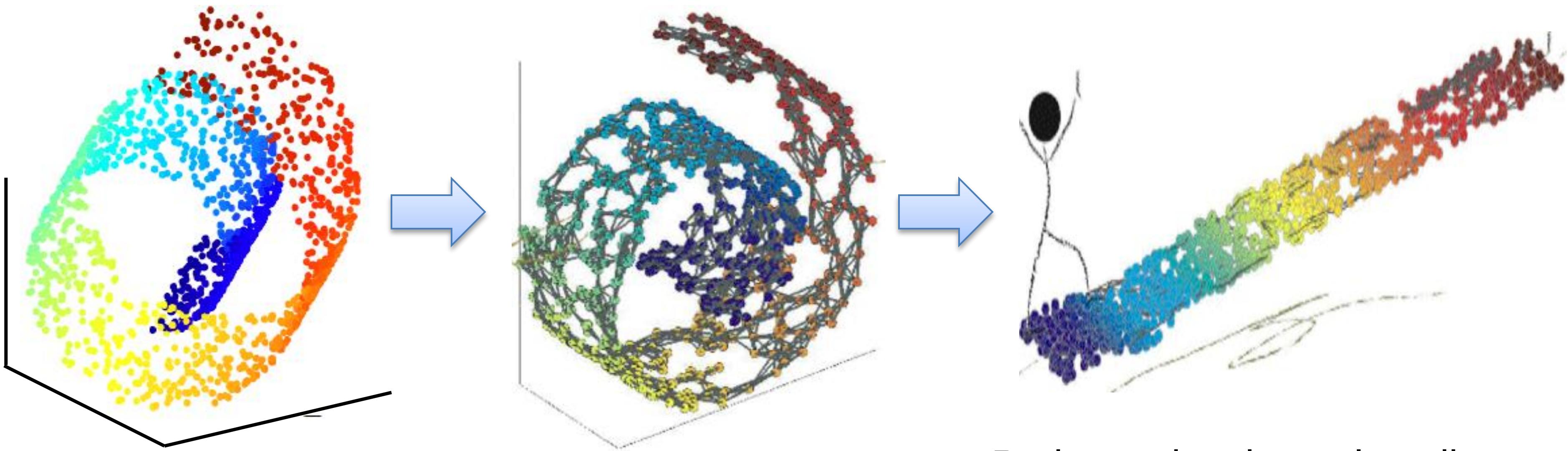
Data often lies on or near a nonlinear low-dimensional curve aka manifold.



Laplacian Eigenmaps

Linear methods – Lower-dimensional linear projection that preserves distances between **all** points

Laplacian Eigenmaps (key idea) – preserve **local** information only

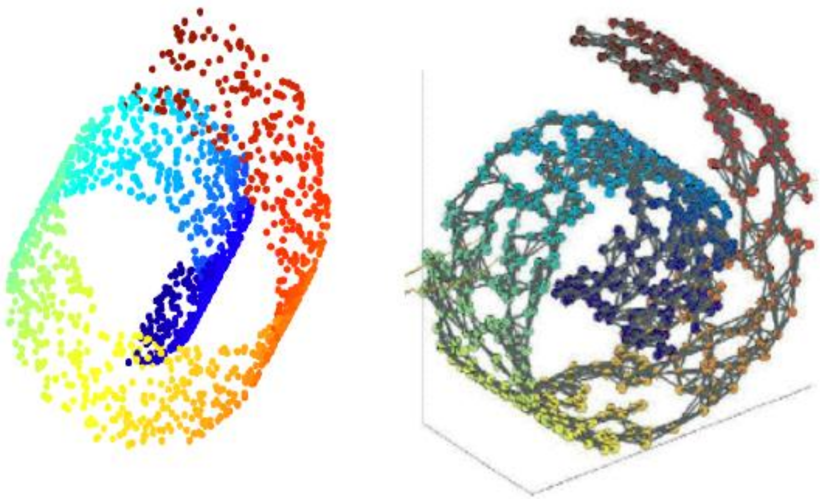


Construct graph from data points (capture local information)

Project points into a low-dim space using "eigenvectors of the graph"

