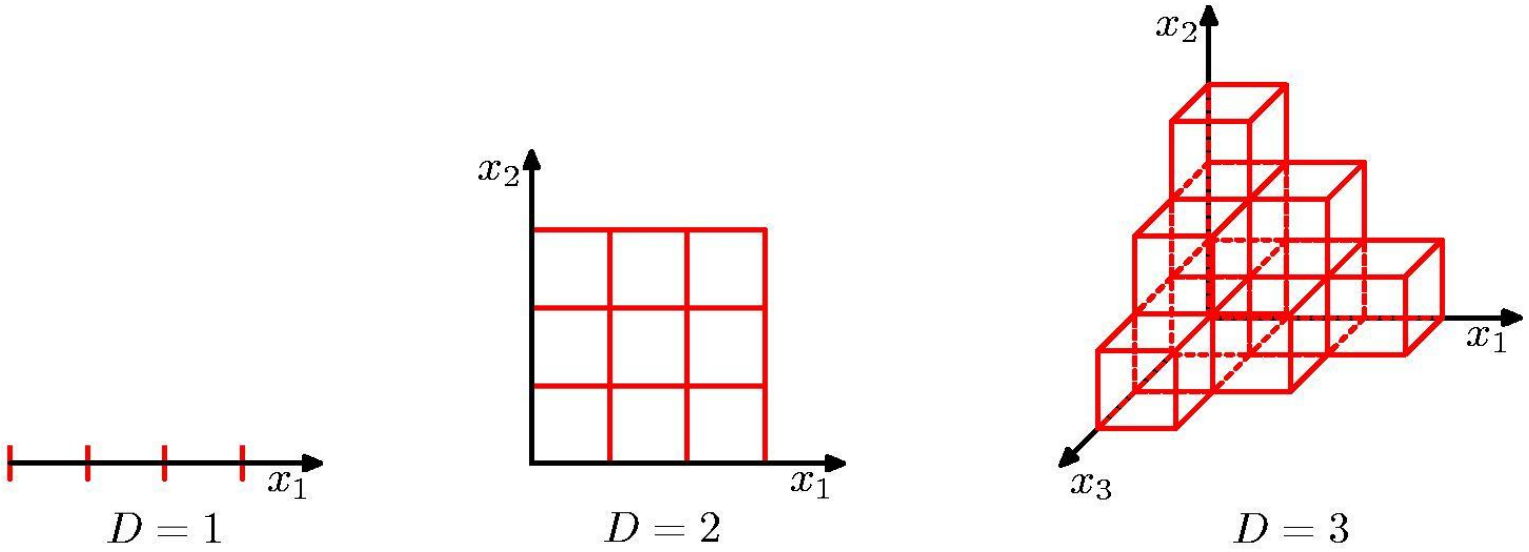
DIMENSIONALITY REDUCTION

Hsin-Min Lu
盧信銘
台大資管系

Curse of Dimensionality (cf. PRML Ch 1.4)

- A typical statistical learning model can learn faster when # of features is lower.
- E.g., linear regression $O(NP^3)$, where N is sample size and P is # of features.
- When # of features are large, some of our intuition may not work well.

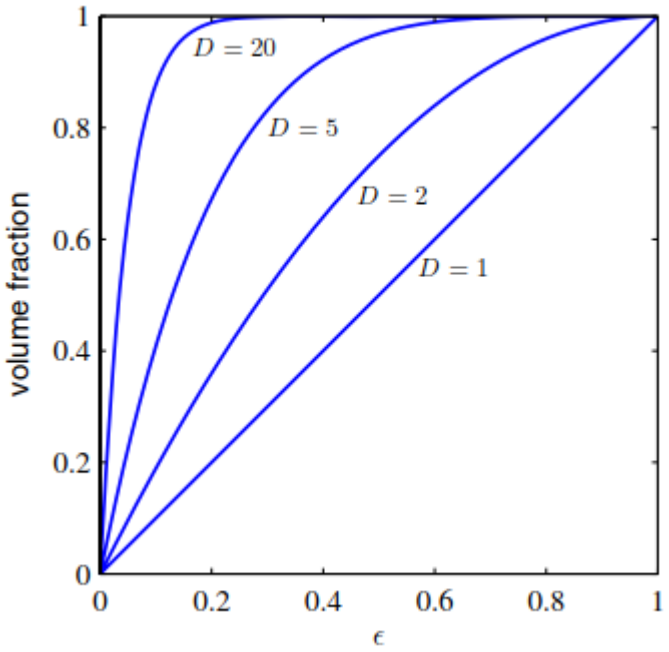


Example 1: Fractions of data points near by

- Consider a sphere of radius $r = 1$ in a space of D dimensions.
- We are interested in knowing the fraction of volume of the sphere that lies between radius $r = 1 - \epsilon$ and $r = 1$.
- Why this question is important?
- We want to know the relation between “distance” and “data volume” because many learning algorithms works based on data points closed by.
- If data points are uniformly distributed in the space, then volume is proportion to number of data points.
- **The question is asking what is the fraction of data points that are very far from us.**

Example 1 (Cont'd.)

- The volume of a sphere of radius r in D dimensions is:
- $V_D(r) = K_D r^D$, where K_D is a constant only depends on D.
- The fraction of volume between $[1-\epsilon, 1]$ is:
- $\frac{V_D(1)-V_D(1-\epsilon)}{V_D(1)} = 1-(1-\epsilon)^D$
- Observation:
- When the dimension is high, large fraction of volume is "on the fringe."
- 當維度很高的時候，大家都是邊緣人



Two Approaches to Reduce Dimension

- We can reduce dimension through (1) feature selection (2) dimensionality reduction methods.
- Feature selection: [Pro] Preserve the “raw” feature; [Con] Cannot “generate” good features
- Dimensionality Reduction: [Pro] May explore the local structure better. [Con] May destroy the intuition or the meaning of features.
- We are going to focus on the second approach here.

Why Dimensionality Reduction?

- Visualization: Projection of high-dimensional data onto 2D or 3D.
- Data compression: Efficient storage and retrieval.
- Noise removal: Positive effect on query accuracy.

Applications

- Useful tools when faced with high-dimensional data

- Text mining
- Image retrieval
- Microarray data analysis
- Face recognition
- Handwritten digit recognition
- Intrusion detection

Zoos of Dimensionality Reduction

- Unsupervised vs. Supervised.
- Unsupervised:
  - Principal component analysis (PCA)
  - Singular value decomposition (SVD)
  - Kernel PCA
  - Locally Linear Embedding (LLE)
  - t-SNE
- Supervised
  - Linear discriminant analysis (LDA) [Note: not Latent Dirichlet Allocation]

Zoos of Dimensionality Reduction

- Linear vs Nonlinear

- Linear:
  - SVD
  - PCA
  - LDA
- Nonlinear
  - Kernel PCA
  - LLE
  - t-SNE

- Will discuss PCA in detail, and present several selected methods as well.

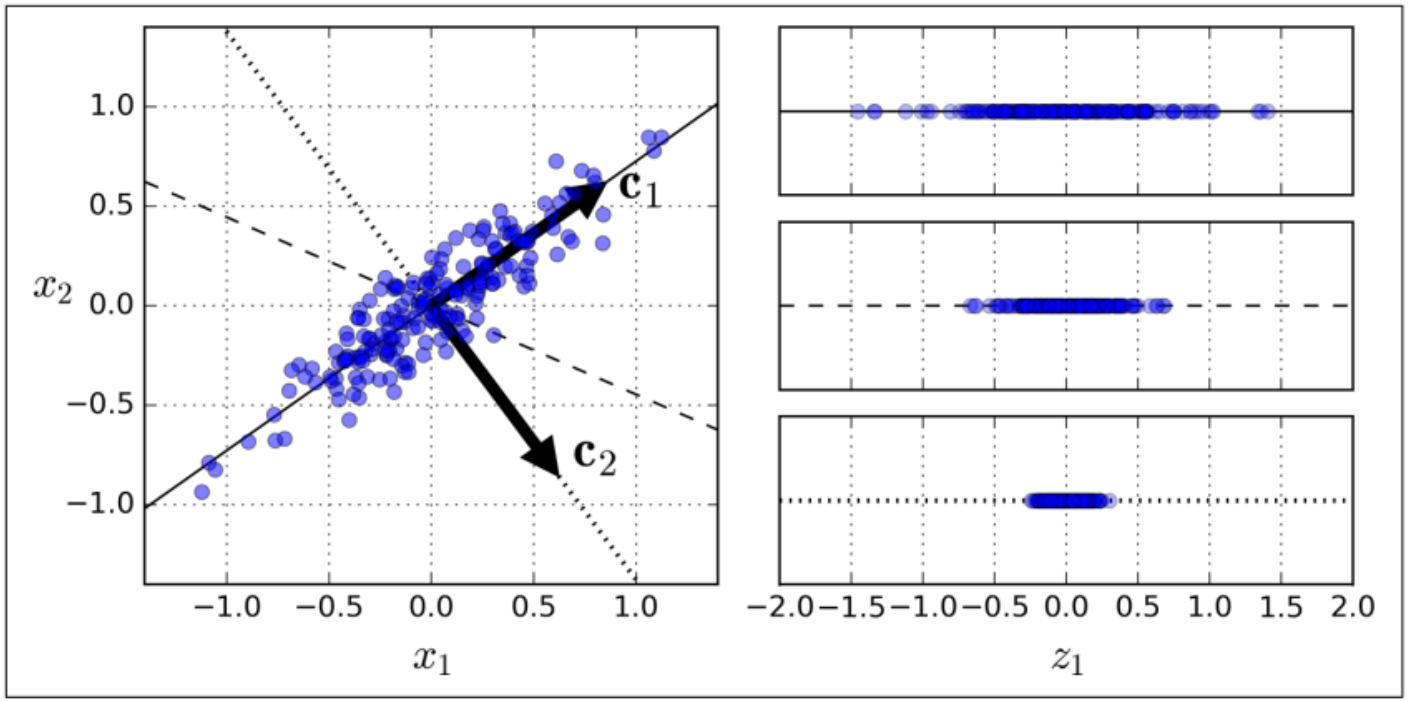
Principal Component Analysis (PCA)

- The central idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the variation present in the data set.
- This is achieved by transforming to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables.

