Dimensionality Reduction and Representation Learning

Ariel Shamir

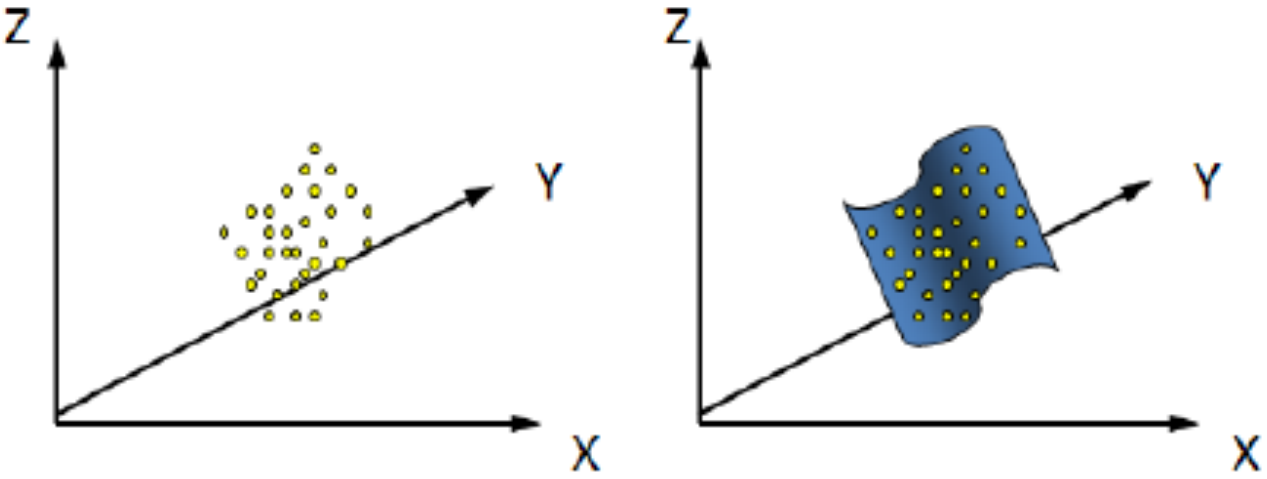
Zohar Yakhini

Reichman University

Overview

- Curse of Dimensionality
- Manifold of the data
- Representation learning

- Linear Mappings:
  - PCA
  - LDA
- Non-Linear Mappings:
  - MDS
  - tSNE

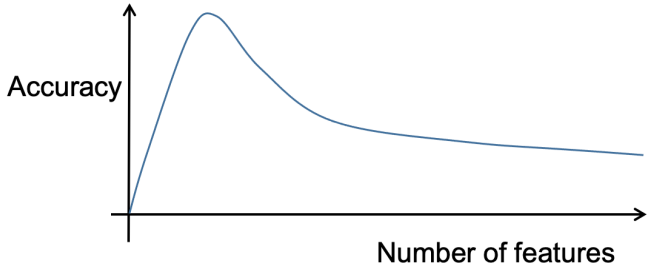


Rule of Thumb

- In more “complex boundary” algorithms that tend to over-fit (KNN, Decision trees) one should try to lower the problem dimension
- In algorithms that generalize more simply (linear classifiers, Naïve Bayes) one can use higher dimensions

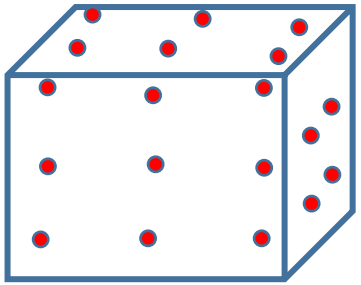
- Another view on this relates to the Kernel-Trick: using simple classifiers in high dimensions corresponds to complex classifiers in lower dimensional space

Curse of Dimensionality



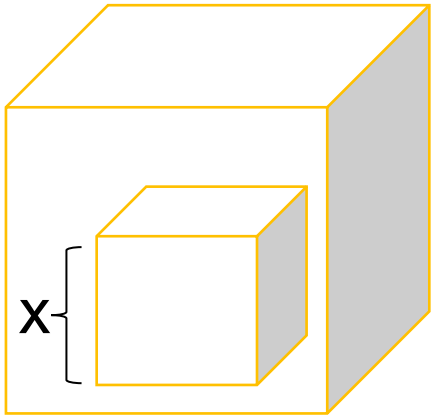
More Feature (higher dimentional space)…