Step 1 - Graph Construction

Similarity Graphs: Model local neighborhood relations between data points



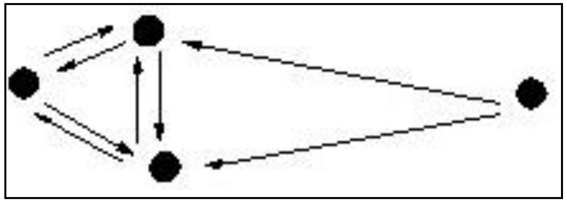
G(V,E) V – Vertices (Data points)

(1) E – Edge if $||x_i - x_j|| \leq \varepsilon$
ε – neighborhood graph

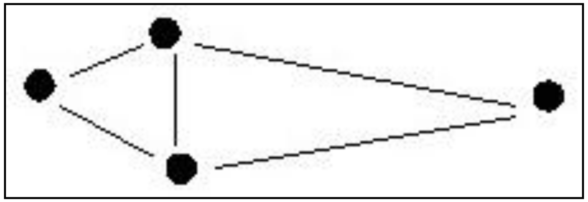
(2) E – Edge if k-NN,

yields directed graph

connect A with B if $A \rightarrow B$ OR $A \leftarrow B$ (symmetric kNN graph)
connect A with B if $A \rightarrow B$ AND $A \leftarrow B$ (mutual kNN graph)



Directed nearest neighbors



(symmetric) kNN graph

mutual kNN graph

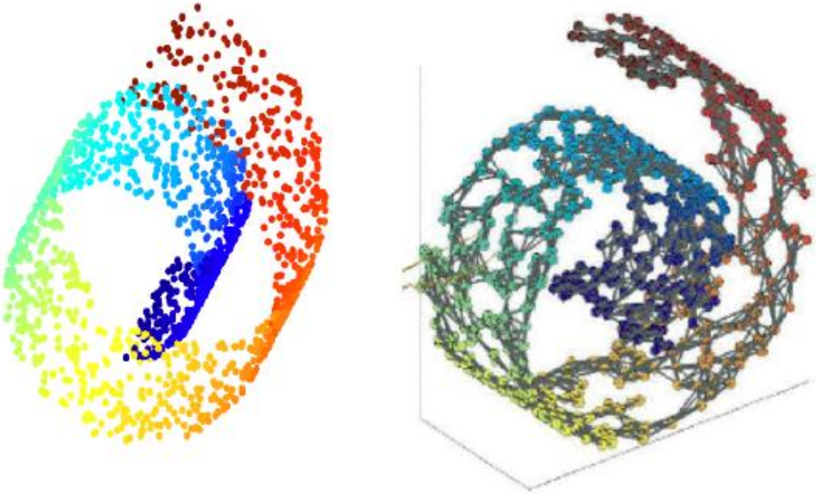
Step 1 - Graph Construction

Similarity Graphs: Model local neighborhood relations between data points

Choice of ε and k :

Chosen so that neighborhood on graphs represent neighborhoods on the manifold (no “shortcuts” connect different arms of the swiss roll)

Mostly ad-hoc



Step 1 - Graph Construction

Similarity Graphs: Model local neighborhood relations between data points

G(V,E,W) V – Vertices (Data points) E – Edges (nearest neighbors)

W - Edge weights

E.g. 1 if connected, 0 otherwise (Adjacency graph)

Gaussian kernel similarity function (aka Heat kernel)

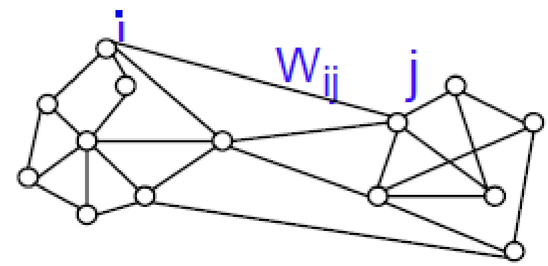
$$W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$

