Dimensionality Reduction

April 29, 2014

Machine Learning CSE512

www.cs.stonybrook.edu/~cse512

Stony Brook University

Computer Science

Leman Akoglu
Stony Brook University

Slides Courtesy: Carlos Guestrin
Aarti Singh

High-Dimensional data

- High-Dimensions = Lot of Features

- Document classification

  Features per document = thousands of words/unigrams millions of bigrams, contextual information

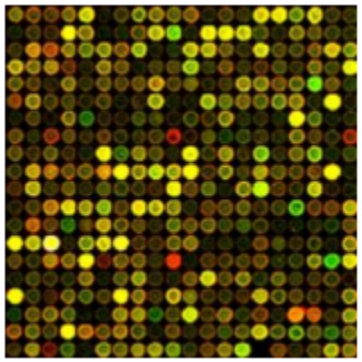
- Surveys - Netflix

  480189 users x 17770 movies

	movie 1	movie 2	movie 3	movie 4	movie 5	movie 6
Tom	5	?	?	1	3	?
George	?	?	3	1	2	5
Susan	4	3	1	?	5	1
Beth	4	3	?	2	4	2

High-Dimensional data

- High-Dimensions = Lot of Features



- Discovering gene networks

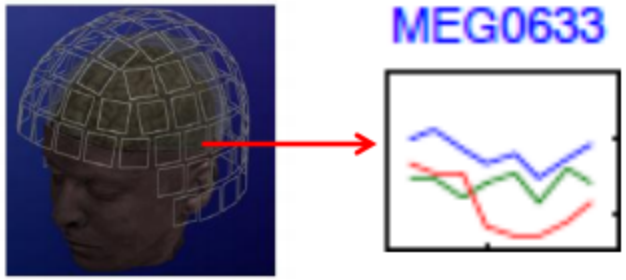
  10,000 genes x 1000 drugs

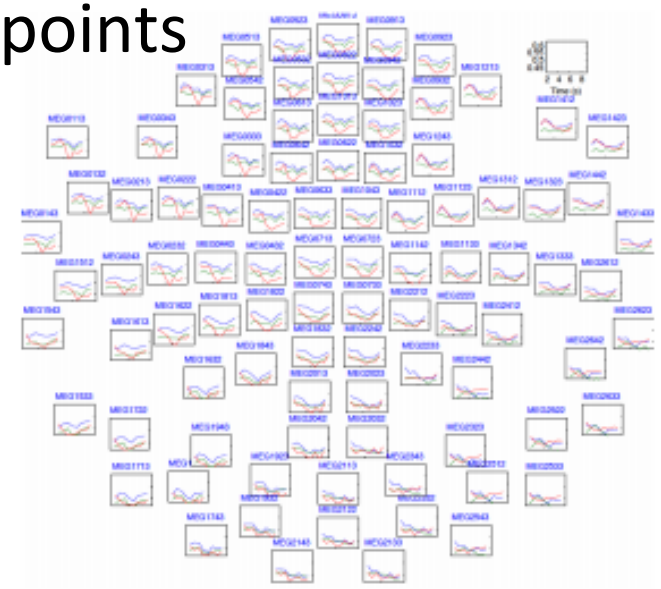
  x several species

- MEG Brain Imaging

  120 locations x 500 time points

  x 20 objects



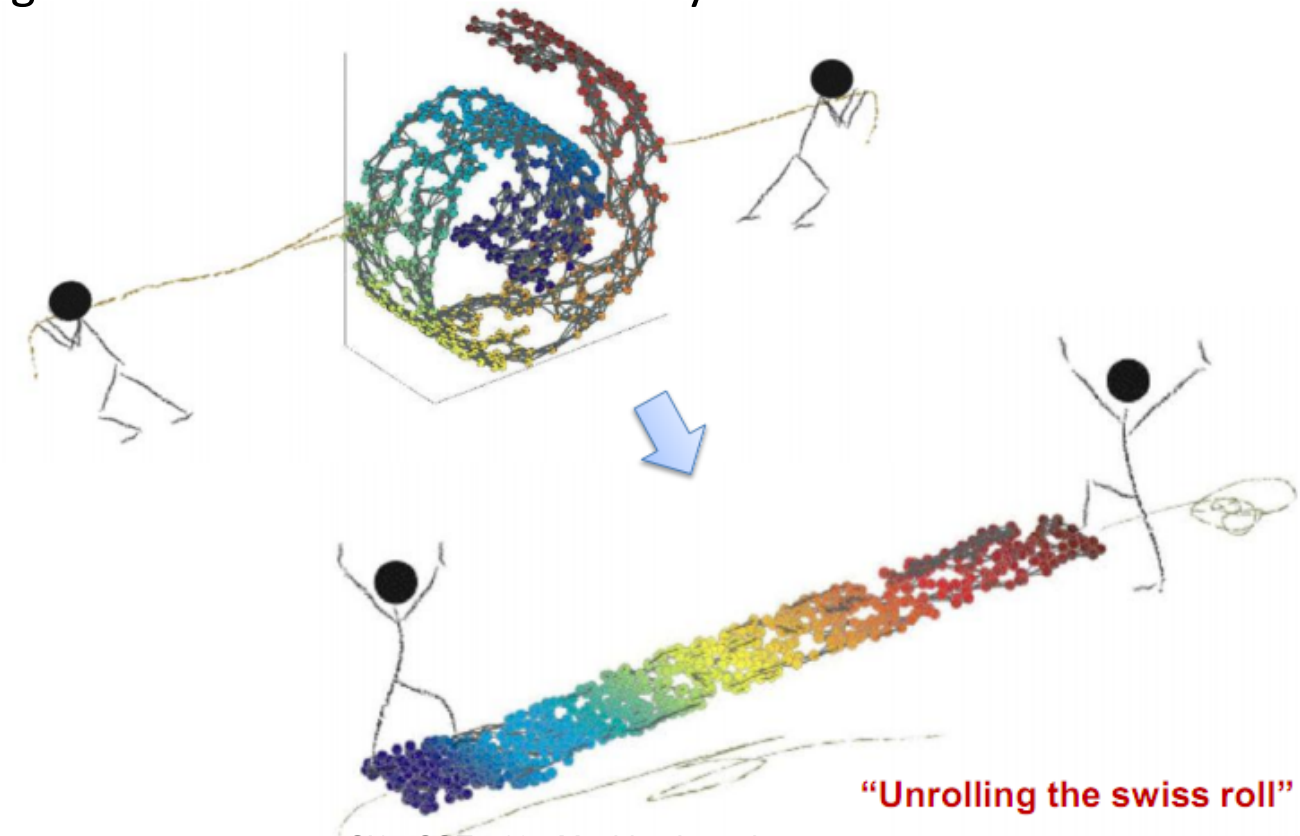


Curse of Dimensionality

- Why are more features bad?
  - Redundant features (not all words are useful to classify a document) more noise added than signal
  - Hard to interpret and visualize
  - Hard to store and process data (computationally challenging)
  - Complexity of decision rule tends to grow with # features. Hard to learn complex rules as VC dimension increases (statistically challenging)

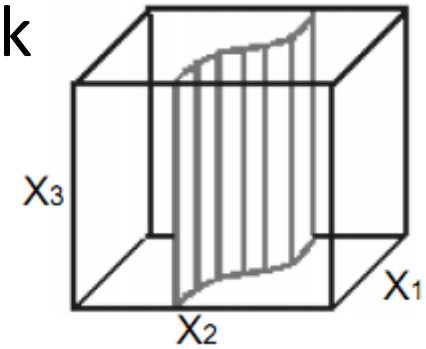
Dimensionality reduction

- Represent data with fewer dimensions
  - easier learning – fewer parameters
  - visualization – hard to visualize more than 3D
  - discover "intrinsic dimensionality" of data
    - high dimensional data that is truly lower dimensional



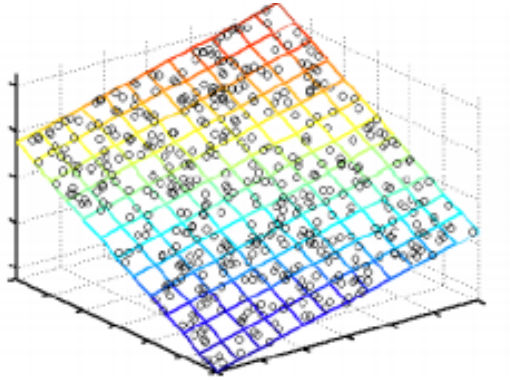
Dimensionality reduction – 2 stories

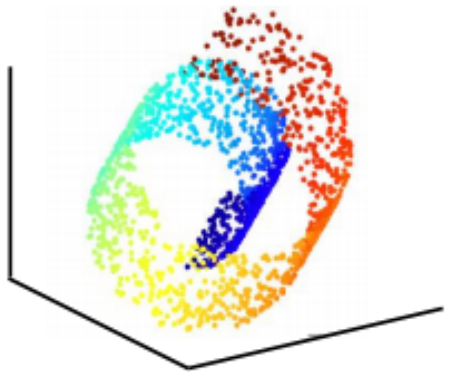
- Feature Selection – Only a few features are relevant to the learning task



X3 - Irrelevant

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





Feature selection

- Want to learn f:**X**→Y
  - **X** = <X_1,…,X_n>
  - but some features are more important than others

- **Approach**: select subset of features to be used by learning algorithm
  - **Score** each feature (or sets of features)
  - **Select** set of features with best score

Feature selection

- Approach 1: **Score each feature and extract a**

Common scoring methods:

- Training or cross-validated accuracy of single-feature classifiers f_i: $X_i \rightarrow Y$
- Estimated mutual information between X_i and Y:

$$\hat{I}(X_i, Y) = \sum_k \sum_y \hat{P}(X_i = k, Y = y) \log \frac{\hat{P}(X_i = k, Y = y)}{\hat{P}(X_i = k)\hat{P}(Y = y)}$$