- Can sometimes lead to **worse** performance
- Can introduce false regularities
- Means more memory and processing time
- Requires more training data – descent sampling grows exponentially
- Makes it difficult to interpret and understand the results (e.g., visualization)

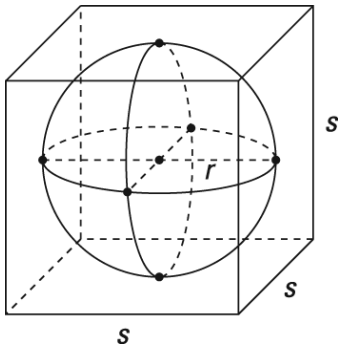


Everything is Far

- Assume we have a d dimensional unit cube and the samples are uniformly distributed in it.
- We want to define a neighborhood to capture a fraction r of the samples (for instance for NN or Parzen window)
- Since it is uniformly sampled this corresponds to a fraction r % of the unit cube volume
- The expected edge length x of such a fraction cube would be $x = r^{1/d}$
- For d=10 what is the edge length to capture 1% of the samples? $x = 0.01^{0.1} = 0.63$
- If we want 10% we get $x = 0.1^{0.1} = 0.8$, which means we must cover 80% of the range of each variable!



Volume of HyperSphere vs. HyperCube

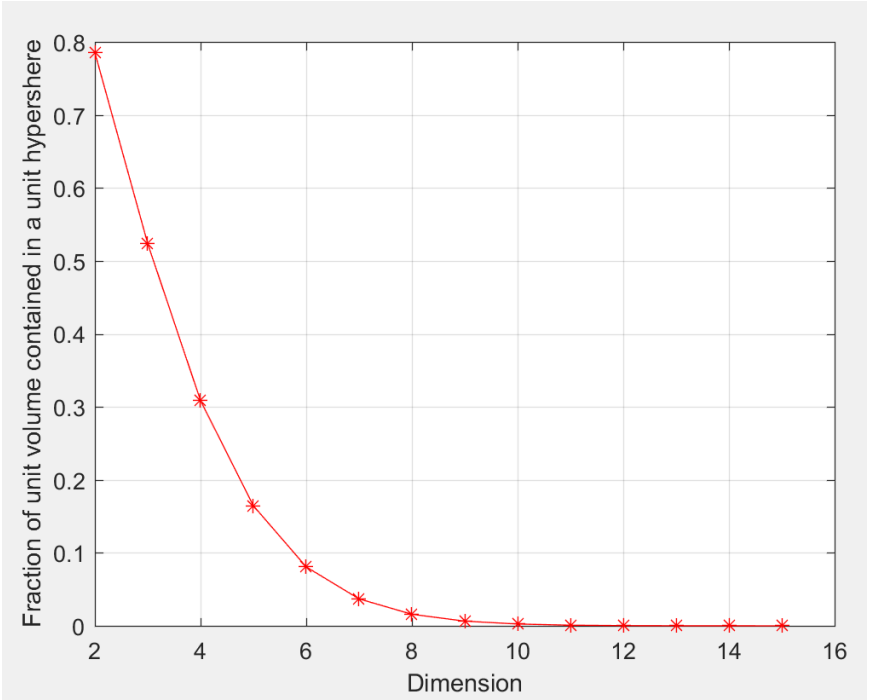


- The neighborhood of a point is defined by a hyper-sphere around the point
- This neighborhood is negligible in high dimensions!

$$\frac{V(S_2(r))}{V(H_2(2r))} = \frac{\pi r^2}{4r^2} = 78.5\%$$

$$\frac{V(S_3(r))}{V(H_3(2r))} = \frac{\frac{4}{3}\pi r^3}{8r^3} = 52.4\%$$

$$\lim_{d\to\infty} \frac{V(S_d(r))}{V(H_d(2r))} = \lim_{d\to\infty} \frac{\pi^{d/2}}{2^d\,\Gamma\left(\frac{d}{2}+1\right)} \to 0$$