- [Jolliffe, Principal Component Analysis, 2nd edition]
- Slides borrowed from Dr. Frank Wood.

Projecting to Lower Dimension

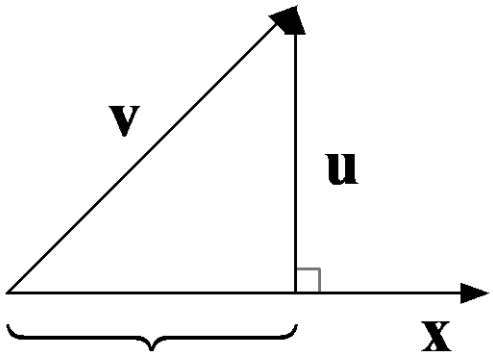
- How to choose the direction?
- (1) maximal variance after projection
- (2) minimal the mean squared difference between the projected points and original points



Recall: Projections

- If vectors **v** and **x** are linearly independent, then **v** can be expressed as the sum of a vector parallel to **x**, the vector **w**, and a vector perpendicular to **x**, the vector **u**. The vector parallel to **x** is the projection of **v** onto **x**.

$$\mathbf{v} = \mathbf{w} + \mathbf{u}$$



$$\mathbf{w} = \text{proj}_{\mathbf{x}}\mathbf{v}$$

$$= \left(\mathbf{v}, \frac{\mathbf{x}}{\|\mathbf{x}\|}\right)\frac{\mathbf{x}}{\|\mathbf{x}\|}$$

$$= \frac{(\mathbf{v},\mathbf{x})}{(\mathbf{x},\mathbf{x})}\mathbf{x}$$

PCA in a Nutshell

- Notation
- $\boldsymbol{x}$ is a vector of p random variables
- $\boldsymbol{\alpha}_k$ is a vector of p constants
- $\boldsymbol{\alpha}_k'\boldsymbol{x} = \sum_{j=1}^{p} \alpha_{kj} x_j$
- Procedural Description
- Find linear function of $\boldsymbol{x}$, $\boldsymbol{\alpha}_1'\boldsymbol{x}$ with maximum variance.
- Next find another linear function of $\boldsymbol{x}$, $\boldsymbol{\alpha}_2'\boldsymbol{x}$, uncorrelated with $\alpha_1'\boldsymbol{x}$, that has maximum variance.
- Iterate.
- Goal
- It is hoped, in general, that most of the variation in x will be accounted for by m PC's where $m \ll p$.

Derivation of PCA

- Assumption and More Notation
- Σ is the known covariance matrix for the random variable $\boldsymbol{x}$.
- Foreshadowing: Σ will be replaced with S, the sample covariance matrix, when Σ is unknown.
- Shortcut to solution
- For $k = 1, 2, \ldots, p$ the k-th PC is given by $z_k = \boldsymbol{\alpha}_k' \boldsymbol{x}$, where $\boldsymbol{\alpha}_k$ is an eigenvector of Σ corresponding to its k-th largest eigenvalue λ_k.
- If $\boldsymbol{\alpha}_k$ is chosen to have unit length (i.e., $\boldsymbol{\alpha}_k' \boldsymbol{\alpha}_k = 1$), then $Var(\boldsymbol{z}_k) = \lambda_k$

Derivation of PCA (Cont'd.)

- First Step
- Find $\boldsymbol{\alpha}_k' x$ that maximizes

$$Var(\boldsymbol{\alpha}_k' \boldsymbol{x}) = \boldsymbol{\alpha}_k' Var(\boldsymbol{x}) \boldsymbol{\alpha}_k = \boldsymbol{\alpha}_k' \Sigma \boldsymbol{\alpha}_k.$$

- Without constraint we could pick a very big $\boldsymbol{\alpha}_k$.
- Choose normalization constraint, namely $\boldsymbol{\alpha}_k' \boldsymbol{\alpha}_k = 1$ (unit length).
- Constrained maximization: Lagrange multipliers
- To maximize $\boldsymbol{\alpha}_k' \Sigma \boldsymbol{\alpha}_k$ subject to $\boldsymbol{\alpha}_k' \boldsymbol{\alpha}_k = 1$ we use the technique of Lagrange multiplier. We maximize the function

$$\boldsymbol{\alpha}_k' \Sigma \boldsymbol{\alpha}_k - \lambda(\boldsymbol{\alpha}_k' \boldsymbol{\alpha}_k - 1)$$

- w.r.t. $\boldsymbol{\alpha}_k$

Lagrange Multipliers

- Constrained maximization: $\boldsymbol{\alpha}_k' \Sigma \boldsymbol{\alpha}_k - \lambda(\boldsymbol{\alpha}_k' \boldsymbol{\alpha}_k - 1)$
- Solve by:
- $\frac{\partial\{\boldsymbol{\alpha}_k' \Sigma \boldsymbol{\alpha}_k - \lambda(\boldsymbol{\alpha}_k' \boldsymbol{\alpha}_k - 1)\}}{\partial \boldsymbol{\alpha}_k} = 2\Sigma\boldsymbol{\alpha}_k - 2\lambda_k \boldsymbol{\alpha}_k = 0$
- ➔ $\Sigma\boldsymbol{\alpha}_k = \lambda_k \boldsymbol{\alpha}_k$ (Wala!)

- $\boldsymbol{\alpha}_k$ is an eigenvector of Σ, with corresponding eigenvalue λ_k.
- Which eigenvector should we choose?

Lagrange Multipliers (Cont'd.)

- If we recognize that the quantity to be maximized
$$\boldsymbol{\alpha}_k'\Sigma\boldsymbol{\alpha}_k = \boldsymbol{\alpha}_k'\lambda_k\boldsymbol{\alpha}_k = \lambda_k\boldsymbol{\alpha}_k'\boldsymbol{\alpha}_k = \lambda_k$$
- Then we should choose λ_k to be as big as possible.
- Let λ_1 be the largest eigenvalue of Σ and $\boldsymbol{\alpha}_1$ the corresponding eigenvector, then the solution to
$$\Sigma\boldsymbol{\alpha}_1 = \lambda_1\boldsymbol{\alpha}_1$$
- Is the first principal component projector of $\boldsymbol{x}$.
- **Using realized data $\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}$, the first principle component is $[\boldsymbol{x_1'\alpha_1}\ \boldsymbol{x_2'\alpha_1}\ \ldots \boldsymbol{x_n'\alpha_1}]^{\boldsymbol{T}}$.**
- In general, $\boldsymbol{\alpha}_k$ will be the k-th PC projector of $\boldsymbol{x}$ and $Var(\boldsymbol{\alpha}_k'\boldsymbol{x}) = \lambda_k$.

The Second Component

- The second PC, $\boldsymbol{\alpha}_2'\boldsymbol{x}$, maximize $\boldsymbol{\alpha}_2'\Sigma\boldsymbol{\alpha}_2$ subject to being uncorrelated with $\boldsymbol{\alpha}_1'\boldsymbol{x}$.
- The uncorrelation constraint can be expressed using any of these equations.
- $cov(\boldsymbol{\alpha}_1'\boldsymbol{x}, \boldsymbol{\alpha}_2'\boldsymbol{x}) = \boldsymbol{\alpha}_1'\Sigma\boldsymbol{\alpha}_2 = \boldsymbol{\alpha}_2'\Sigma\boldsymbol{\alpha}_1 = \boldsymbol{\alpha}_2'\lambda_1\boldsymbol{\alpha}_1 = \lambda_1\boldsymbol{\alpha}_2'\boldsymbol{\alpha}_1 = 0$

- Thus we solve for the second PC by
- $\max_{\alpha_2} \boldsymbol{\alpha}_2'\Sigma\boldsymbol{\alpha}_2 - \lambda_2(\boldsymbol{\alpha}_2'\boldsymbol{\alpha}_2 - 1) - \phi\boldsymbol{\alpha}_2'\boldsymbol{\alpha}_1$

The Second Component (Cont'd.)