- χ^2 statistic to measure independence between X_i and Y
- Domain specific criteria
  - Text: Score "stop" words ("the", "of", …) as zero
  - fMRI: Score voxel by T-test for activation versus rest condition
  - …

Feature selection

- Approach 1: **Score each feature and extract a subset**

Common subset selection methods:

- One step: Choose d highest scoring features
- It
  - Choose single highest scoring feature X_k
  - Rescore all features, conditioned on the set of already-selected features
    - E.g., Score(X_i | X_k) = I(X_i,Y |X_k)
    - E.g, Score(X_i | X_k) = Accuracy(predicting Y from X_i and X_k)
  - Repeat, calculating new scores on each iteration, conditioning on set of selected features

Simple greedy forward feature selection algorithm

- Pick a dictionary of features
  - e.g., polynomials for linear regression
- Greedy heuristic:
  - Start from empty (or simple) set of features $F_0 = \emptyset$
  - Run learning algorithm for current set of features F_t
    - Obtain h_t
  - Select **next best feature X_i**
    - e.g., X_j that results in lowest cross-validation error when learning with $F_t \cup \{X_j\}$
  - $F_{t+1} \leftarrow F_t \cup \{X_i\}$
  - Recurse

Simple greedy backward feature selection algorithm

- Pick a dictionary of features
  - e.g., polynomials for linear regression
- Greedy heuristic:
  - Start from all features $F_0 = F$
  - Run learning algorithm for current set of features F_t
    - Obtain h_t
  - Select **next worst feature X_i**
    - e.g., X_j that results in lowest cross-validation error when learning with $F_t - \{X_j\}$
  - $F_{t+1} \leftarrow F_t - \{X_i\}$
  - Recurse

Impact of feature selection on classification of fMRI data [Pereira et al. '05]

Accuracy classifying category of word read by subject

#voxels	mean	subjects 233B	329B	332B	424B	474B	496B	77B	86B
50	0.735	0.783	0.817	0.55	0.783	0.75	0.8	0.65	0.75
100	0.742	0.767	0.8	0.533	0.817	0.85	0.783	0.6	0.783
200	0.737	0.783	0.783	0.517	0.817	0.883	0.75	0.583	0.783
300	**0.75**	**0.8**	**0.817**	**0.567**	**0.833**	**0.883**	**0.75**	**0.583**	**0.767**
400	0.742	0.8	0.783	0.583	0.85	0.833	0.75	0.583	0.75
800	0.735	0.833	0.817	0.567	0.833	0.833	0.7	0.55	0.75
1600	0.698	0.8	0.817	0.45	0.783	0.833	0.633	0.5	0.75
all (~2500)	0.638	0.767	0.767	0.25	0.75	0.833	0.567	0.433	0.733

Table 1: **Average accuracy across all pairs of categories, restricting the procedure to use a certain number of voxels for each subject.** The highlighted line corresponds to the best mean accuracy, obtained using 300 voxels.

Voxels scored by p-value of regression to predict voxel value from the task

Impact of feature selection on text classification [Rogati & Yang '02]

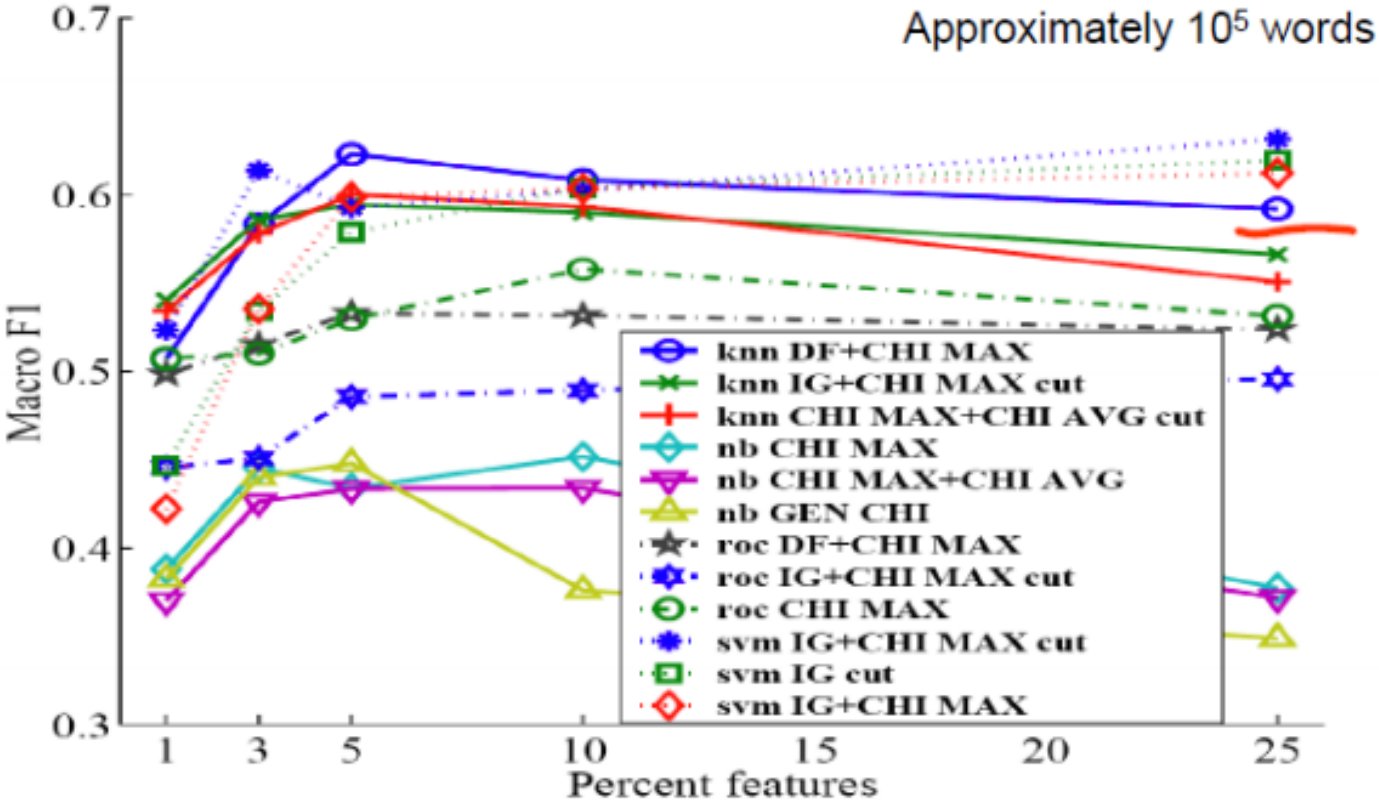


Figure 2: Top 3 feature selection methods for Reuters-21578 (Macro F1)

IG=information gain, chi= χ^2 , DF=doc frequency,

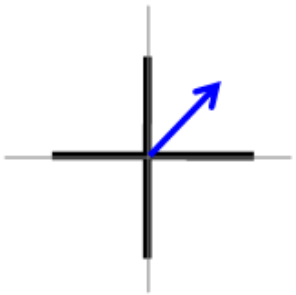
Feature selection

- Approach 2: **Regularization (MAP)**
  - Integrate feature selection into learning objective by penalizing number of features with non-zero weights