Distance Function Sensitivity Decrease

- In high dimensions, volume of a hyper-sphere is insignificant inside a hyper cube - i.e. all points are “far away” from each other…
- A measure such as a Euclidean distance is defined using many coordinates, there is little difference in the distances between different pairs of samples
- It is more difficult to generalize! Every instance is very different from other instances

Mapping to Lower Dimensional Space

- **Feature selection** find a subset of features that are relevant to the problem at hand and work only with this subset.

$$\{x_1, x_2, \dots, x_n\} \rightarrow \{x_{i1}, x_{i2}, \dots, x_{im}\}$$

where $\{x_{i1}, x_{i2}, \dots, x_{im}\} \subset \{x_1, x_2, \dots, x_n\}$

- **Feature extraction** find new set of features (combinations of existing?) that better represent the data (usually in lower dimension).

$$\{x_1, x_2, \dots, x_n\} \rightarrow \{y_1, y_2, \dots, y_m\}$$

where $y_i = f(x_1, x_2, \dots, x_n)$

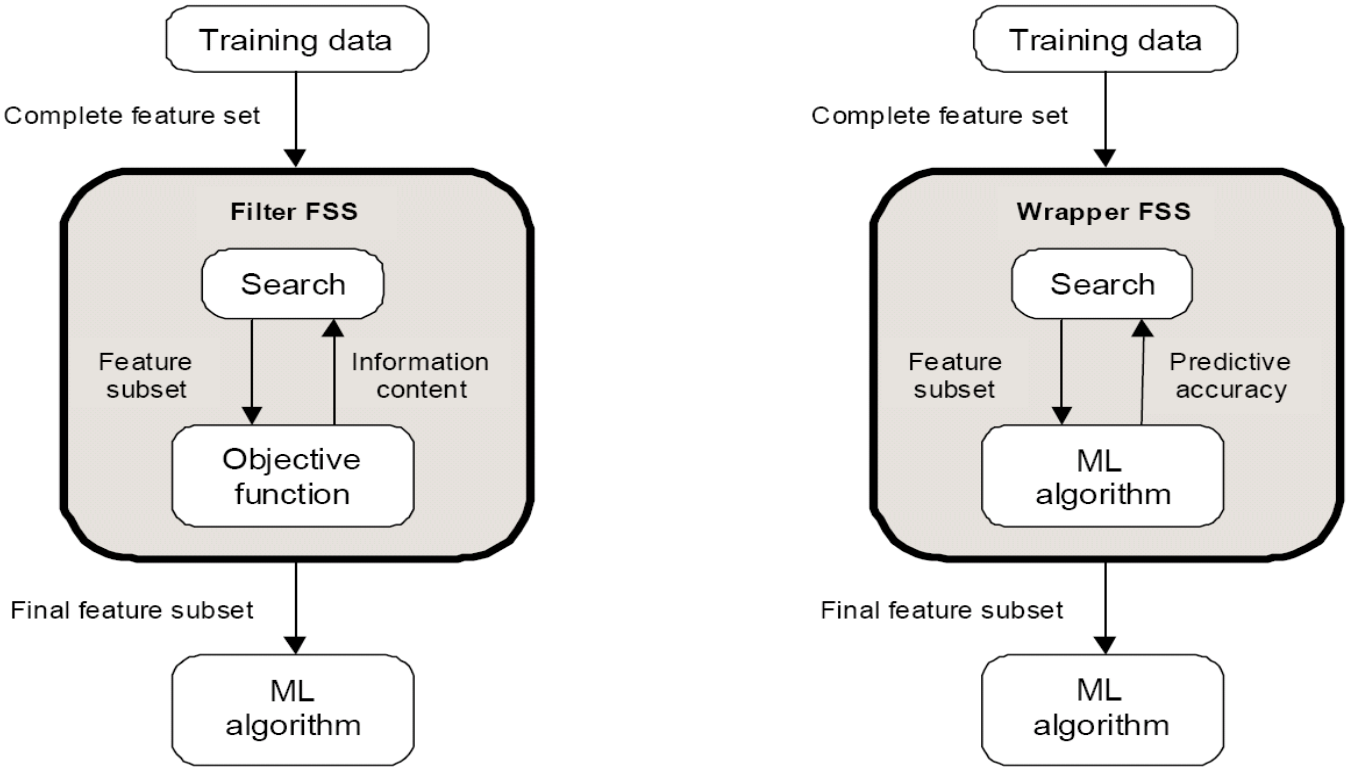
Different Approaches to Feature Selection

- Wrapper Methods
  - Select the best features to optimize your ML performance
- Filter Methods
  - Select the best features according to a reasonable criteria
  - Run your classifier on the subset

- How many feature subsets are possible?