- $\max_{\alpha_2} \boldsymbol{\alpha}_2' \Sigma \boldsymbol{\alpha}_2 - \lambda_2(\boldsymbol{\alpha}_2' \boldsymbol{\alpha}_2 - 1) - \phi \boldsymbol{\alpha}_2' \boldsymbol{\alpha}_1$
- Solve by:
- $\frac{\partial\{\boldsymbol{\alpha}_2' \Sigma \boldsymbol{\alpha}_2 - \lambda_2(\boldsymbol{\alpha}_2' \boldsymbol{\alpha}_2 - 1) - \phi \boldsymbol{\alpha}_2' \boldsymbol{\alpha}_1\}}{\partial \alpha_2} = 2\Sigma \boldsymbol{\alpha}_2 - 2\lambda_2 \boldsymbol{\alpha}_2 - \phi \boldsymbol{\alpha}_1 = 0$
- If we left multiple $\boldsymbol{\alpha}_1$ to the above equation,
- $2\boldsymbol{\alpha}_1' \Sigma \boldsymbol{\alpha}_2 - 2\lambda_2 \boldsymbol{\alpha}_1' \boldsymbol{\alpha}_2 - \phi \boldsymbol{\alpha}_1' \boldsymbol{\alpha}_1 = 0 = 0 - 0 - \phi 1$
- Thus $\phi = 0$, and we are left with
- $\Sigma \boldsymbol{\alpha}_2 - \lambda_2 \boldsymbol{\alpha}_2 = 0$

The Second Component (Cont'd.)

- Clearly, $\Sigma\boldsymbol{\alpha}_2 - \lambda_2\boldsymbol{\alpha}_2 = 0$ is another eigenvalue equation.
- The second PC of $\boldsymbol{x}$ is $\boldsymbol{\alpha}_2'\boldsymbol{x}$.

- The process can be repeated for $k = 1, 2, \ldots, p$, yielding up to p different eigenvectors of Σ along with the corresponding eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_p$.

- Furthermore, the variance of each PC's are given by:

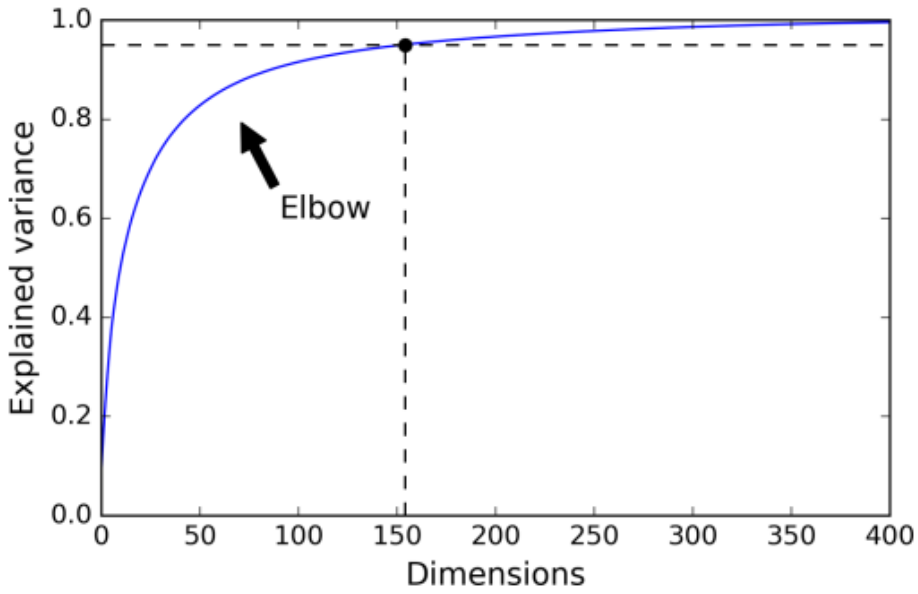
$$Var[\boldsymbol{\alpha}_k'\boldsymbol{x}] = \lambda_k, \qquad k = 1, 2, \ldots, p$$

Properties of PCA (Variance Explained)

- Our discussion on PCA suggest that we can decompose $\boldsymbol{x}$ by $\boldsymbol{x} = \boldsymbol{\alpha_1}\boldsymbol{\alpha}_1'\boldsymbol{x} + \boldsymbol{\alpha_2}\boldsymbol{\alpha}_2'\boldsymbol{x} + \cdots + \boldsymbol{\alpha_p}\boldsymbol{\alpha}_p'\boldsymbol{x}$.
- Moreover, because all principle components are uncorrelated,
- $Var(\boldsymbol{x}) = Var(\boldsymbol{\alpha}_1\boldsymbol{\alpha}_1'\boldsymbol{x}) + Var(\boldsymbol{\alpha}_2\boldsymbol{\alpha}_2'\boldsymbol{x}) + \cdots + Var\left(\boldsymbol{\alpha}_p\boldsymbol{\alpha}_p'\boldsymbol{x}\right)$
- We are interested in the "total variance of x," which is the sum of the diagonal elements in $Var(\boldsymbol{x})$, i.e., $tr\left(Var(\boldsymbol{x})\right)$.
- Note that $Var(\boldsymbol{\alpha}_1\boldsymbol{\alpha}_1'\boldsymbol{x}) = \alpha_1\alpha_1'\Sigma\alpha_1\alpha_1' = \alpha_1\lambda_1\alpha_1' = \lambda_1\alpha_1\alpha_1'$.
- Thus $tr\left(Var(\boldsymbol{\alpha}_1\boldsymbol{\alpha}_1'\boldsymbol{x})\right) = tr(\lambda_1\alpha_1\alpha_1') = \lambda_1 tr(\alpha_1'\alpha_1) = \lambda_1$.
- This derivation leads to $tr\left(Var(\boldsymbol{x})\right) = \lambda_1 + \lambda_2 + \cdots + \lambda_p$
- 總變異 = 特徵值之和

Variance Explained

- If we take the first q elements to approximate x, that is
- $\boldsymbol{x} \approx \tilde{\boldsymbol{x}} = \boldsymbol{\alpha}_1 \boldsymbol{\alpha}_1' \boldsymbol{x} + \boldsymbol{\alpha}_2 \boldsymbol{\alpha}_2' \boldsymbol{x} + \cdots + \boldsymbol{\alpha}_q \boldsymbol{\alpha}_q' \boldsymbol{x}$, then
- $tr\big(Var(\tilde{\boldsymbol{x}})\big) = \boldsymbol{\lambda}_1 + \boldsymbol{\lambda}_2 + \cdots + \boldsymbol{\lambda}_q$.
- That is, we can capture $\frac{\sum_{i=1}^{q} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \times 100$ percent of variation by using $\tilde{\boldsymbol{x}}$ instead of $\boldsymbol{x}$.



Plugging in Sample Covariance Matrix

- Our discussion are using sample covariance matrix Σ.
- When applying PCA, need to replace Σ with the unbiased estimator S.
- Let $\bar{\boldsymbol{x}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$,
- $S = \frac{1}{n-1}\sum_{i=1}^{n}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(\boldsymbol{x}_i - \bar{\boldsymbol{x}})^T$

PCA for Compression

- Apply PCA on MNIST dataset.
- MNIST is a set of 70,000 small images of digits handwritten by high school students and employees of the US Census Bureau.
- Each image is labeled with the digit it represent.
- Resolution: 28x28 = 784 features.

- Apply PCA and keep 90% of variance
  → 150 dimensions

PCA for Compression

```
pca = PCA(n_components = 154)
X_reduced = pca.fit_transform(X_train)
X_recovered = pca.inverse_transform(X_reduced)
```

Original

Compressed

Alternative Interpretation of PCA

- The way we construct PCA is to maximize explained variance.
- Alternatively, we can motivate the formulation via minimizing reconstruction error.
- To get start, assume that x can be decomposed into a linear combination of orthogonal vectors.
- $x = z_1 u_1 + z_2 u_2 + \cdots + z_p u_p$.
- We want to approximately reconstruct x via $\tilde{x}$ by taking the first q elements
- $\tilde{x} = \bar{x} + z_1 u_1 + z_2 u_2 + \cdots + z_q u_q$
- Goal: find $u_1, u_2, \ldots, u_q$ that minimize reconstruction error
- $E_m = E[(x - \tilde{x})^T (x - \tilde{x})]$

Minimizing Reconstruction Error

- $E_m = E[(x - \tilde{x})^T (x - \tilde{x})]$
- $= \sum_{i=(q+1)}^{p} E\left[u_i^T (x - \bar{x})(x - \bar{x})^T u_i\right]$
- $= \sum_{i=q+1}^{p} u_i^T \Sigma u_i.$
- That is, we try to minimize the variance not included.
- This is equivalent to maximize the projected variance as formulated in the original setting.