$$\widehat{W} = \arg\min_W \sum_{i=1}^{n} -\log P(Y_i|X_i;W) + \lambda\|W\|$$

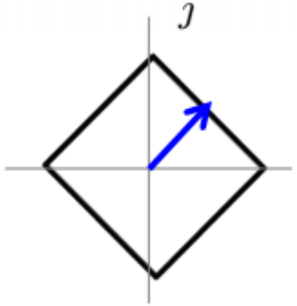
-ve log likelihood — penalty

$\|W\|_0 = \#\{W_j > 0\}$



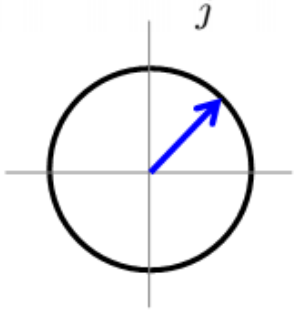
Minimizes # features chosen

$\|W\|_1 = \sum_j |W_j|$



Convex compromise

$\|W\|_2 = \sum_j W_j^2$

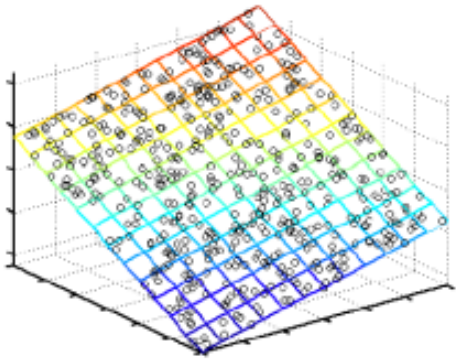


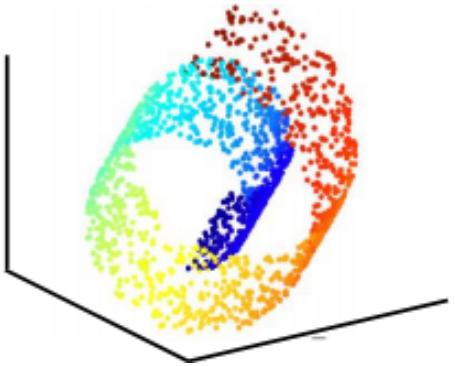
Small weights of features chosen

Latent Feature Extraction

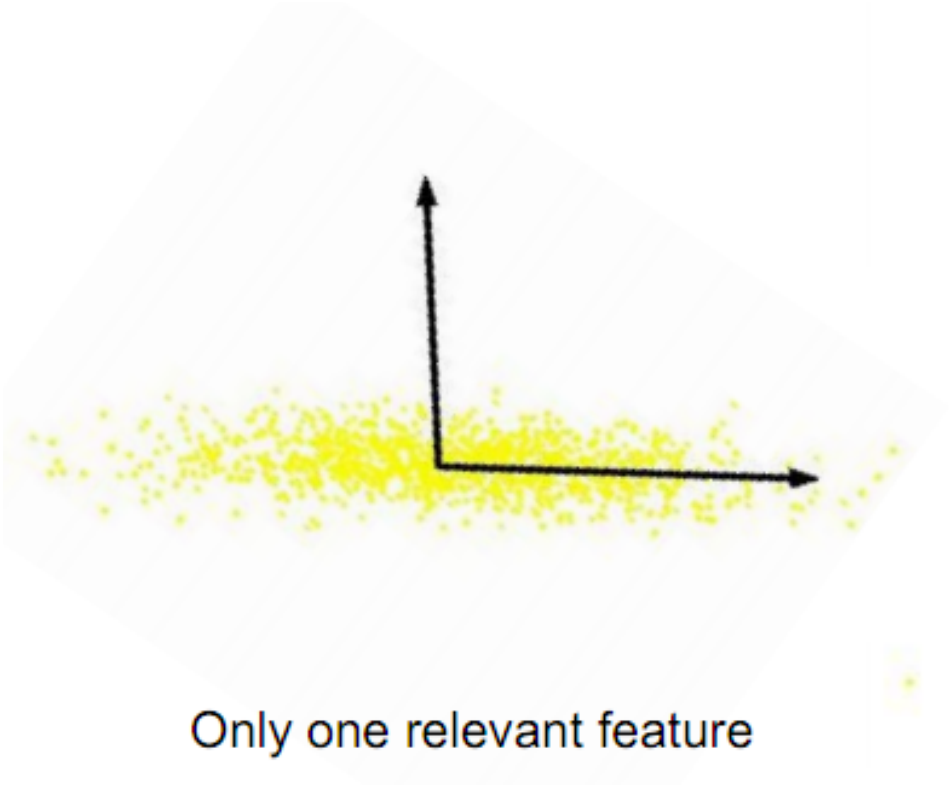
- Combinations of features provide more efficient representation, and capture underlying relations that govern the data
  - E.g. Ego, personality, and intelligence are hidden attributes that characterize human behavior instead of survey questions
  - E.g. Topics (sports, science, news, etc.) instead of words
- Often may not have physical meaning

- Linear
  - **Principal Component Analysis (PCA)**
  - Factor Analysis
  - Independent Component Analysis (ICA)
- Nonlinear
  - **Laplacian Eigenmaps**
  - ISOMAP
  - Local Linear Embedding (LLE)

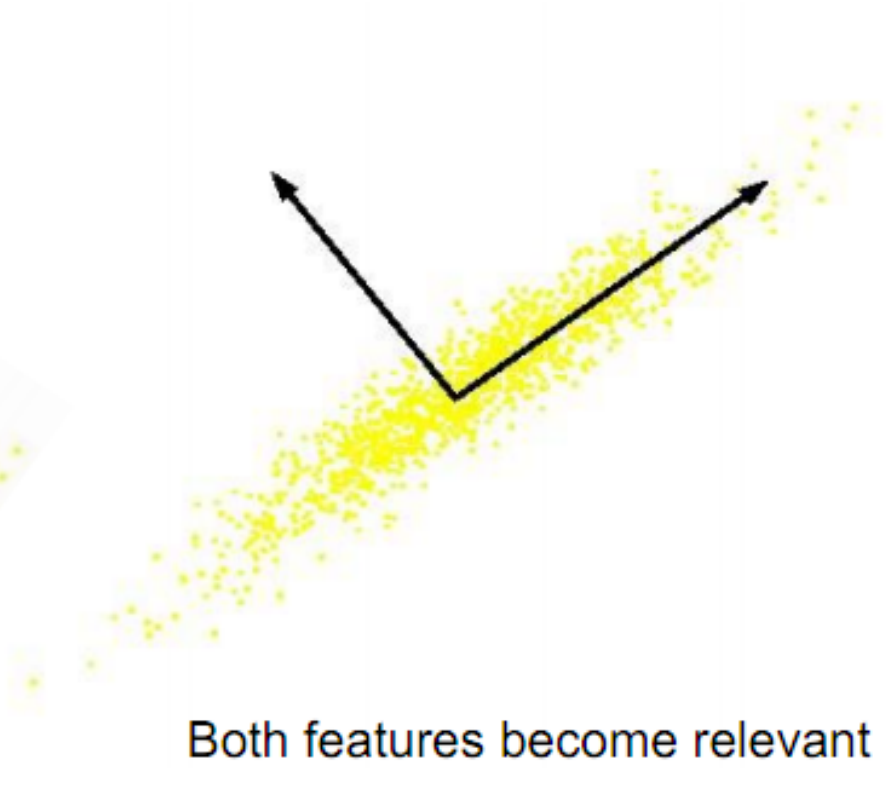




Principal Component Analysis (PCA)



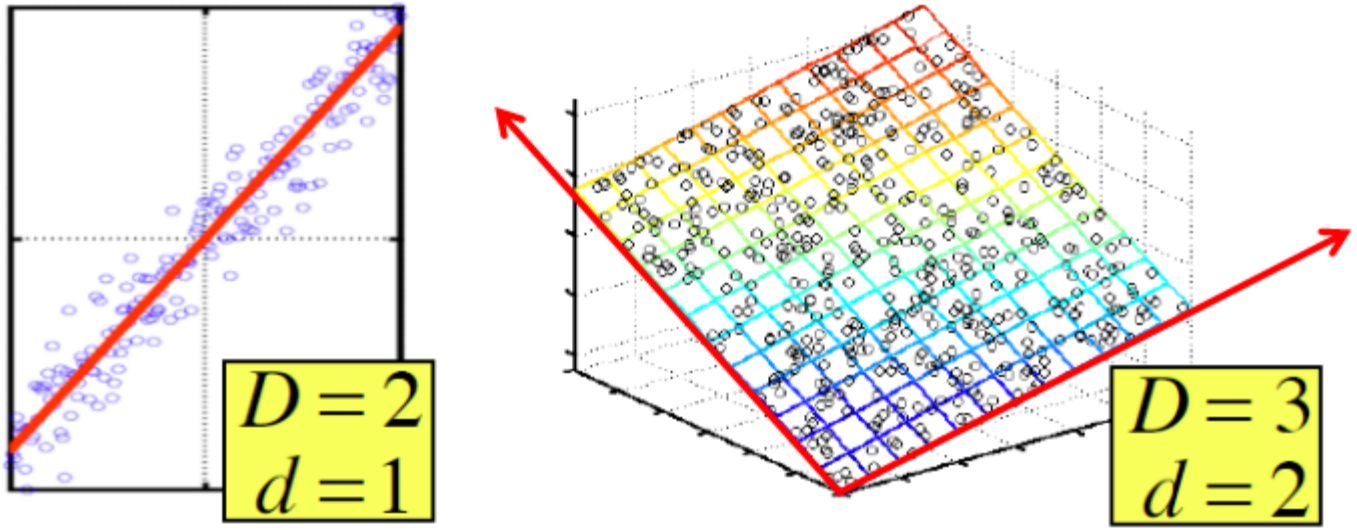
Only one relevant feature



Both features become relevant

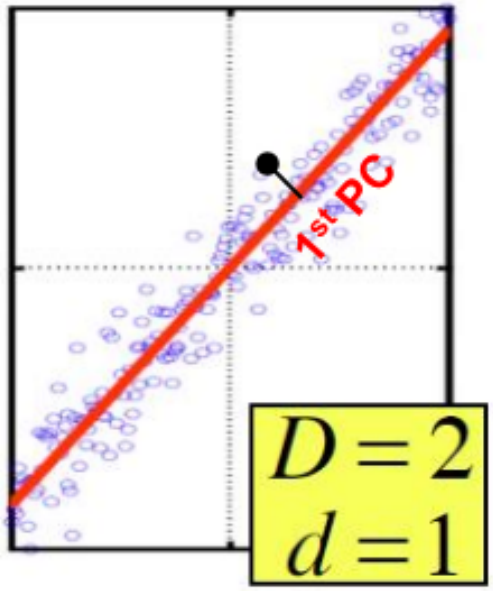
Can we transform the features so that we only need to preserve one latent feature? Find linear projection so that projected data is uncorrelated.

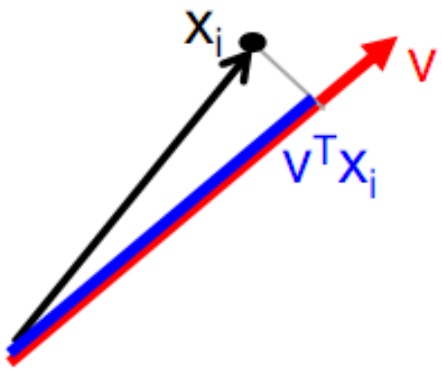
Principal Component Analysis (PCA)



- Assumption: Data lies on or near a low d-dimensional linear subspace.
- Axes of this subspace are effective representation of the data
- Identifying the axes is known as Principal Components Analysis, and can be obtained by Eigen or Singular value decomposition

Principal Component Analysis (PCA)





Principal Components (PC) are orthogonal directions that capture most of the variance in the data

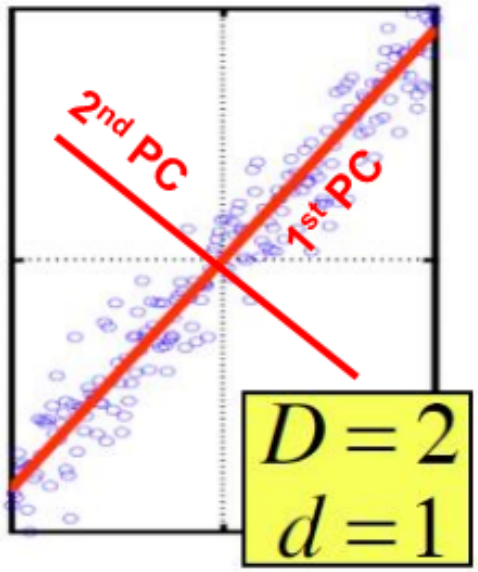
1st PC – direction of greatest variability in data