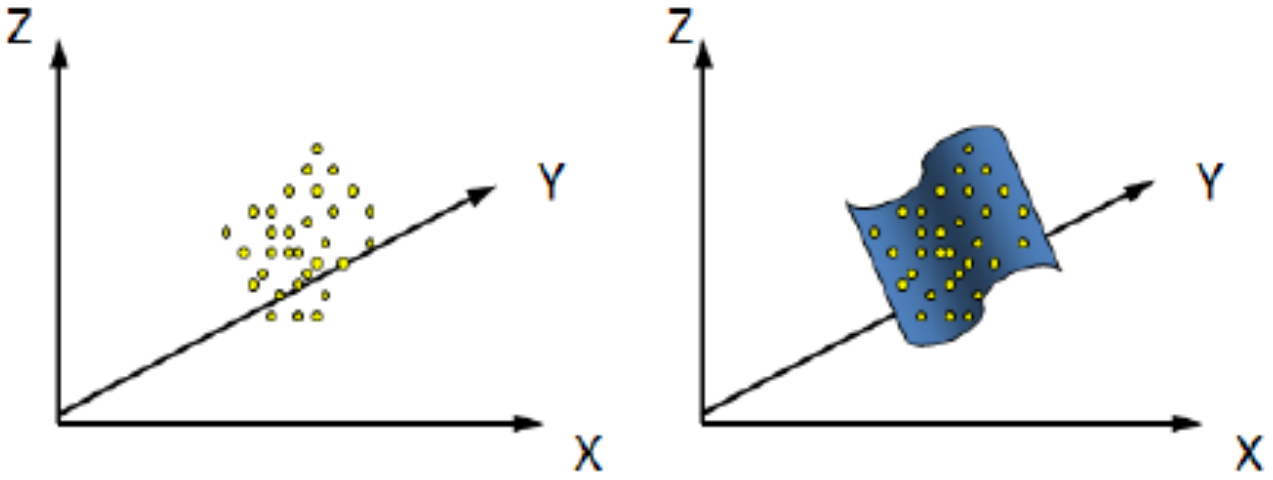
2^n Exponential!

Filters vs. Wrappers



Low Dimension Manifold of the Data

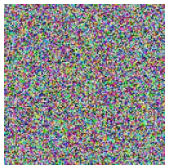
- It is often assumed that your data lies in a low dimensional sub-space (often called “manifold”) of your feature space



Low Dimension Manifold of the Data

- It is often assumed that your data lies in a low dimensional sub-space (often called “manifold”) of your feature space
- Example: all real photographs inside the space of possible matrices of randome colors

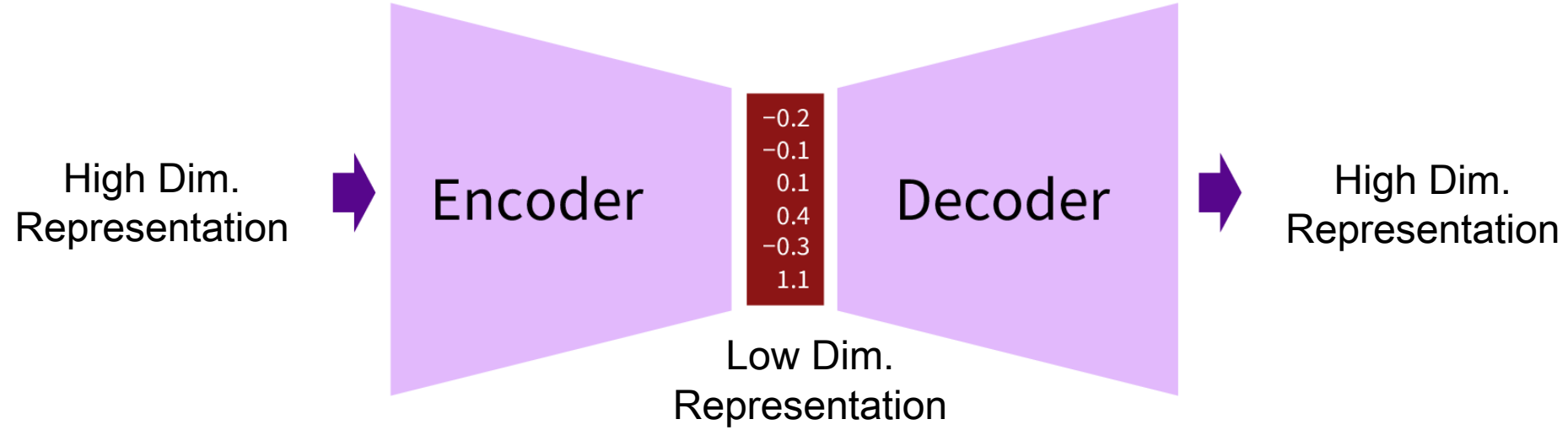
Vs.