$\sigma^2 \rightarrow \infty$ results in adjacency graph

Step 2 – Embed using Graph Laplacian

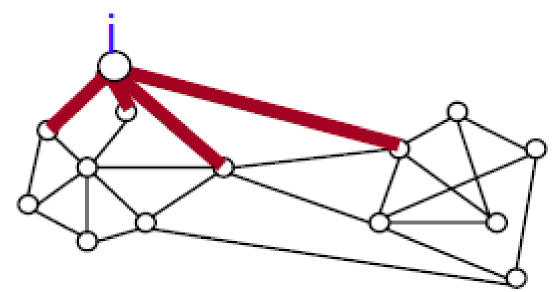
- Graph Laplacian (unnormalized version)

$$L = D - W$$



W – Weight matrix

D – Degree matrix = $diag(d_1, \ldots, d_n)$



$d_i = \sum_j w_{ij}$ degree of a vertex

Note: If graph is connected,
1 is an eigenvector

$$\mathbf{L1} = \begin{bmatrix} d_1 - \sum_j w_{1j} \\ d_2 - \sum_j w_{2j} \\ \cdots \\ d_n - \sum_j w_{nj} \end{bmatrix} = 0$$

Step 2 – Embed using Graph Laplacian

- Graph Laplacian (unnormalized version)

$$L = D - W$$

Solve generalized eigenvalue problem $\boxed{\mathbf{Lf} = \lambda\mathbf{Df}}$

Order eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \lambda_3 \leq \ldots \leq \lambda_n$

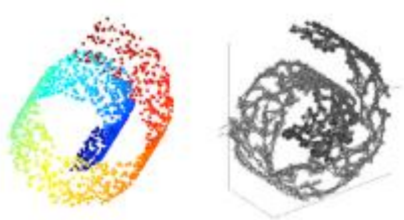
To embed data points in d-dim space, project data points onto eigenvectors associated with $\lambda_2, \lambda_3, \ldots, \lambda_{d+1}$

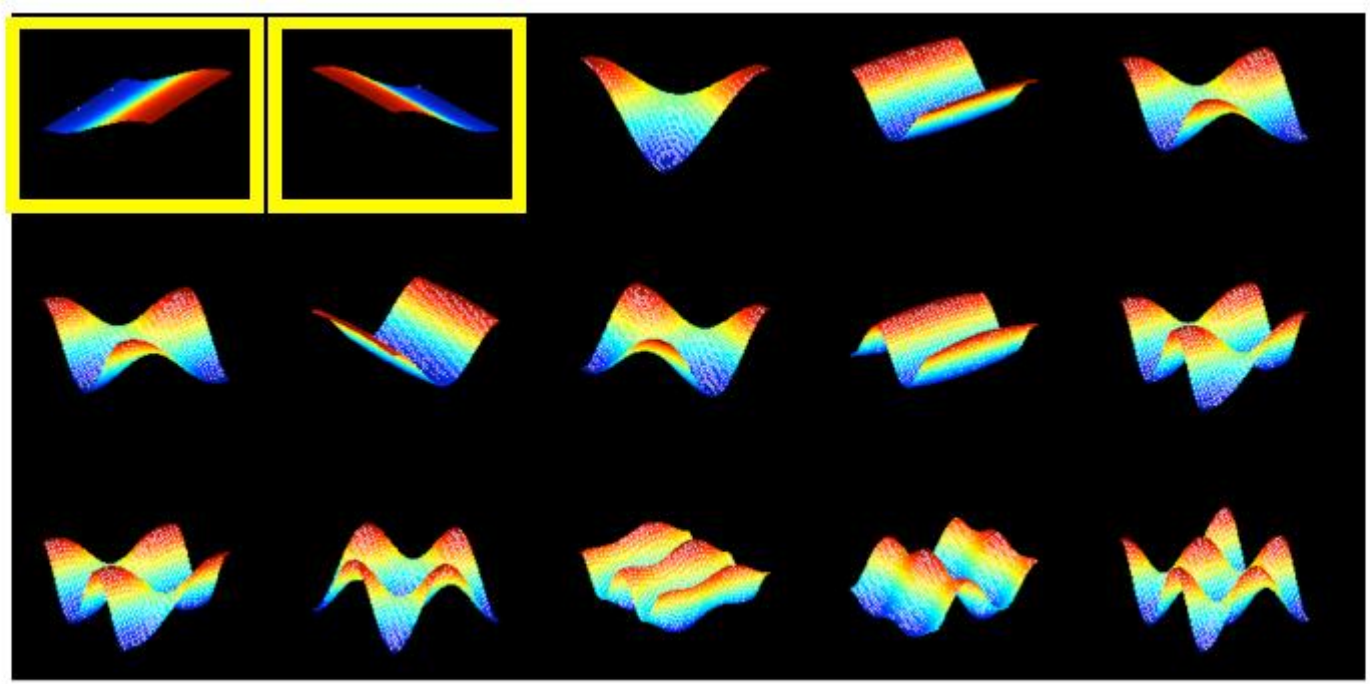
ignore 1st eigenvector – same embedding for all points