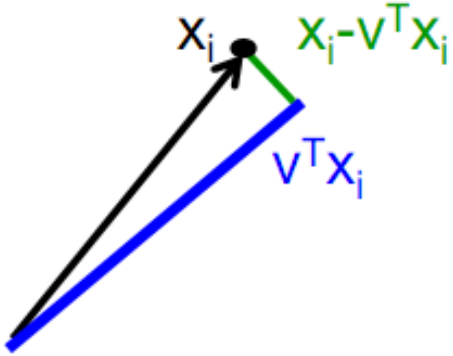
Projection of data points along 1st PC discriminate the data most along any one direction

Take a data point x_i (D-dimensional vector)

Projection of x_i onto the 1st PC v is $v^T x_i$

Principal Component Analysis (PCA)





Principal Components (PC) are orthogonal directions that capture most of the variance in the data

1st PC – direction of greatest variability in data

2nd PC – Next orthogonal (uncorrelated) direction of greatest variability

(remove all variability in first direction, then find next direction of greatest variability)

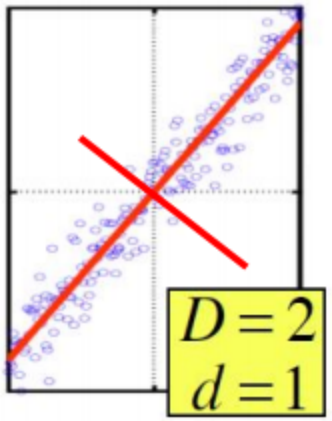
And so on …

Principal Component Analysis (PCA)

Let $v_1, v_2, \ldots, v_d$ denote the principal components

Orthogonal and unit norm $v_i^T v_j = 0 \quad i \neq j$

$v_i^T v_i = 1$



Find vector that maximizes sample variance of projection

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{v}^T\mathbf{x}_i)^2 = \mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v}$$

Assume data are centered

Data points $X = [\, x_1\ x_2\ \ldots\ x_n]$

$$\max_{\mathbf{v}} \ \mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v} \quad \text{s.t.} \quad \mathbf{v}^T\mathbf{v} = 1$$

Lagrangian: $\max_{\mathbf{v}} \mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v} - \lambda\mathbf{v}^T\mathbf{v}$

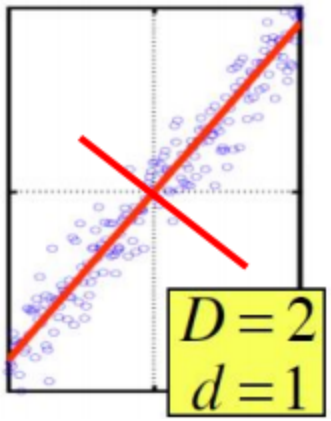
Wrap constraints into the objective function

$$\partial/\partial\mathbf{v} = 0 \qquad (\mathbf{X}\mathbf{X}^T - \lambda\mathbf{I})\mathbf{v} = 0 \qquad \Rightarrow \boxed{(\mathbf{X}\mathbf{X}^T)\mathbf{v} = \lambda\mathbf{v}}$$

Principal Component Analysis (PCA)

$$(\mathbf{X}\mathbf{X}^T)\mathbf{v} = \lambda\mathbf{v}$$

Therefore, v is the eigenvector of sample correlation/ covariance matrix $\mathbf{X}\mathbf{X}^T$



Sample variance of projection $= \mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v} = \lambda\mathbf{v}^T\mathbf{v} = \lambda$

Thus, the eigenvalue λ denotes the amount of variability captured along that dimension (aka amount of energy along that dimension).

Eigenvalues $\lambda_1 > \lambda_2 > \lambda_3 > \ldots$

The 1st Principal component v_1 is the eigenvector of the sample covariance matrix $\mathbf{X}\mathbf{X}^T$ associated with the largest eigenvalue λ_1

The 2nd Principal component v_2 is the eigenvector of the sample covariance matrix $\mathbf{X}\mathbf{X}^T$ associated with the second largest eigenvalue λ_2

And so on …

Computing the PCs

Eigenvectors are solutions of the following equation:

$$(\mathbf{X}\mathbf{X}^T)\mathbf{v} = \lambda\mathbf{v} \qquad (\mathbf{X}\mathbf{X}^T - \lambda\mathbf{I})\mathbf{v} = 0$$

Non-zero solution $v \neq 0$ possible only if

$$\det(\mathbf{X}\mathbf{X}^T - \lambda\mathbf{I}) = 0$$ Characteristic Equation

This is a D^{th} order equation in λ, can have at most D distinct solutions (roots of the characteristic equation)

Once eigenvalues are computed, solve for eigenvectors (Principal Components) using

$$(\mathbf{X}\mathbf{X}^T - \lambda\mathbf{I})\mathbf{v} = 0$$

For symmetric matrices, eigenvectors for distinct eigenvalues are orthogonal.

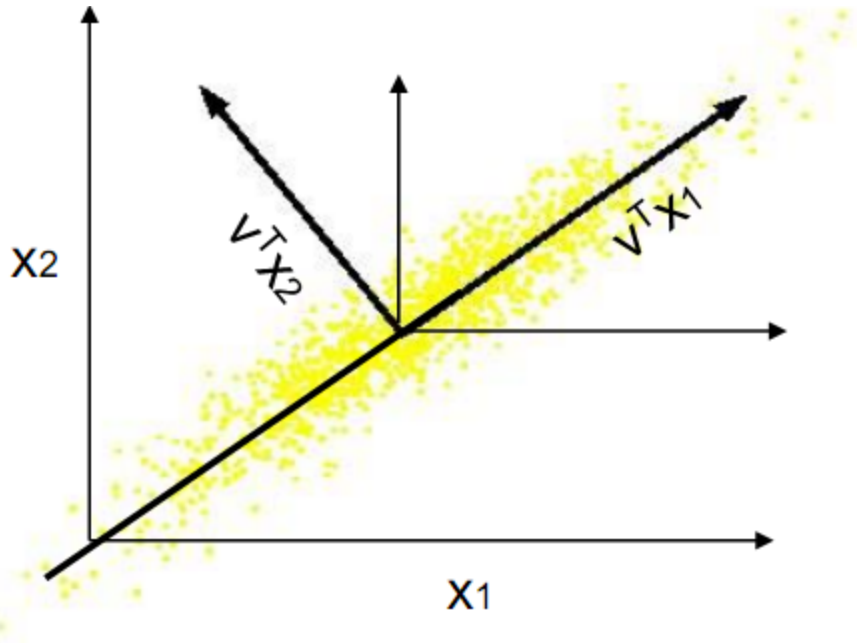
Basic (Naïve) PCA algoritm

- Start from m (features) by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X_c} \leftarrow \mathbf{X} - \mathbf{X}\boldsymbol{\mu}$ (**μ:** empirical mean along each dimension)
- **Compute covariance matrix (m x m)**:
  - $\Sigma \leftarrow 1/m\ \mathbf{X_c}^T \mathbf{X_c}$
- Find **eigen- vectors and values** of Σ
- **Principal components:** k eigen-vectors with highest eigen-values

Principal Component Analysis (PCA)

So, the new axes are the eigenvectors of the matrix of sample correlations XX^T of the data, which capture the similarities of the original features based on how data samples project to the new axes.

Transformed features are uncorrelated.



Geometrically: centering followed by rotation

- Linear transformation

Principal Component Analysis (PCA)

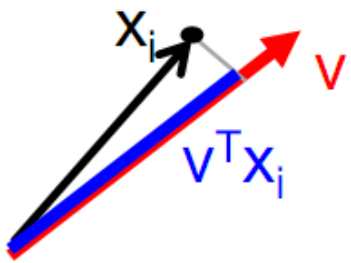
- Another interpretation:

Maximum Variance Subspace: PCA finds vectors v such that projections on to the vectors capture maximum variance in the data

$$\frac{1}{n}\sum_{i=1}^{n}(\mathbf{v}^T\mathbf{x}_i)^2 = \mathbf{v}^T\mathbf{X}\mathbf{X}^T\mathbf{v}$$

Minimum Reconstruction Error: PCA finds vectors v such that projection on to the vectors yields minimum MSE reconstruction

$$\frac{1}{n}\sum_{i=1}^{n}\|\mathbf{x}_i - (\mathbf{v}^T\mathbf{x}_i)\mathbf{v}\|^2$$

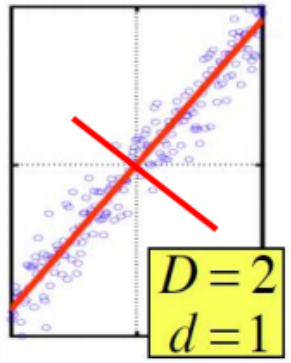


Dimensionality Reduction using PCA

The eigenvalue λ denotes the amount of variability captured along that dimension.