Singular Vector Decomposition (SVD)

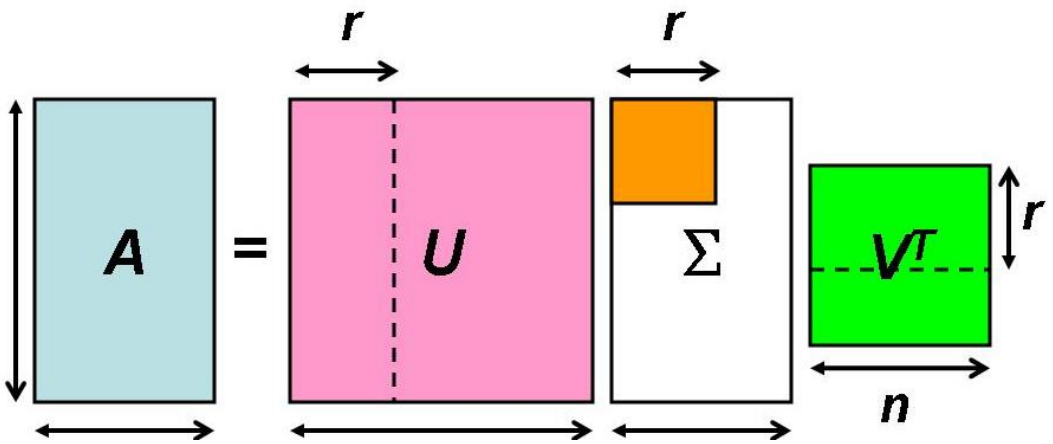
- SVD is closely related to PCA.
- In fact, you can compute PCA from SVD, or the other way around.
- The formulation, nonetheless, is somewhat different.
- SVD is usually introduced in Management Math (Linear Algebra).
- Will provide less details here.
- SVD is less “statistically driven” and is more “linear algebra.”

SVD

- Given a data matrix X $(n \times p)$, SVD decompose X into three matrices:
- $X = U\Sigma V^T$
- Both U and V are orthogonal matrix, which means

$$UU^T = I, U^{-1} = U^T$$
$$VV^T = I, V^{-1} = V^T$$

- Σ has nonzero elements on its diagonal, and zero elsewhere. Σ is usually not a squared matrix (unless $n = p$).



SVD (Cont'd.)

- $X = U\Sigma V^T = [u_1\ u_2\ \ldots u_p] \begin{bmatrix} \sigma_1 & & \\ & \sigma_2 & \\ & & \ddots \end{bmatrix} \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_n^T \end{bmatrix}$
- $\approx \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$
- (Take the first r singular values only).
- Meaning: the observed data matrix X is roughly the summation of r squared matrix, each is an outer product of two vectors.

$$\mathbf{u} \otimes \mathbf{v} = \mathbf{u}\mathbf{v}^\top = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix} = \begin{bmatrix} u_1 v_1 & u_1 v_2 & u_1 v_3 \\ u_2 v_1 & u_2 v_2 & u_2 v_3 \\ u_3 v_1 & u_3 v_2 & u_3 v_3 \\ u_4 v_1 & u_4 v_2 & u_4 v_3 \end{bmatrix}.$$

Motivating SVD

- $X \approx \sigma_1 u_1 v_1^T + \sigma_2 u_2 v_2^T + \cdots + \sigma_r u_r v_r^T$
- Why this is useful or meaningful?
- This decomposition corresponds to the idea that the data matrix X is the result of two "factors."
- Consider the following example.
- X is a rating matrix of 4 users on 3 movies.
- Each user rates a move based on two characteristics:
  (1) whether there are many explosions
  (2) whether there are fun carton characters.
- A user may like of dislike the two characteristics.
- For each user, we use a vector of length two to represent their preference. Each element is between 0 and 1.
- 0 means the characteristics does not matter; 1 means it matters a lot.

Rating Data

- Example:
- Preference vector for Alex = [0.1, 0.9]
- Mary [0.9, 0.2]
- Bob [0.4, 0.5]
- Eva [0.9, 0.9]
- Each movie also has two measures for the two characteristics. For example, movieA = [10, 3] means that this movie has 10 explosions and 3 minutes of showing funny carton characters.
- movieB = [2, 10]: 2 explosions, 10 minutes of cartons.
- movieC = [6, 6]

Rating Data Computation

- User preference
- $U = [u_1\ u_2] = \begin{bmatrix} 0.1 & 0.9 \\ 0.9 & 0.2 \\ 0.4 & 0.5 \\ 0.9 & 0.9 \end{bmatrix}$

- Movie characteristics
- $V^T = \begin{bmatrix} v_1^T \\ v_2^T \end{bmatrix} = \begin{bmatrix} 10 & 2 & 6 \\ 3 & 10 & 6 \end{bmatrix}$

- Assuming the two characteristics are equally important ($\sigma_1 = \sigma_2 = 1$)
- Rating based on the first and second characteristics:

- $u_1 v_1^T = \begin{bmatrix} 1 & 0.2 & 0.6 \\ 9 & 1.8 & 5.4 \\ 4 & 0.8 & 2.4 \\ 9 & 1.8 & 5.4 \end{bmatrix}, \mathrm{u_2 v_2^T} = \begin{bmatrix} 2.7 & 9 & 5.4 \\ 0.6 & 2 & 1.2 \\ 1.5 & 5 & 3.0 \\ 2.7 & 9 & 5.4 \end{bmatrix}, X = \begin{bmatrix} 3.7 & 9.2 & 6.0 \\ 9.6 & 3.8 & 6.6 \\ 5.5 & 5.8 & 5.4 \\ 11.7 & 10.8 & 10.8 \end{bmatrix}$

Rating Data Interpretation

- To generate rating data, we sum over the outer product of user preference and movie characteristics.
- In this example, the rating data was generated by two "factors"
- SVD is trying to do the reverse, it try to "recover" factors given the final results.

- Another way to generate X_{ij} is to take the i-th row in U and the j-th column in V^T and do inner product.
- This can be interpreted as the "internal compatibility" of user i and movie j.

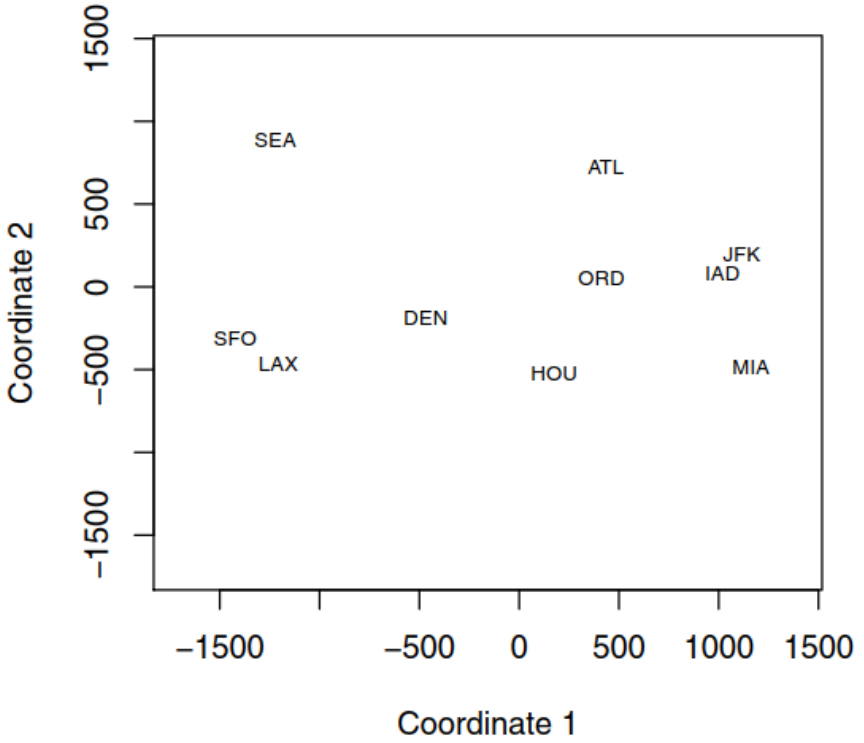
Real World Applications of SVD

- In real word application, only X is observed.
- The goal is often to do prediction via computing SVD from X, the user-item matrix.
- In recommender systems, the goal is to fill-in the “missing” ratings via the computed SVD from observed data.
- For visualization, the first few columns of U are good candidate for movie visualization;
  the first few columns of V are good candidate for user visualization.

Multidimensional Scaling (MDS)

- Goal of Multidimensional Scaling (MDS): Given pairwise dissimilarities, reconstruct a map that preserves distance.
- MDS comes with three flavors:
  - Classic: equivalent to PCA. Will not discuss here.
  - Metric: distance respect triangular inequality
  - Non-metric: Can handle different types of distance definitions

	ATL	ORD	DEN	HOU	LAX	MIA	JFK	SFO	SEA	IAD
ATL	0									
ORD	587	0								
DEN	1212	920	0							
HOU	701	940	879	0						
LAX	1936	1745	831	1374	0					
MIA	604	1188	1726	968	2339	0				
JFK	748	713	1631	1420	2451	1092	0			
SFO	2139	1858	949	1645	347	2594	2571	0		
SEA	218	1737	1021	1891	959	2734	2408	678	0	
IAD	543	597	1494	1220	2300	923	205	2442	2329	0



Perception of Color in Human Vision

- To study the perceptions of color in human vision (Ekman, 1954, Lzenman 13.2.1)
- 14 colors differing only in their hu (i.e., wavelengths from 434 um to 674 um)
- 31 people to rate on a five-point scale from 9 (no similarity at all) to 4 (identical) for each of C_2^{14} pairs of colors.
- Average of 31 rating for each pair (representing similarity) is then scaled and subtracted from 1 to represent dissimilarities.