What is the feature space size?

Representation Learning/ Manifold Learning

- A techniques that allows a system to automatically discover the representations needed for feature detection or classification from raw data = it finds the “manifold” of the data.
- Very popular unsupervised method uses “encoder/decoder” or “auto-encoder” – you encode your data into a low dimension space and use this feature space for later learning tasks



Principal Component Analysis

- Input: A set of m n-dimensional vectors/points

(m is the size of data, n is the number of features)

- Output: A better representation of the vectors
- Main idea:
  - Transform the vector space and represent the vectors in a new basis where axes better describe **the shape** of the given data (set of all vectors).
  - Minimize the dependency (linear correlation) between the different dimensions of the data

PCA Objective

- PCA finds an orthogonal transformation of the coordinate system

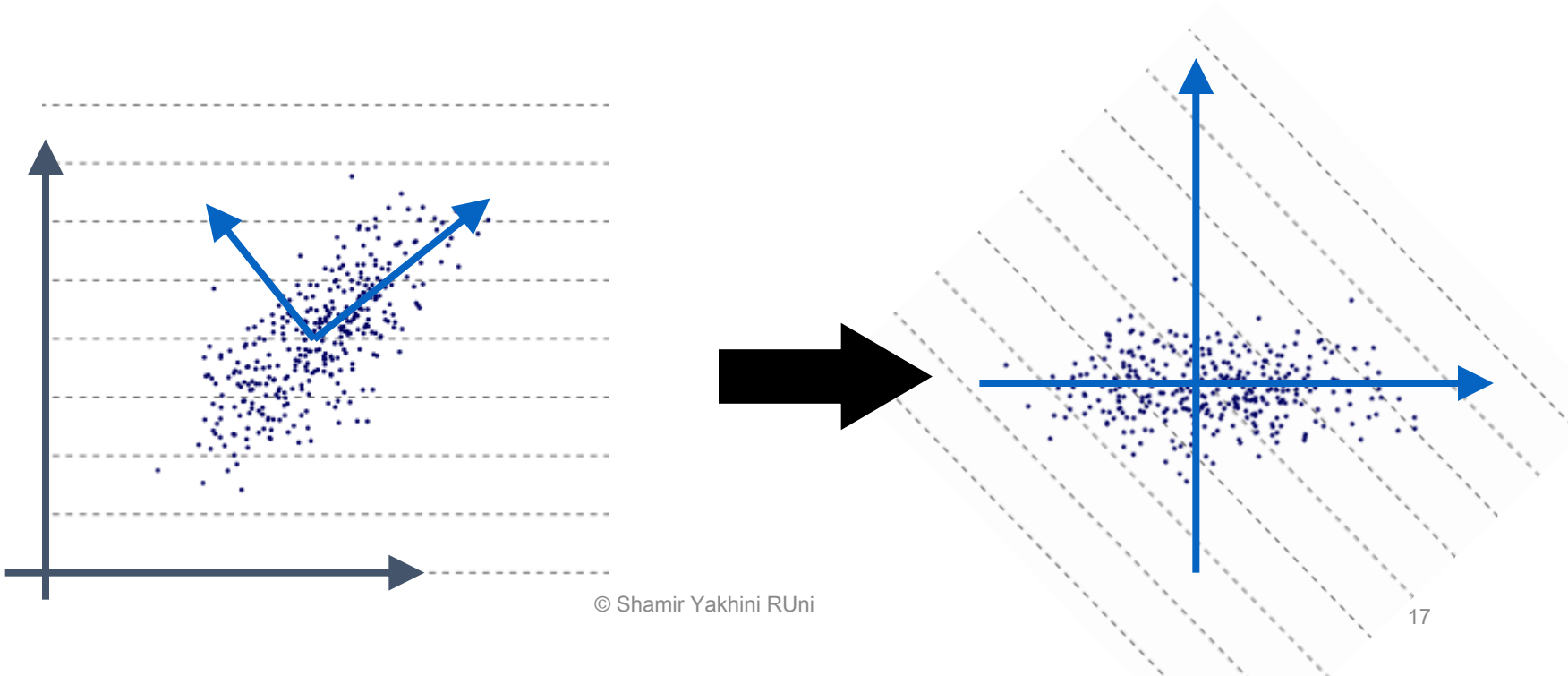
  (orthogonal = rotation)
- The new coordinates are called Principal Components.
- Usually, a small number of PCs are needed to describe most of the structure in the data (structure = variance of the data).