Zero eigenvalues indicate no variability along those directions => data lies exactly on a linear subspace

Only keep data projections onto principal components with non-zero eigenvalues, say $v_1, \ldots, v_d$ where d = rank (XX^T)



Original Representation

data point

$x_i = [x_i^1, x_i^2, \ldots . x_i^D]$

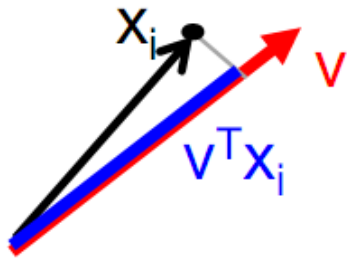
(D-dimensional vector)

Transformed representation

projections

$[v_1^T x_i, v_2^T x_i, \ldots v_d^T x_i]$

(d-dimensional vector)

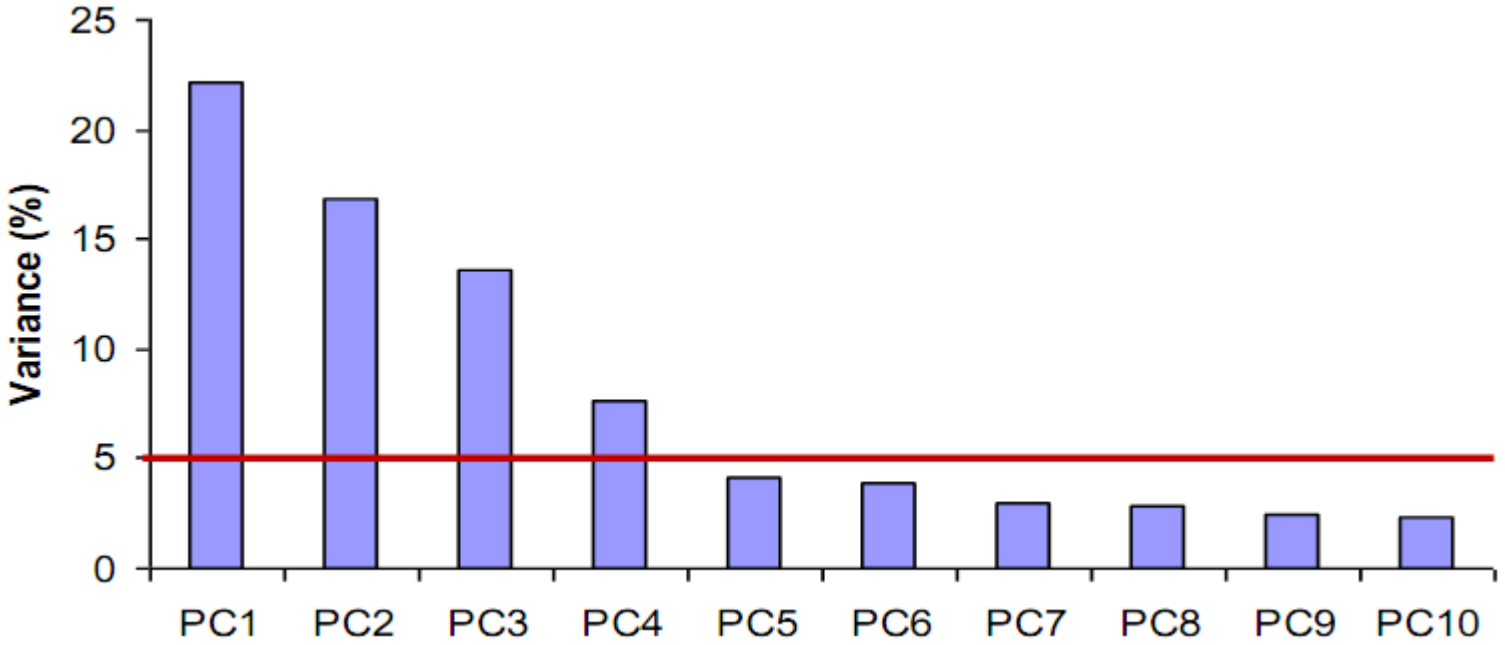


Dimensionality Reduction using PCA

In high-dimensional problem, data usually lies near a linear subspace, as noise introduces small variability

Only keep data projections onto principal components with **large** eigenvalues

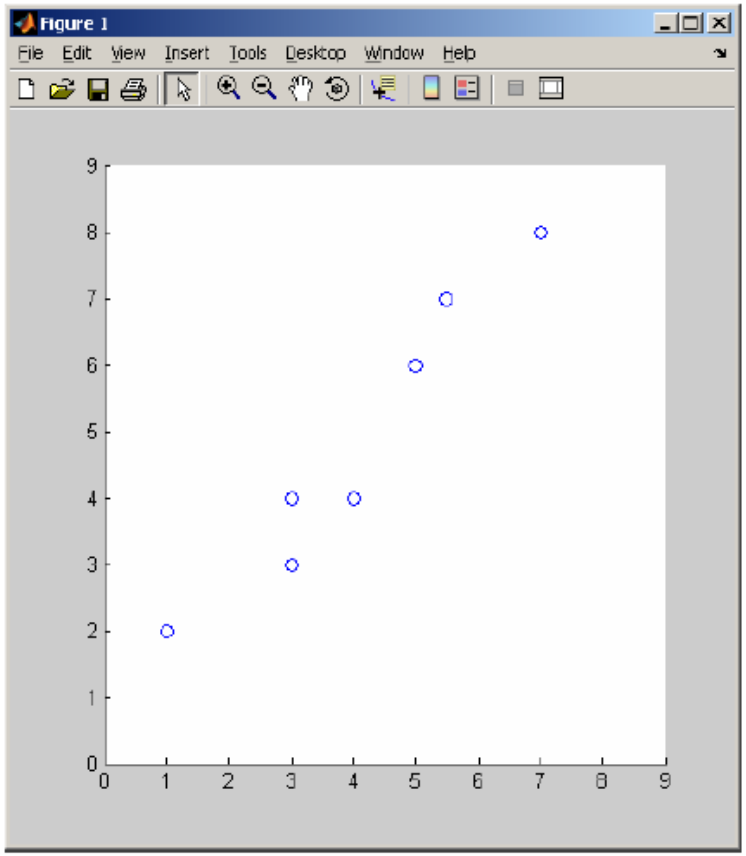
Can *ignore* the components of lesser significance.

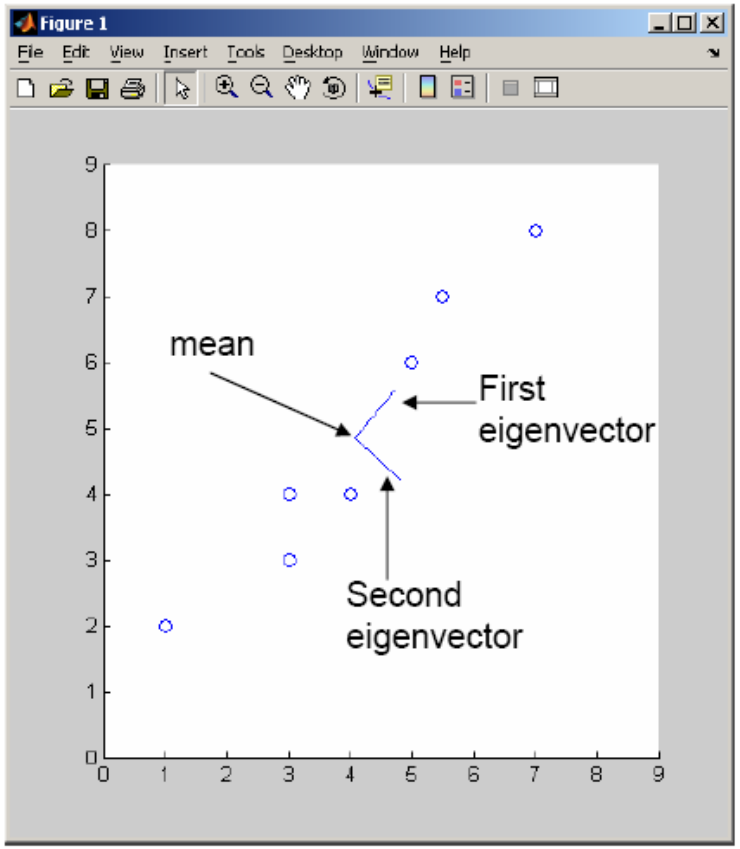


You might lose some information, but if eigenvalues are small, you don't lose much