Original Representation		Transformed representation
data point		projections
x_i	$\rightarrow$	$(f_2(i), \ldots, f_{d+1}(i))$
(D-dimensional vector)		(d-dimensional vector)

Understanding Laplacian Eigenmaps

- Best projection onto a 1-dim space
  - Put all points in one place (1st eigenvector – all 1s)
  - If two points are close on graph, their embedding is close (eigenvector values are similar – captured by smoothness of eigenvectors)





Laplacian eigenvectors of swiss roll example (for large # data points)

Step 2 – Embed using Graph Laplacian

- Justification – points connected on the graph stay as close as possible after embedding

$$\min_{\mathbf{f}} \sum_{ij} w_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2 \equiv \min_{\mathbf{f}} \mathbf{f}^T \mathbf{L} \mathbf{f}$$

RHS = $f^T(D\text{-}W)\, f \;=\; f^T D\, f - f^T W\, f = \sum_i d_i f_i^2 - \sum_{i,j} f_i f_j w_{ij}$

$$= \frac{1}{2}\left(\sum_i \Big(\sum_j w_{ij}\Big) f_i^2 - 2\sum_{ij} f_i f_j w_{ij} + \sum_j \Big(\sum_i w_{ij}\Big) f_j^2\right)$$

$$= \frac{1}{2}\sum_{ij} w_{ij}(f_i - f_j)^2$$ **= LHS**

Step 2 – Embed using Graph Laplacian

- Justification – points connected on the graph stay as close as possible after embedding

$$\min_{\mathbf{f}} \sum_{ij} w_{ij}(\mathbf{f}_i - \mathbf{f}_j)^2 \equiv \min_{\mathbf{f}} \mathbf{f}^T \mathbf{L} \mathbf{f} \qquad s.t.\ \mathbf{f}^T \mathbf{D} \mathbf{f} = 1$$