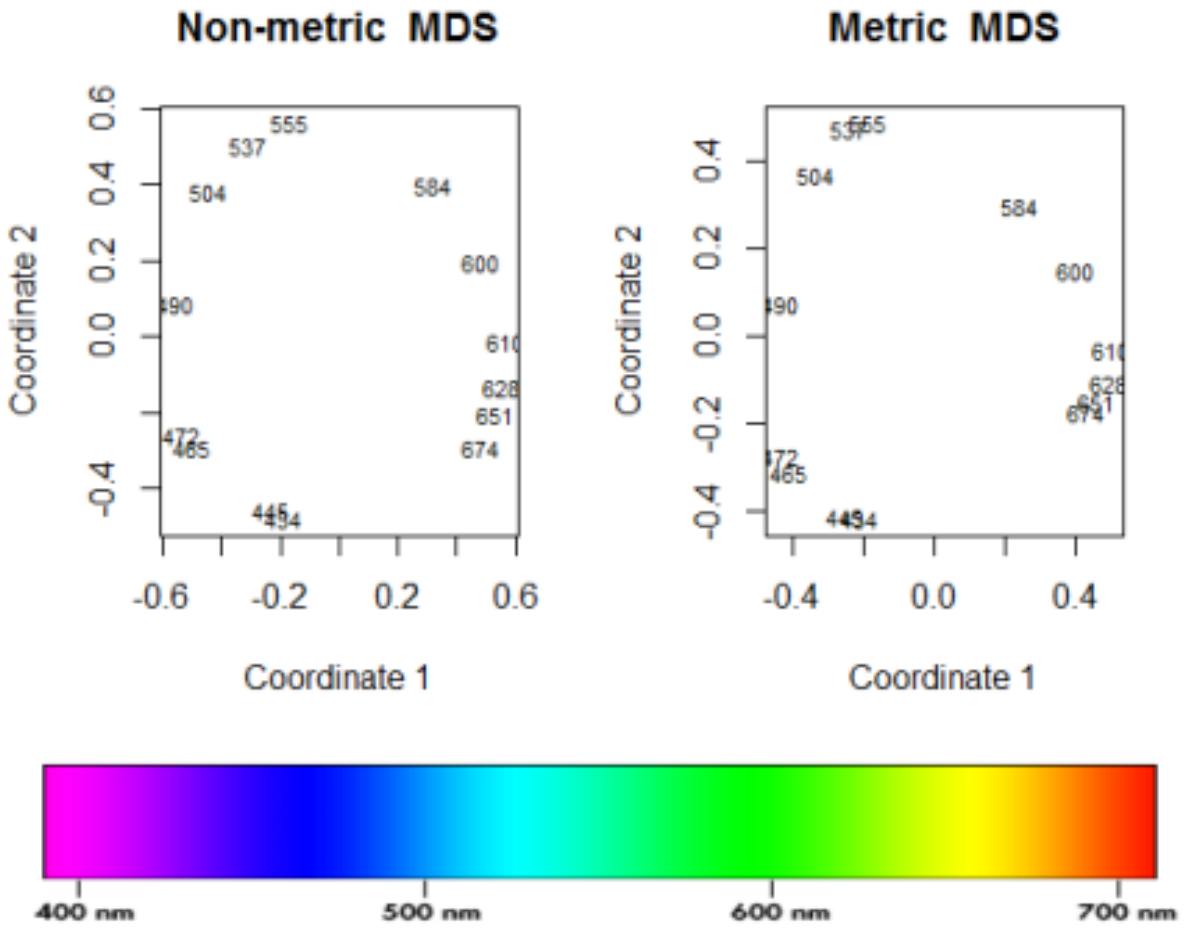
- [Adopted from Dr. Sungkyu Jun's slides.]

Color Dissimilarity Matrix

	434	445	465	472	490	504	537	555	584	600	610	628	651
445	0.14												
465	0.58	0.50											
472	0.58	0.56	0.19										
490	0.82	0.78	0.53	0.46									
504	0.94	0.91	0.83	0.75	0.39								
537	0.93	0.93	0.90	0.90	0.69	0.38							
555	0.96	0.93	0.92	0.91	0.74	0.55	0.27						
584	0.98	0.98	0.98	0.98	0.93	0.86	0.78	0.67					
600	0.93	0.96	0.99	0.99	0.98	0.92	0.86	0.81	0.42				
610	0.91	0.93	0.98	1.00	0.98	0.98	0.95	0.96	0.63	0.26			
628	0.88	0.89	0.99	0.99	0.99	0.98	0.98	0.97	0.73	0.50	0.24		
651	0.87	0.87	0.95	0.98	0.98	0.98	0.98	0.98	0.80	0.59	0.38	0.15	
674	0.84	0.86	0.97	0.96	1.00	0.99	1.00	0.98	0.77	0.72	0.45	0.32	0.24

MDS Mapping Results

MDS reproduces the well-known two-dimensional *color circle*.



Distance, Dissimilarity, and Similarity

- Distance or metric between a pair of objects need to satisfy:
- $d(x, y) \geq 0$
- $d(x, y) = 0 \; if \; and \; only \; if \; x = y$
- $d(x, y) = d(y, x)$
- $d(x, z) \leq d(x, y) + d(y, z)$
- Distance could be Euclidian or not.
- Dissimilarity and similarity can be adopted from distance.

Metric MDS

- Given the distances between n objects, a low dimension p (e.g., 2), and a monotone function f, metric MDS seeks to find an optimal configuration $x_1, x_2, \ldots, x_n \in R^p$ such that for each pair of object $f(d_{ij})$ is as close as $\hat{d}_{ij} = \|x_i - x_j\|_2$ as possible.
- The function f is usually set to $f(d_{ij}) = \alpha + \beta d_{ij}$.
- "As close as possible:" loss (i.e. stress)
- $L(x_1 \ldots x_n) = \sqrt{\frac{\sum_{i<j}\left(\hat{d}_{ij} - f(d_{ij})\right)^2}{\sum_{i<j} d_{ij}^2}}$
- Minimize $L(\cdot)$ w.r.t. x_i, α, β.
- Often set $\alpha = 0$ and $\beta = 1$ (the "usual" MDS).

Non-metric MDS

- When the dissimilarities are only meaningful in their rank order, non-metric MDS may be more suitable.
- Non-metric MDS:
- Given the dissimilarities between n objects, a low dimension p (e.g., 2), and a monotone function f, metric MDS seeks to find an optimal configuration $x_1, x_2, \dots, x_n \in R^p$ such that for each pair of object $f(d_{ij})$ is as close as $\hat{\mathrm{d}}_{ij} = \|x_i - x_j\|_2$ as possible.
- $f(d_{ij})$ only preserve the order of d_{ij}.
- $d_{ij} < d_{kl}$ ⟷ $f(d_{ij}) \leq f(d_{kl})$.
- $f(\cdot)$ is found by monotonic regression.

$$L(X) = \sqrt{\frac{\sum_{i<j}\left(\hat{d}_{ij} - f(d_{ij})\right)^2}{\sum_{i<j} d_{ij}^2}}$$

Kernel PCA (cf. PRML Ch 12.3)

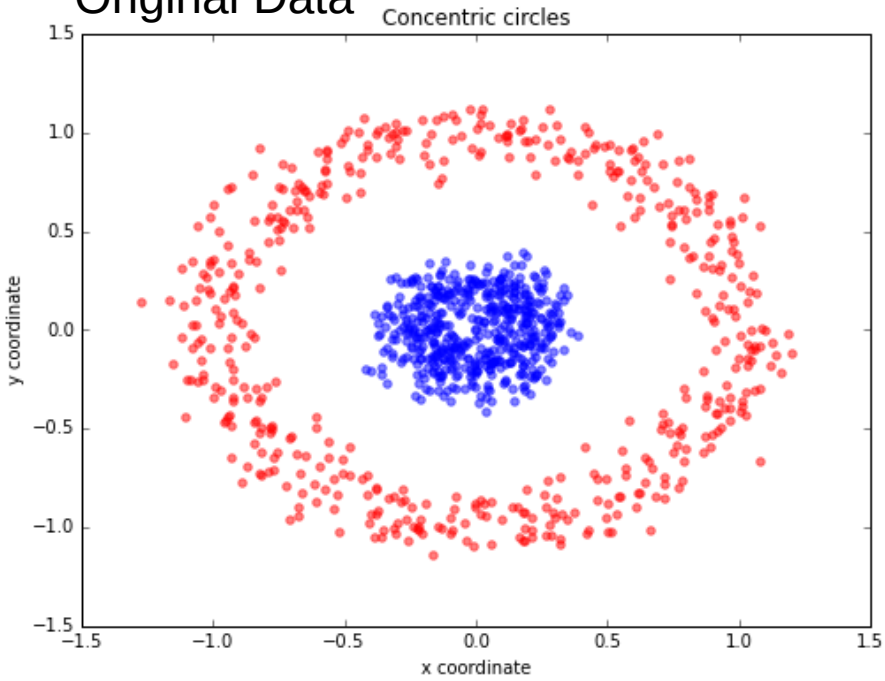
- MDS adopted a “distance” or “dissimilarity” metric to visualize data.
- A similar idea is to adopt the “kernel trick” on PCA.
- A kernel function gives the similarity between two objects. A commonly used kernel function is the Radio Basis Function (RBF)
- $k(x_a, x_b) = \exp\left\{\frac{-\|x_a - x_b\|_2^2}{t}\right\}$.
- Kernel PCA operate on the “similarity matrix” generated by the kernel function.
- It starts from the standard PCA and adopted the kernel trick to derive kernel PCA.
- Kill not discuss “kernel trick” here.