PCA Applications

- Modeling and dimensionality reduction
- Compaction/Compression
- Outlier detection
- Separation of signal and noise
- Variable selection
- Classification

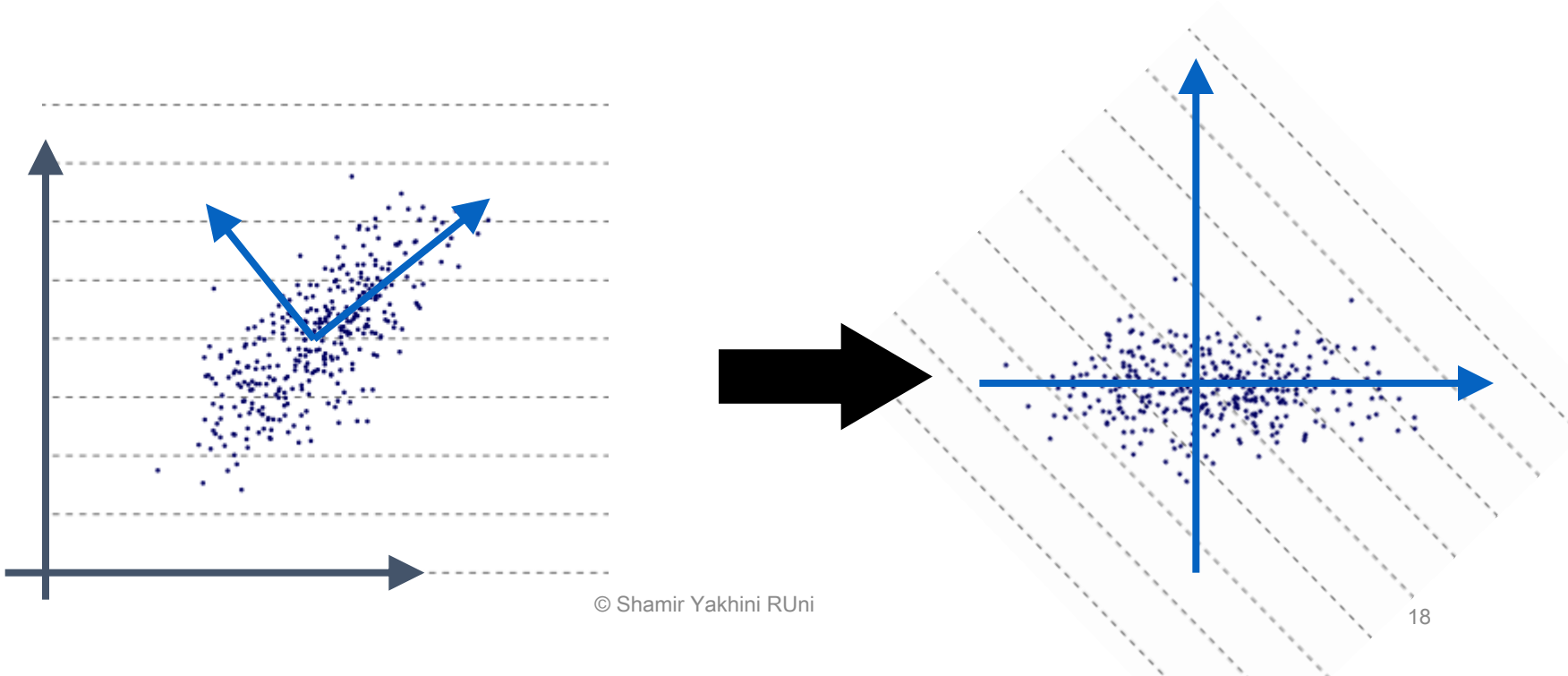
Illustrating PCA

- Sometimes the data lies in a low dimensional sub-space
- The new origin is the mean of the data
- The new axes are a rotation of the old axes



Filtering Low Variance Features

- In the “old space” both features have the same variance – impossible to get rid of any of them
- In the “new space” one has very large variance while the other very small so… we can get rid of the small variance one!



A Measure of Dependency between Features is the Sample Covariance

- Assume i and j are two **features** (dimensions)
- The mean of each feature is

$$\mu_j = \frac{1}{m}\sum_{k=1}^{m} x_j^{(k)} \quad \text{and} \quad \mu_i = \frac{1}{m}\sum_{k=1}^{m} x_i^{(k)}$$

- The sample covariance of i and j is:

$$\sigma_{ij} = \frac{1}{m}\sum_{k=1}^{m}\left(x_i^{(k)} - \mu_i\right)\left(x_j^{(k)} - \mu_j\right) = \sigma_{ji}$$

- If features i and j are statistically independent then $\sigma_{ij} = 0$, and when $\sigma_{ij} \neq 0$ then there is some dependency
- **Note: not the other way around!** features can have $\sigma_{ij} = 0$ but still be statistically dependent

The Covariance Matrix

- We defined $S = \frac{1}{m}\sum_{k=1}^{m}\left(\vec{x}^{(k)} - \mu\right)\left(\vec{x}^{(k)} - \mu\right)^T$

$$\vec{x}^{(k)} = \left(x_1^{(k)}, x_2^{(k)}, \dots, x_n^{(k)}\right) \qquad \mu = (\mu_1, \mu_2, \dots, \mu_n) = \frac{1}{m}\sum_{k=1}^{m}\vec{x}^{(k)}$$