PCA example

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

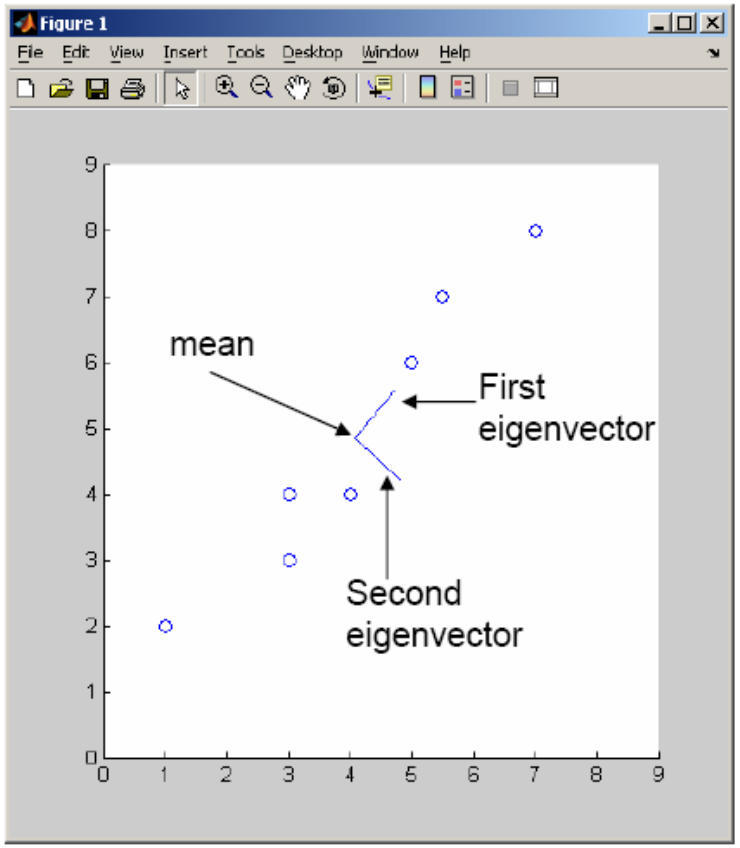


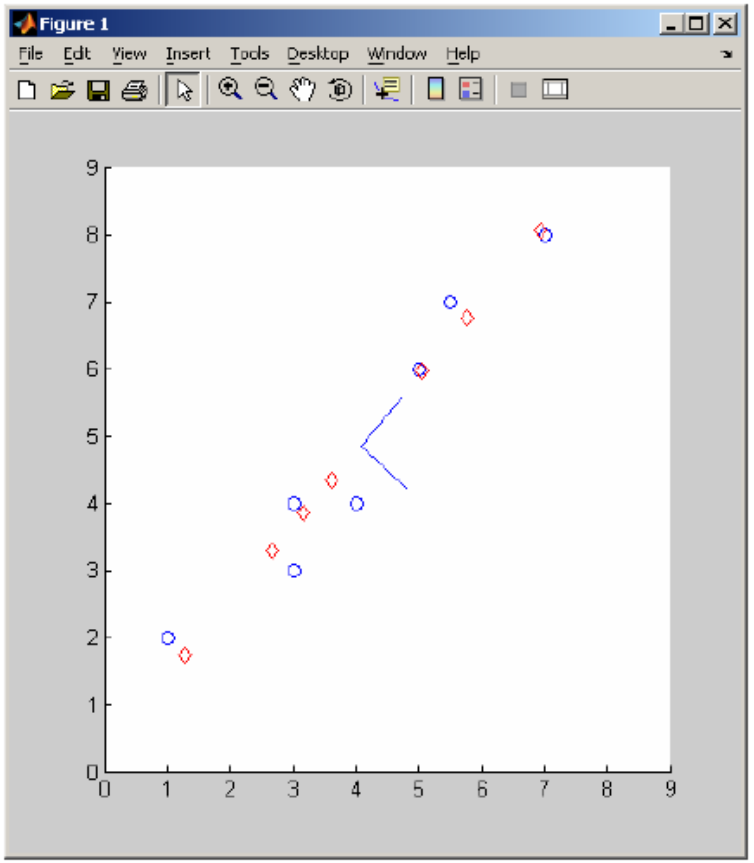


PCA example – reconstruction

$$\hat{\mathbf{x}}^i = \bar{\mathbf{x}} + \sum_{j=1}^{k} z_j^i \mathbf{u}_j$$

only used first principal component

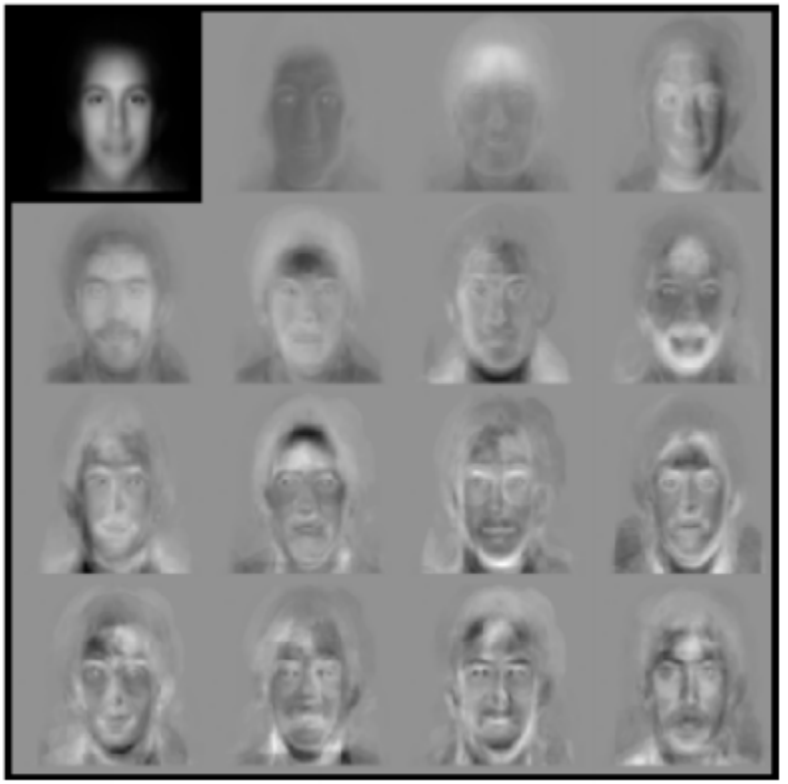




Eigenfaces [Turk, Pentland '91]

- Input images:

- Principal components:



Top left image is
linear combination of rest

Eigenfaces reconstruction

- Each image corresponds to adding 8 principal components:

Scaling up!

- Covariance matrix can be really big!
  - Σ is n by n
  - 10000 features ! $|\Sigma|$
  - finding eigenvectors is very slow…

- Use singular value decomposition (SVD)
  - finds k eigenvectors
  - great implementations available, e.g., Matlab svd(.)

SVD

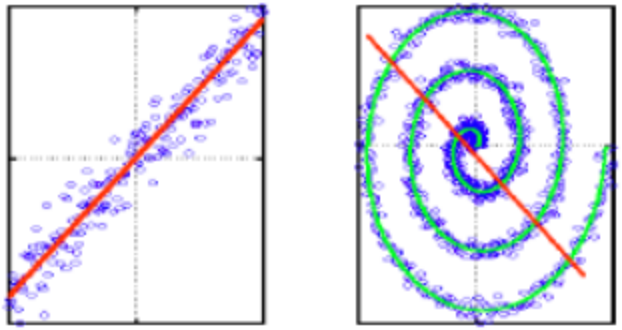
- Write $\mathbf{X} = \mathbf{U}\,\mathbf{S}\,\mathbf{V}^T$
  - $\mathbf{X}$ ← data matrix, one row per data point
  - $\mathbf{U}$ ← weight matrix, one row per data point – coordinate of $\mathbf{x}^i$ in eigen-space
  - $\mathbf{S}$ ← singular value matrix (diagonal)
    - in our setting each entry is eigenvalue λ_j
  - $\mathbf{V}^T$ ← singular vector matrix
    - in our setting each row is eigenvector $\mathbf{v}_j$

PCA using SVD algorithm

- Start from m by n data matrix **X**
- **Recenter**: subtract mean from each row of **X**
  - $\mathbf{X}_c \leftarrow \mathbf{X} - \mathbf{X}\boldsymbol{\mu}$
- Call SVD algorithm on $\mathbf{X}_c$ – ask for k (right) singular vectors
- **Principal components:** k singular vectors with highest singular values (rows of $\mathbf{V}^T$)

Properties of PCA

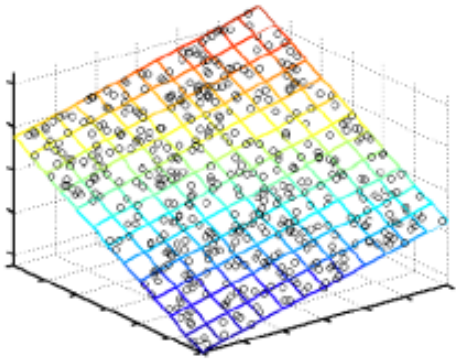
- Strengths (+)
  - Eigenvector method
  - No parameter tuning
  - Non-iterative
  - No local optima
- Weaknesses (-)
  - Limited to linear projections
  - Limited to 2nd order statistics

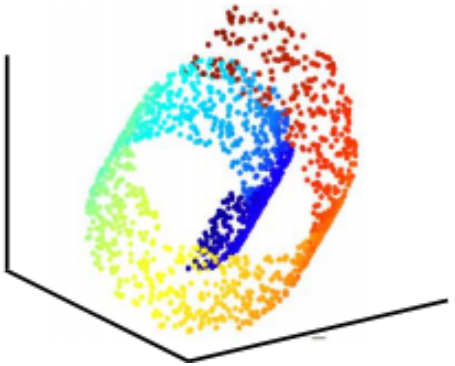


Latent Feature Extraction

- Combinations of features provide more efficient representation, and capture underlying relations that govern the data
  - E.g. Ego, personality, and intelligence are hidden attributes that characterize human behavior instead of survey questions
  - E.g. Topics (sports, science, news, etc.) instead of words
- Often may not have physical meaning

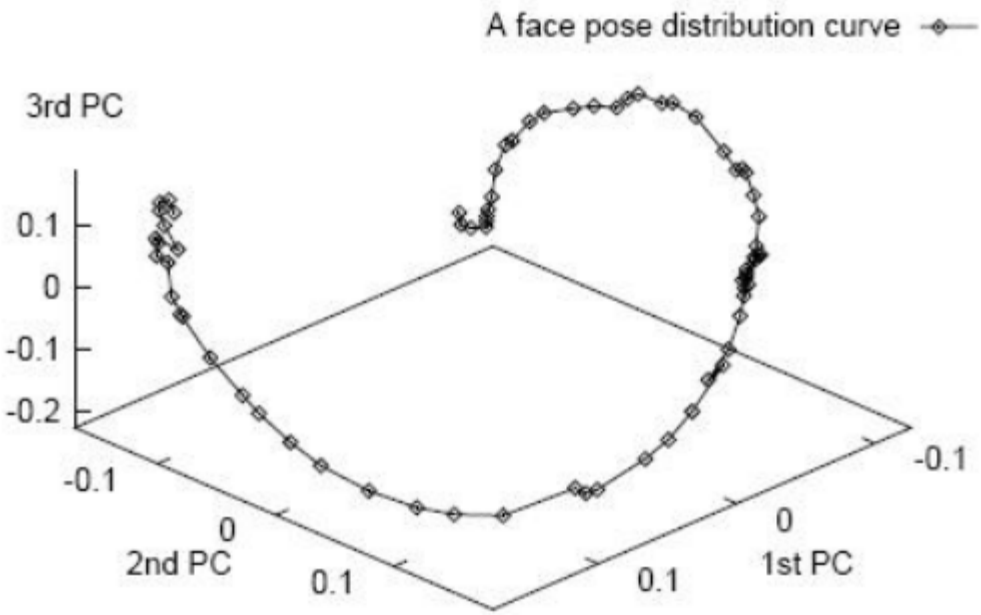
- Linear
  - **Principal Component Analysis (PCA)**
  - Factor Analysis
  - Independent Component Analysis (ICA)
- Nonlinear
  - **Laplacian Eigenmaps**
  - ISOMAP
  - Local Linear Embedding (LLE)