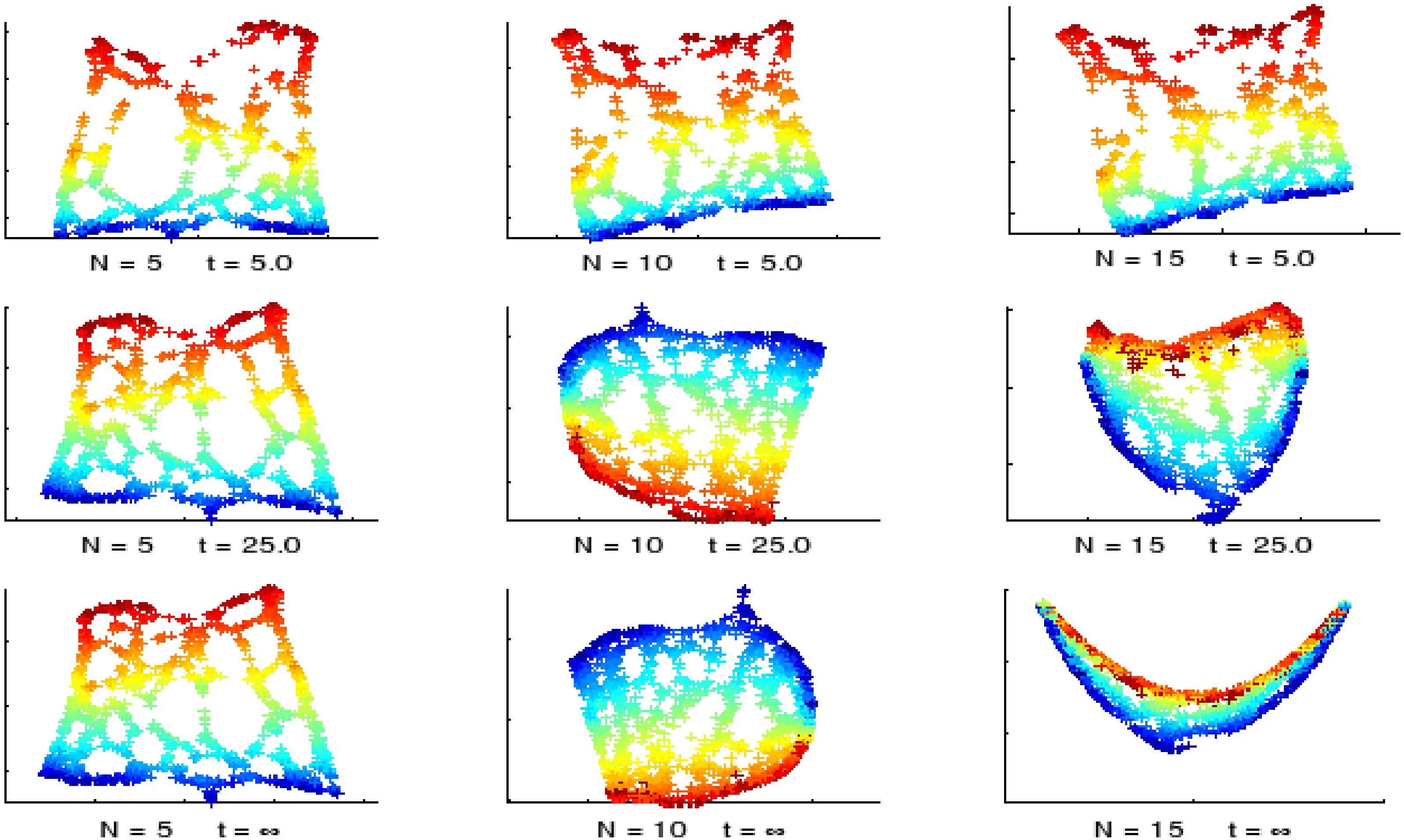
constraint removes arbitrary scaling factor in embedding

Lagrangian: $\min_{\mathbf{f}} \mathbf{f}^{\mathbf{T}} \mathbf{L} \mathbf{f} - \lambda \mathbf{f}^{\mathbf{T}} \mathbf{D} \mathbf{f}$

Wrap constraint into the objective function

$\partial / \partial \mathbf{f} = 0$ $\qquad (\mathbf{L} - \lambda \mathbf{D})\mathbf{f} = 0$ $\qquad \boxed{\mathbf{L}\mathbf{f} = \lambda \mathbf{D} \mathbf{f}}$