PCA and Kernel PCA

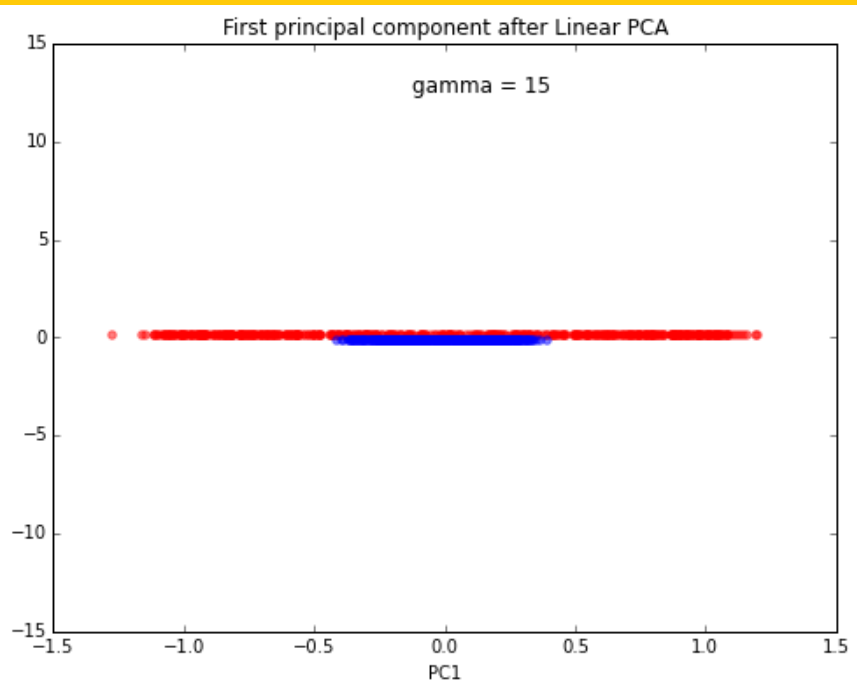
- Recall that given $x_1, x_2, \ldots, x_N$, PCA computes the covariance matrix S from data, and find eigenvectors by
- $Su_i = \lambda_i u_i,\ \|u_i\| = 1$
- The loading on the i-th PC (for data point j): $x_j' u_i$
- In Kernel PCA, we construct the gram matrix K by: $K_{ij} = k(x_i, x_j)$.
- Solve the following eigenvalue problem:
- $Ka_i = \lambda_i N a_i$.
- The loading on the i-th PC (for data point j): $\sum_{n=1}^{N} a_{in}\, k(x_j, x_n)$