$$S = \begin{bmatrix} \sigma_{11} & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_{nn} \end{bmatrix} = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & \sigma_2^2 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & \sigma_n^2 \end{bmatrix}$$

Variance

Key Idea: De-correlation

- Remember the covariance matrix represents the linear relations between pairs of features in the data
- "De-correlate" means removing correlation (linear dependencies) between features in the data
- How can this be done?
- Two features are not correlated if their covariance $\sigma_{ij} = 0$
- All features are decorrelated if the covariance matrix is diagonal – all off-diagonal entries are 0
- Can we find a transformation T that would convert the covariance matrix to a diagonal one?

Eigenvectors

- A non-zero vector v is an eigenvector of matrix M with eigenvalue λ if Mv = λv.
- By the spectral theorem we know that since S is n by n real and symmetric, we can find n orthogonal eigenvectors with n eigenvalues.
- We can arrange $\lambda_1,\lambda_2,\ldots\lambda_n$ in descending order
- We create a matrix A whose rows are the eigenvectors of S in descending order
- A is **orthogonal** ($A^T = A^{-1}$) i.e. it is a rotation matrix (note that in A^T the **columns** are the eigenvectors).

Hotelling transform

- We use A to transform the feature vectors x to a new feature vectors x':

$$x' = A(x - \mu)$$

- Note that $\mu_{x'} = \varepsilon[x'] = 0$
- The covariance matrix of x' vectors is

$$S' = \frac{1}{m}\sum_{k=1}^{m}(x' - \mu_{x'})^T(x' - \mu_{x'}) = \frac{1}{m}\sum_{k=1}^{m}(x')(x')^T$$

$$= \frac{1}{m}\sum_{k=1}^{m} A(x - \mu)(x - \mu)^T A^T$$

$$= A\left[\frac{1}{m}\sum_{k=1}^{m}\left(x^{(k)} - \mu\right)\left(x^{(k)} - \mu\right)^T\right]A^T = ASA^T$$

What is ASA^T?