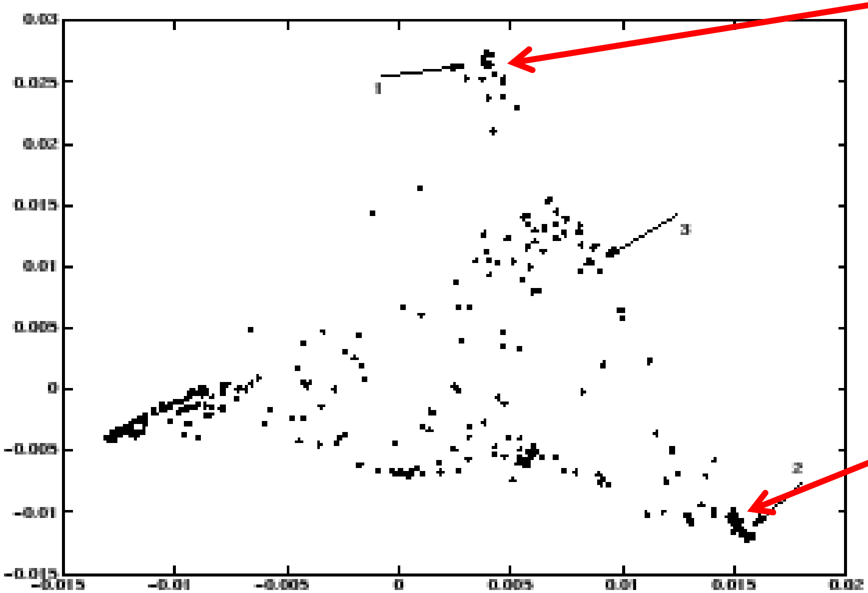
Example – Unrolling the swiss roll

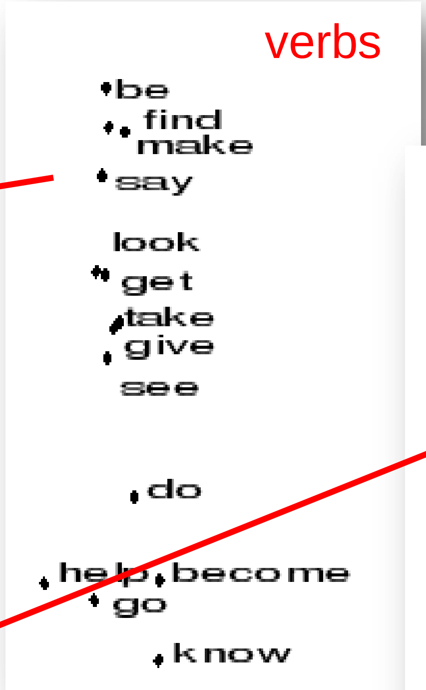


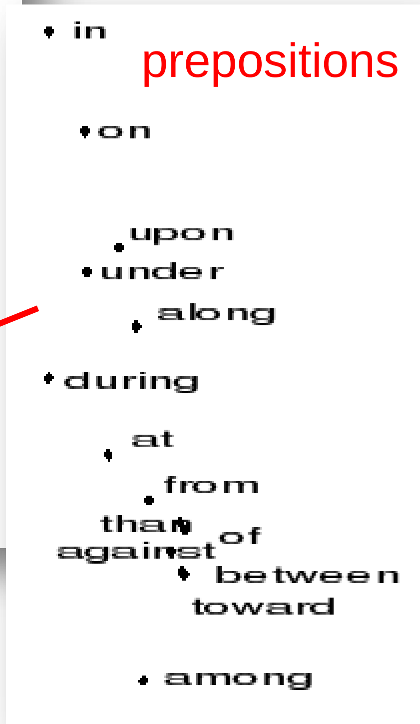
N=number of nearest neighbors, t = the heat kernel parameter (Belkin & Niyogi'03)

Example – Understanding syntactic structure of words

- 300 most frequent words of Brown corpus
- Information about the frequency of its left and right neighbors (600 Dimensional space.)
- The algorithm run with N = 14, t = 1







PCA vs. Laplacian Eigenmaps

PCA	Laplacian Eigenmaps
Linear embedding	Nonlinear embedding
based on largest eigenvectors of D x D correlation matrix $\Sigma = XX^T$ between features	based on smallest eigenvectors of n x n Laplacian matrix $L = D - W$ between data points
eigenvectors give latent features - to get embedding of points, project them onto the latent features	eigenvectors directly give embedding of data points
$\underbrace{x_i}_{D \times 1} \rightarrow \underbrace{[v_1^T x_i, v_2^T x_i, \ldots v_d^T x_i]^T}_{d \times 1}$	$\underbrace{x_i}_{D \times 1} \rightarrow \underbrace{[f_2(i), \ldots, f_{d+1}(i)]^T}_{d \times 1}$

Dimensionality Reduction Methods

- Feature Selection - Only a few features are relevant to the learning task

  Score features (mutual information, prediction accuracy, domain knowledge)
  Regularization

- Latent features – Some linear/nonlinear combination of features provides a more efficient representation than observed feature

  Linear: Low-dimensional linear subspace projection

  PCA (Principal Component Analysis),
  MDS (Multi Dimensional Scaling),
  Factor Analysis, ICA (Independent Component Analysis)

  Nonlinear: Low-dimensional nonlinear projection that preserves local information along the manifold

  Laplacian Eigenmaps
  ISOMAP, Kernel PCA, LLE (Local Linear Embedding),
  Many, many more …