Example

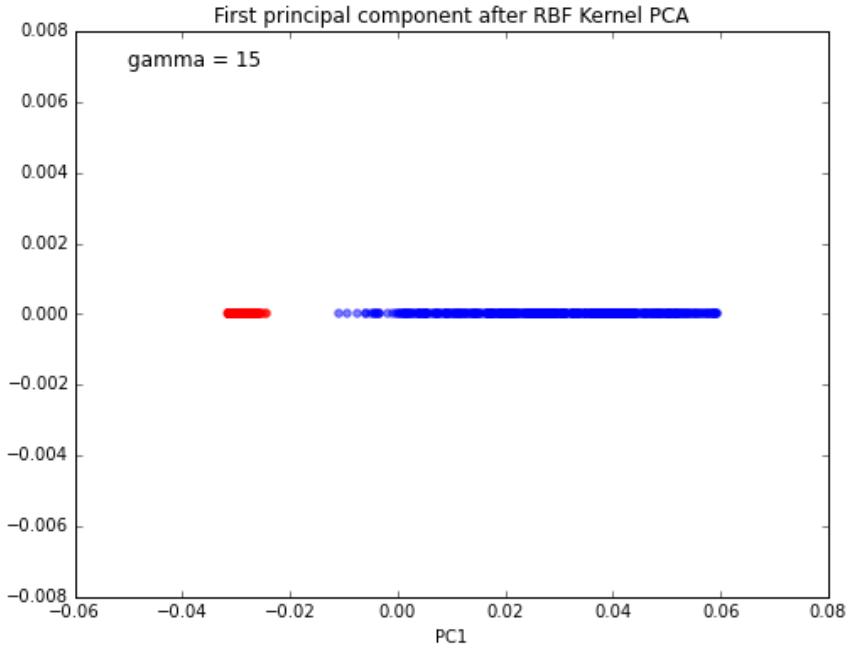
Original Data



PCA

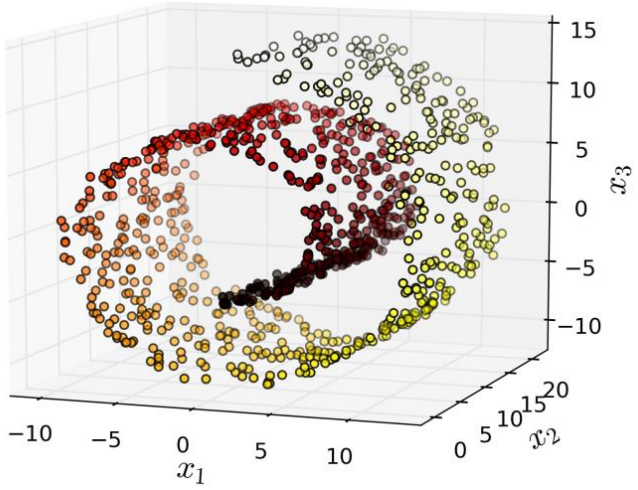


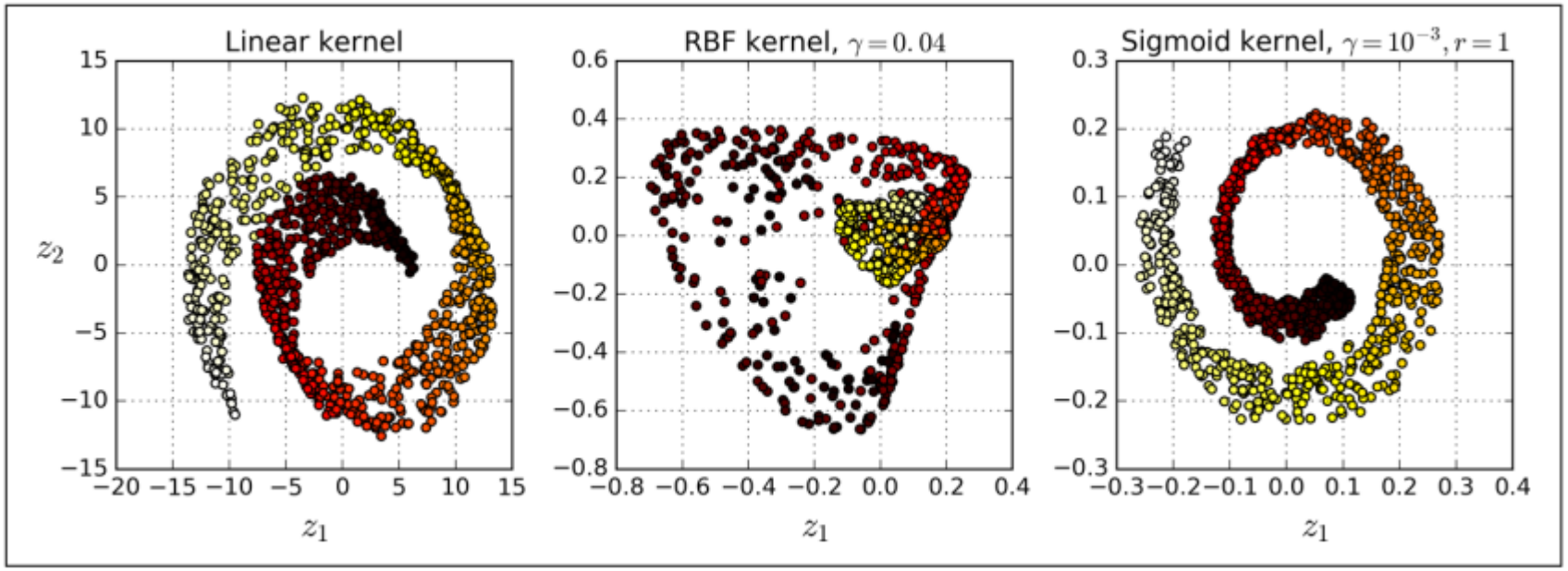
Kernel PCA



Swiss Roll Reduced to 2D

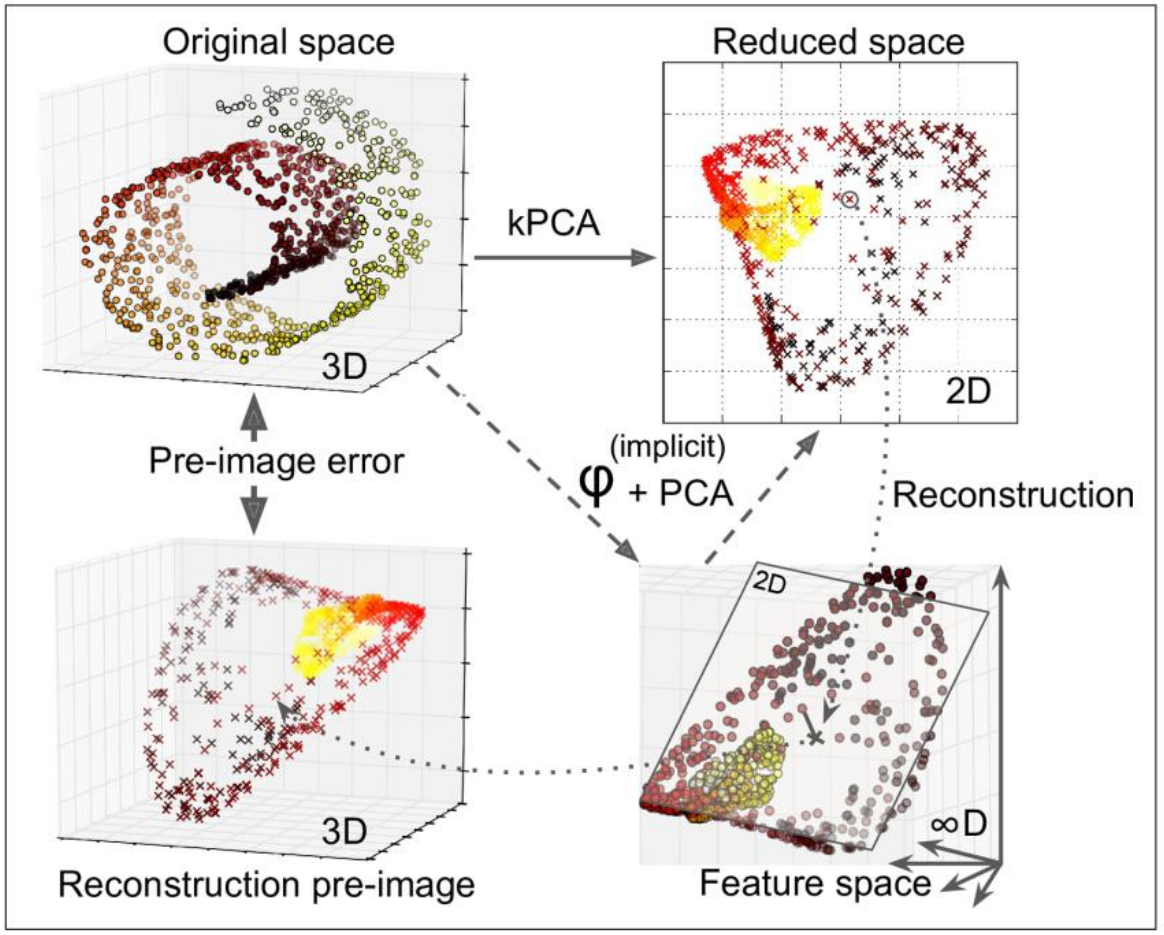
- Original dataset:





Parameter Tuning for Kernel PCA

- Two approach:
- (1) Based on the associated supervised learning task
- (2) Based on reconstruction error.



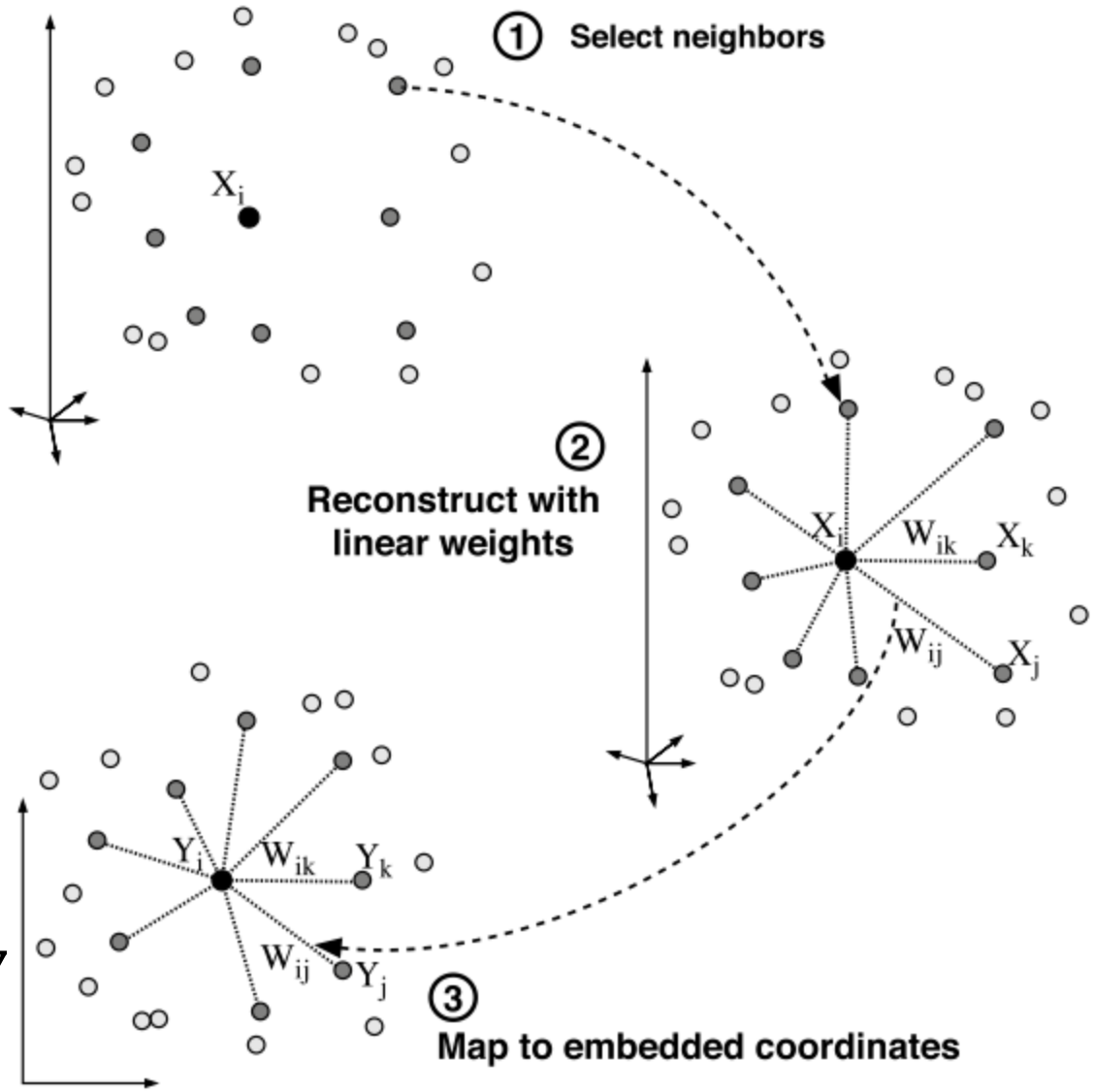
Locally Linear Embedding (LLE)

- LLE computes low-dimensional, neighborhood-preserving embedding of high-dimensional inputs.

- Given a set of data points, LLE computes the embedding in three steps:
- (1) Identify the neighborhood for each data point
- (2) Construct a weight matrix W.
- (3) Compute the final embedding coordinate using W.