Nonlinear methods

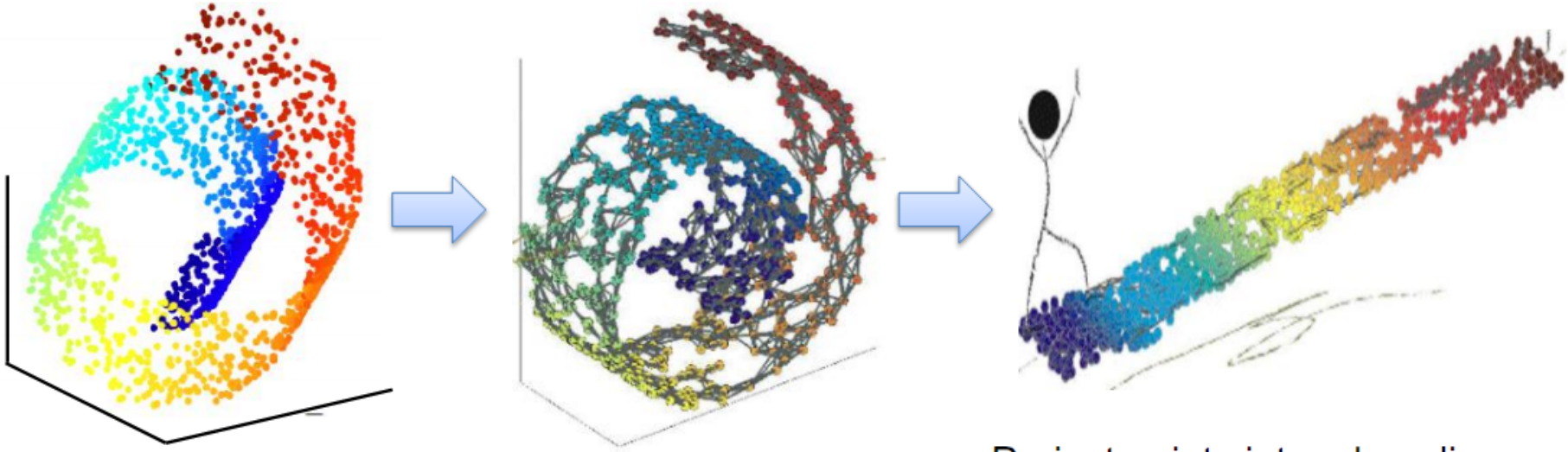
- Data often lies on or near a nonlinear low-dimensional curve aka manifold.



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)

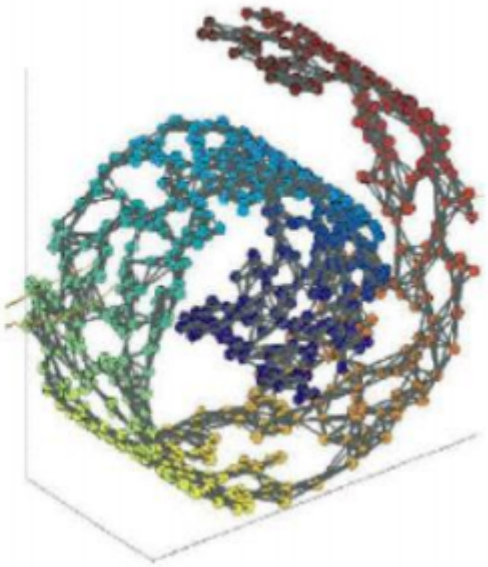
Project points into a low-dim space using "eigenvectors of the graph"

Step 1. Graph construction

Similarity Graphs: Model local neighborhood relations between data points

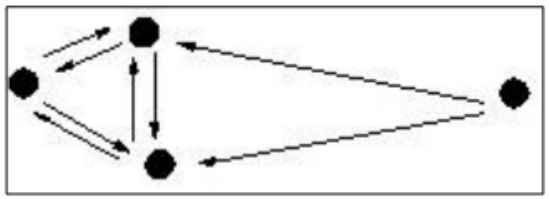
G(V,E) V – Vertices (Data points)

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

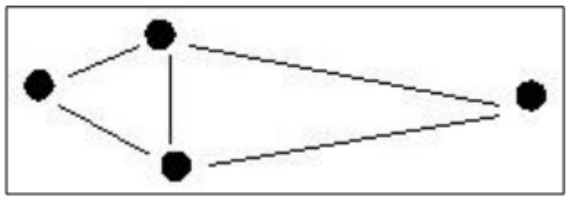


(2) E – Edge if k-NN,
yields directed graph

connect A with B if A → B OR A ← B (symmetric kNN graph)
connect A with B if A → B AND A ← B (mutual kNN graph)



Directed nearest neighbors



(symmetric) kNN graph

mutual kNN graph

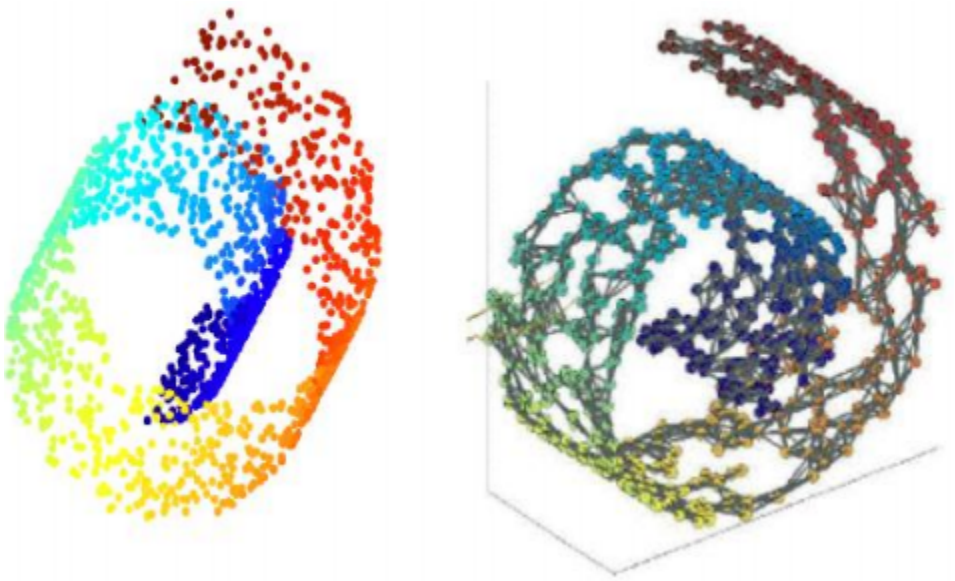
Step 1. Graph construction

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)

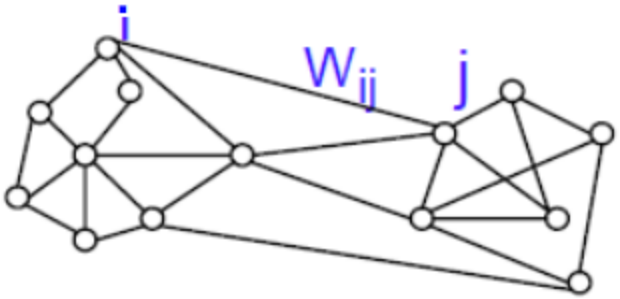
$$W_{ij} = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}$$

$\sigma^2 \to \infty$ 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

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

Step 2. Embed using Graph Laplacian

Graph Laplacian (unnormalized version)

$$L = D - W$$

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

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 $\lambda_2, \lambda_3, \ldots, \lambda_{d+1}$

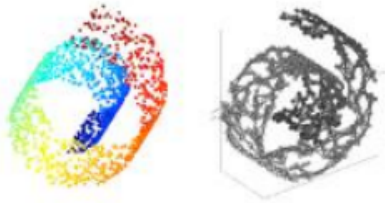
ignore 1st eigenvector – same embedding for all points