LLE

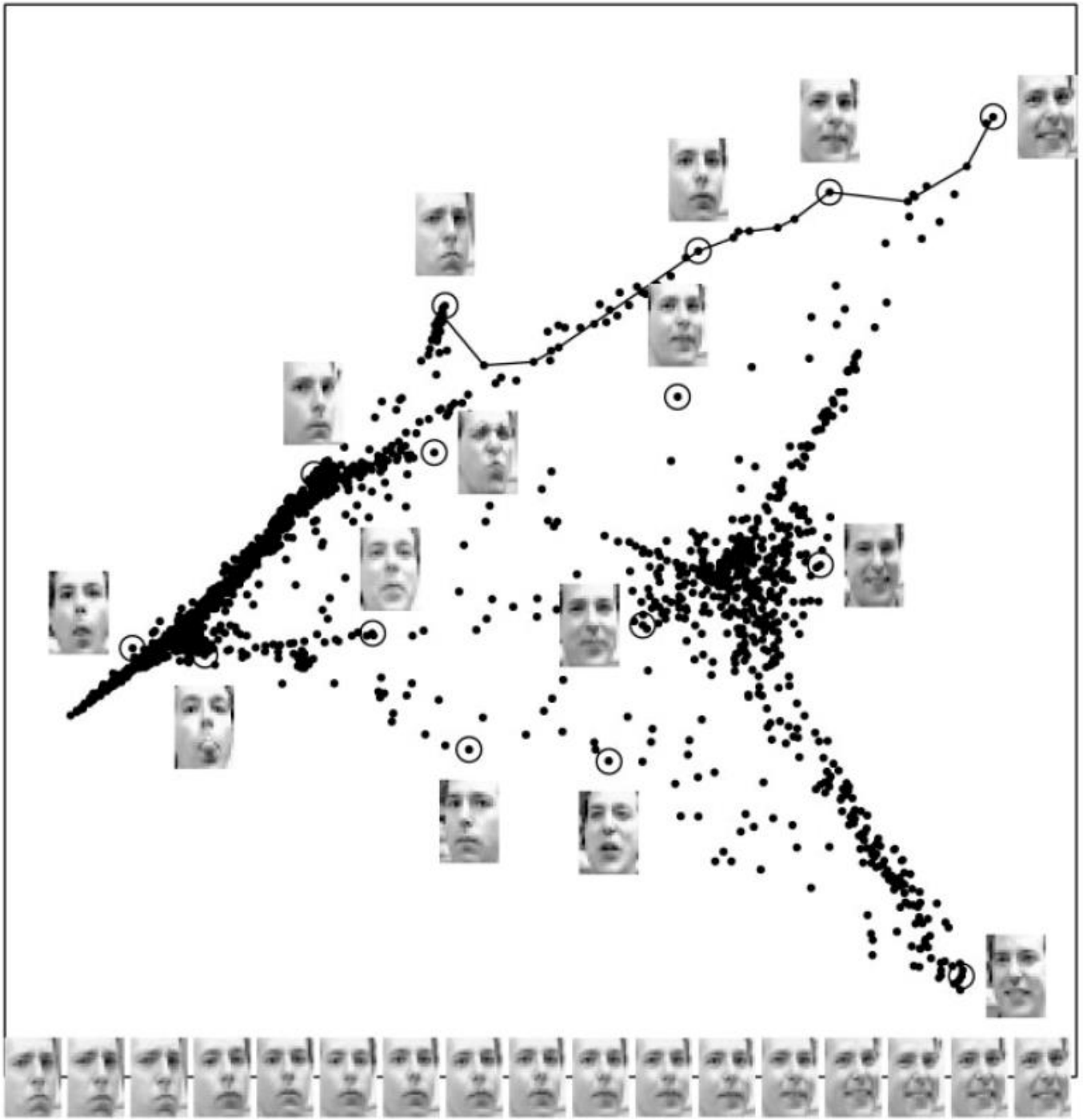
- (1) Assign neighbors to each data point using the KNN.
- (2) Compute the weight W_{ij} that best reconstruct x_i from its neighbors.
- (3) Compute the low dimensional embedding vector y_i best reconstructed W



LLE (Cont'd.)

- Step 0. Set the k for kNN. (e.g. 10).
- Step 1. Identify closet neighbors for each data points.
- Step 2. Solve for W_{ij} that minimize
- $L(W) = \sum_{i=1}^{n} \left(x_i - \sum_j W_{ij} x_j \right)^2$
- s.t. (1) $W_{ij} = 0$ if x_j is not one of the k nearest neighbors of x_i.
- (2) $\sum_j W_{ij} = 1$
- Step 3. Solve for y_i that minimize
- $L_y(Y) = \sum_i \left(y_i - \sum_j W_{ij} y_j \right)^2$
- ➔ First two principle components of W.

LLE Example



t-Distributed Stochastic Neighbor Embedding (t-SNE)

- t-SNE uses probability to represent similarity.
- If x_i and x_j are similar, you will see them appear together (as neighbors) more often.
- $$p_{ij} = \frac{\exp\left\{-\frac{\|x_i - x_j\|^2}{2\sigma^2}\right\}}{\sum_{k \neq l} \exp\left\{-\frac{\|x_k - x_l\|^2}{2\sigma^2}\right\}}$$
- We are trying to find low dimensional vector y_i that matches the probability distribution defined by p_{ij}.

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- In order to do so, we define another probability distribution q_{ij} by

- $$q_{ij} = \frac{\left(1+\|y_i - y_j\|^2\right)^{-1}}{\sum_{k \neq l}(1+\|y_k - y_l\|^2)^{-1}}$$

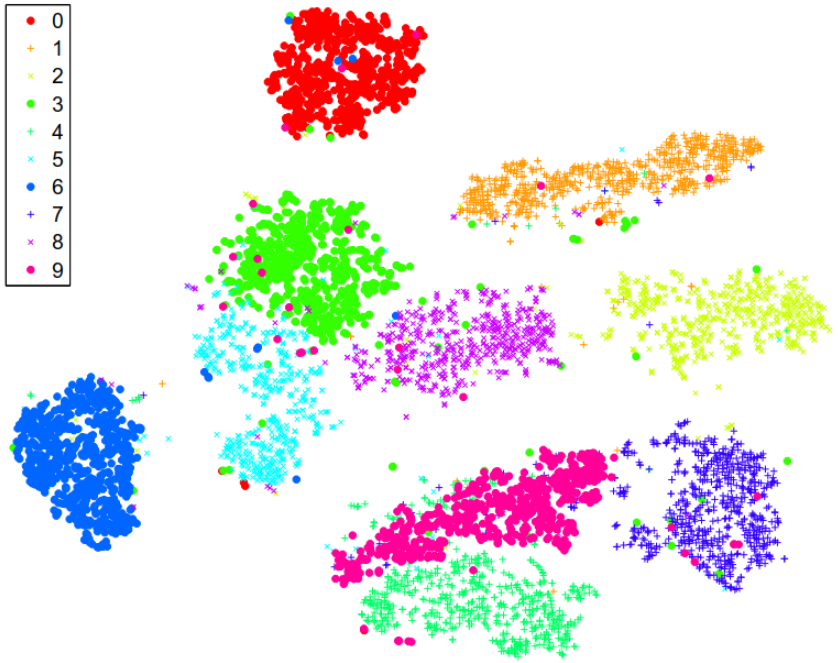
$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})}\left(1+\frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

- The function form is inspired by t-distribution of 1 degree freedom.
- Why? We want to allow more freedom in the embedded space so that the algorithm can “squeeze” high-dimensional data into the embedded space.
- t-distribution has heavier tail than the Gaussian distribution, thus is more tolerant to minor incompability.

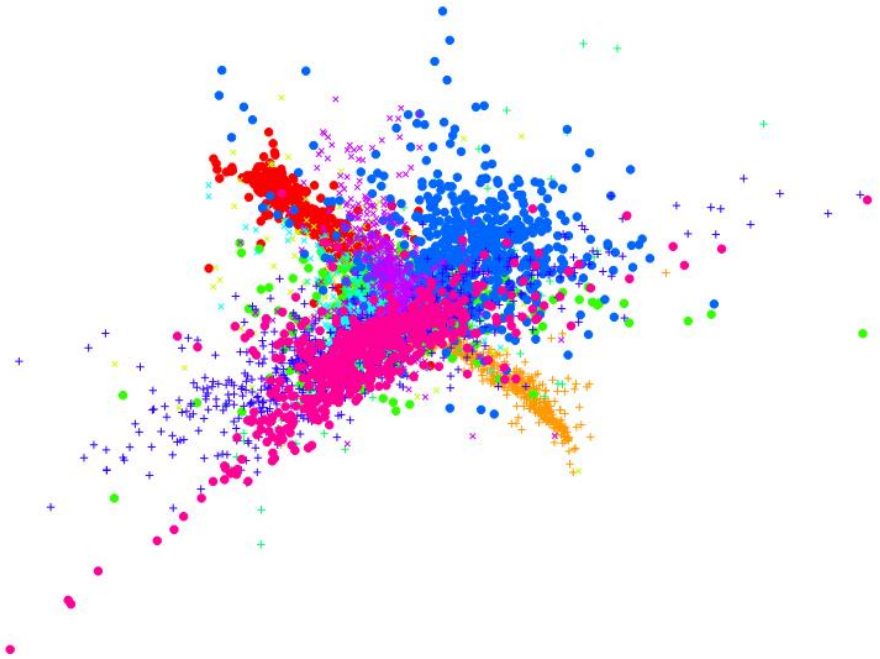
Model Training

- We matches q_{ij} with p_{ij} by minimizing the discrepancy between these two distributions.
- This is done by minimizing Kullback-Leibler (KL) divergence:
- $C = KL(P\|Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$
- Optimization achieved by gradient descent.
- Will not get into details here.

t-SNE Example on MNIST



(a) Visualization by t-SNE.



(b) Visualization by LLE.

Reading

- PRML Ch 1.4
- Geron Ch 8
- PRML Ch 12.1 and 12.3