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

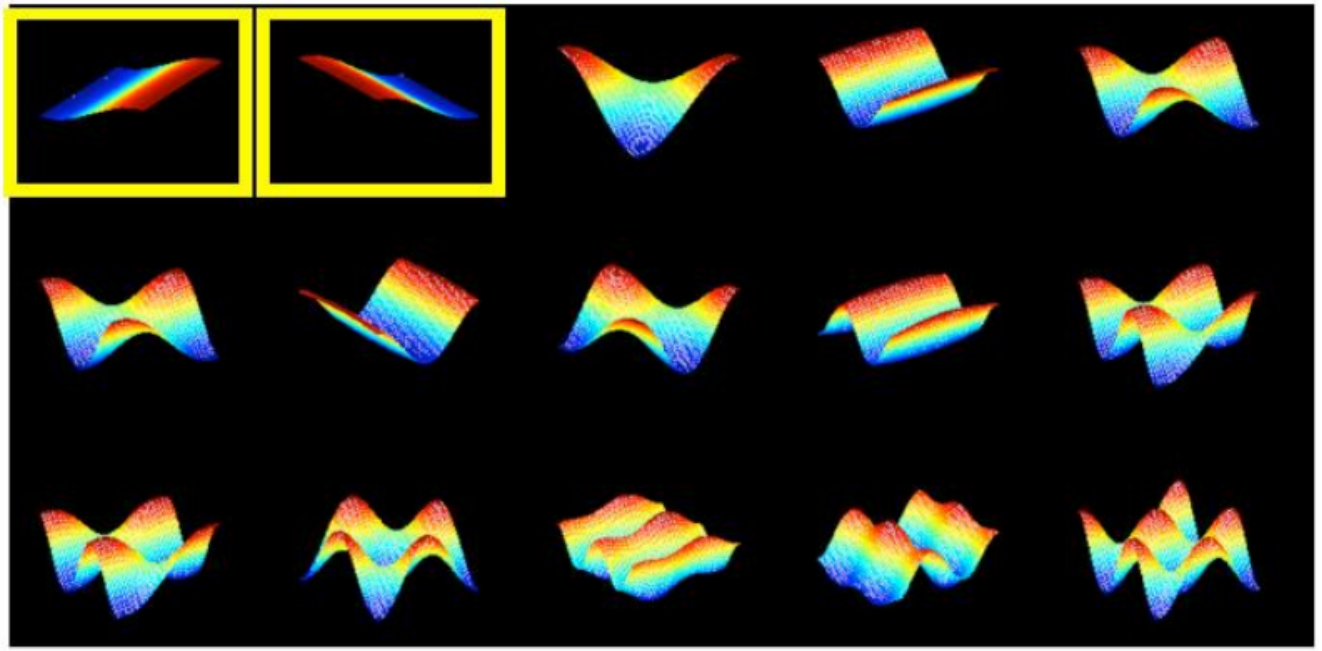
Understanding Laplacian Eigenmaps

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

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

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

Step 2. Embed using Graph Laplacian

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

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

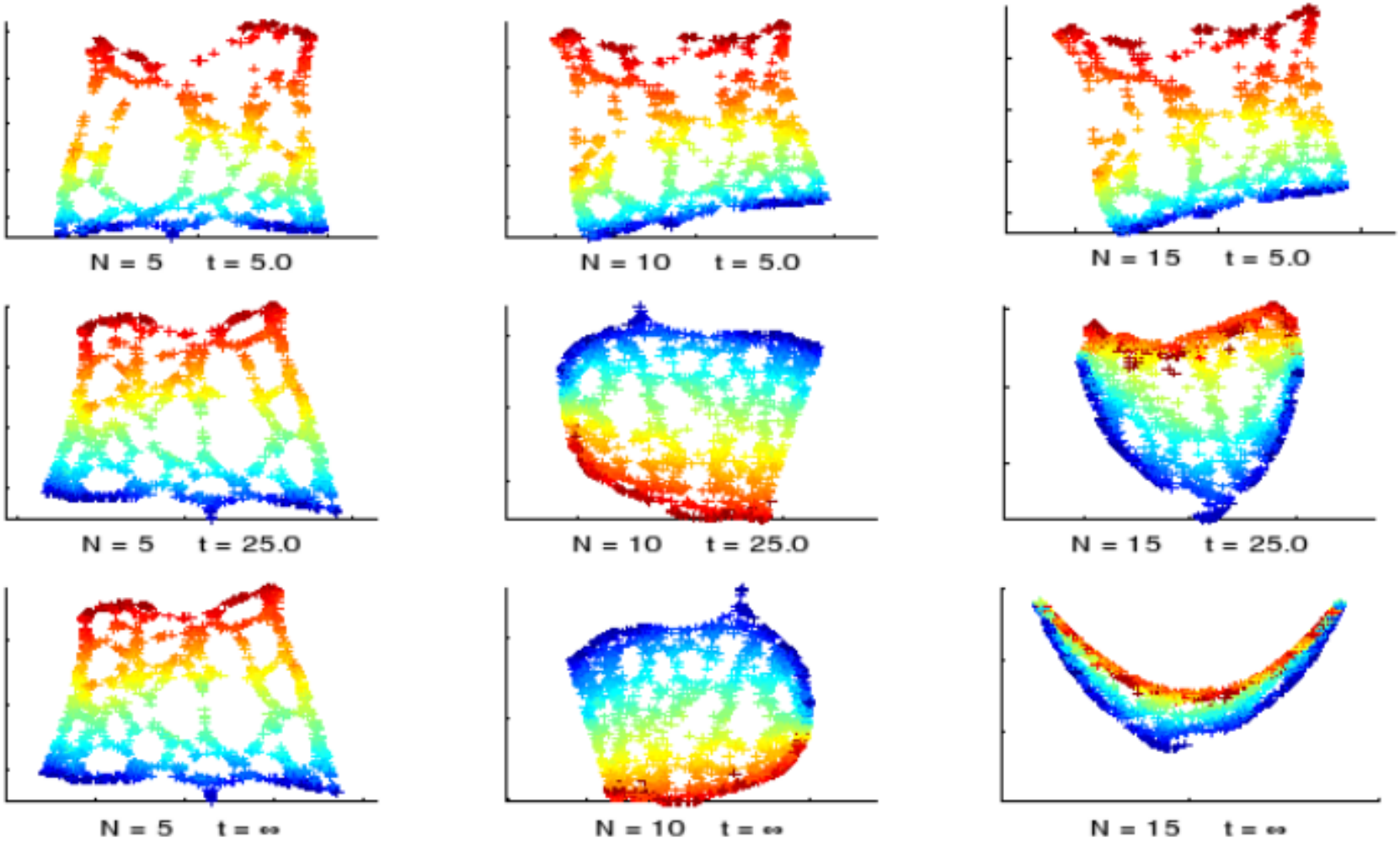
constraint removes arbitrary scaling factor in embedding

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

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

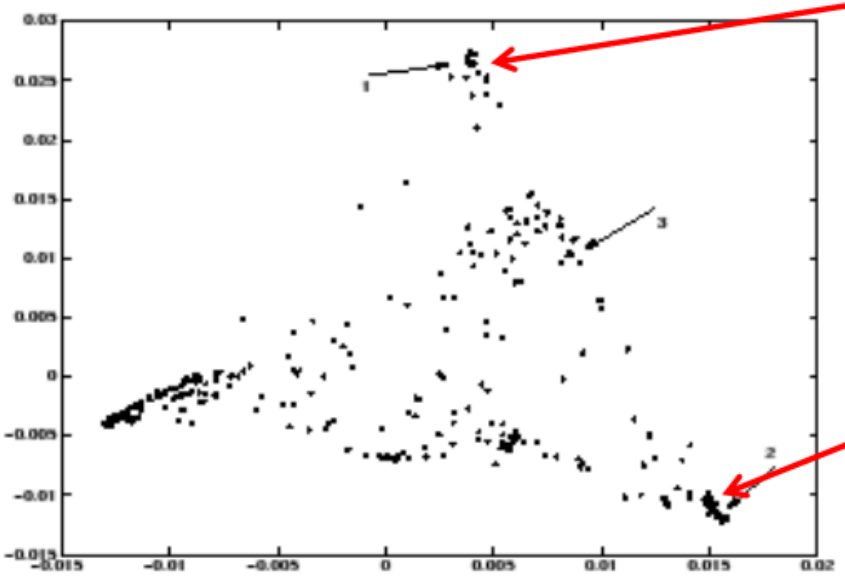
Example – Unrolling the swiss roll

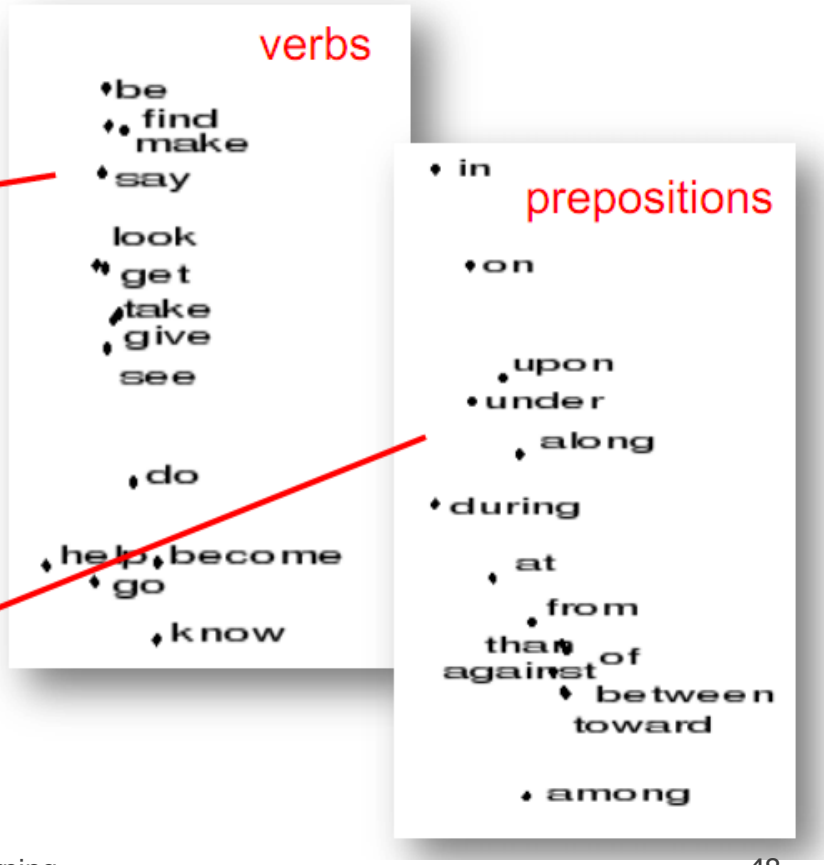


N=number of nearest neighbors, t = the heat kernel parameter (Belkin & Niyogi'03)

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
$x_i \rightarrow [v_1^T x_i, v_2^T x_i, \ldots v_d^T x_i]^T$	$x_i \rightarrow [f_2(i), \ldots, f_{d+1}(i)]^T$
D x1 d x1	D x1 d x1

Summary - Dimensionality Reduction Methods

- Feature Selection – Only a few features relevant to 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 …

What you should know

- Dimensionality reduction
  - why and when it's important
- Simple feature selection
- Principal component analysis (PCA)
  - minimizing reconstruction error
  - relationship to covariance matrix and eigenvectors
  - using SVD
- Nonlinear methods
  - Laplacian eigenmaps
  - Graph construction
  - Laplacian embedding
- PCA vs. Laplacian eigenmaps