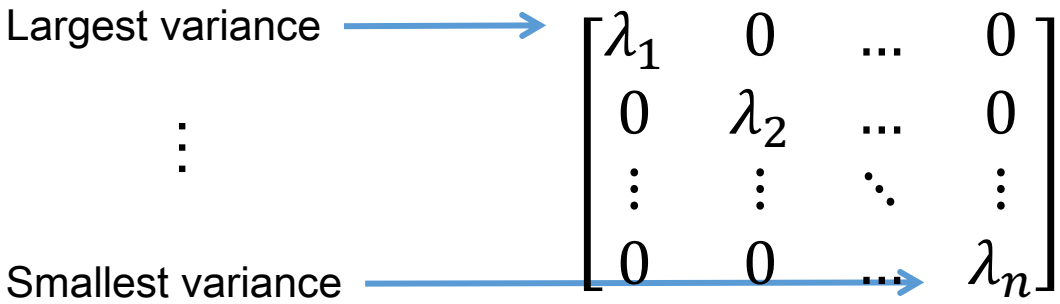
- The colums of $A^T = [v_1\ v_2 \cdots v_n]$ are eigenvectors of S so
  $SA^T = [\lambda_1 v_1\ \lambda_2 v_2 \cdots \lambda_n v_n]$ (a matrix) and

$$ASA^T = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \cdot [\lambda_1 v_1\ \lambda_2 v_2 \cdots \lambda_n v_n] =$$

$$= \begin{bmatrix} \lambda_1 v_1 v_1 & \lambda_1 v_1 v_2 & \dots & \lambda_n v_1 v_n \\ \lambda_1 v_2 v_1 & \lambda_2 v_2 v_2 & \dots & \lambda_n v_2 v_n \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1 v_n v_1 & \lambda_2 v_n v_2 & \dots & \lambda_n v_n v_n \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_n \end{bmatrix}$$

Diagonal Matrix

- Therefore, we found that S', the covariance matrix of x', is a diagonal matrix of the form:



- We also know that $A^{-1} = A^T$ and so we can get back to x by using:

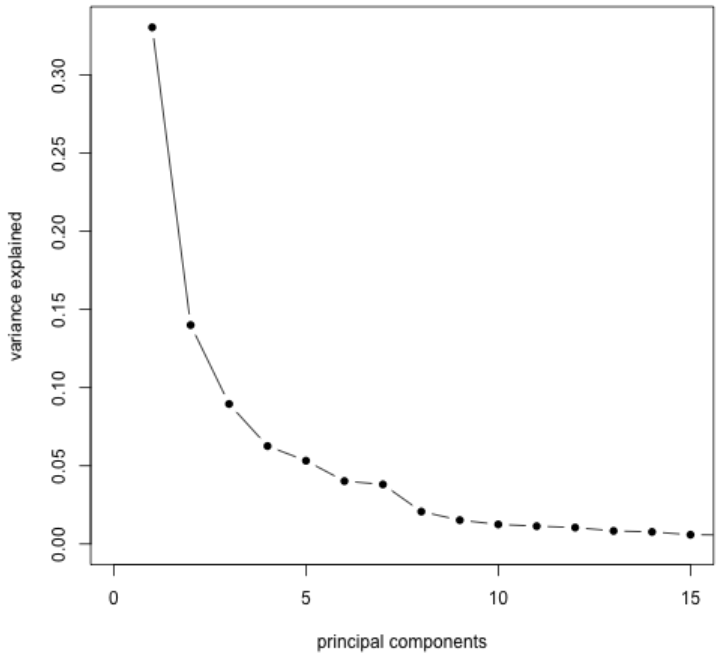
$$x = \mu_x + A^T x'$$

Sorting Eigenvalues = Decreasing Variance

- S' is the covariance matrix of the transformed features
- Since outside the diagonal the entries are zero – the new features are de-correlated!
- The entries in the diagonal are the variance of each dimension.
- Since they are sorted from largest to smallest it means that the variance of the first feature is the largest, then the second, then the third and so on…
- This means that the last features almost have no variance – and we can get rid of them!

Dimensionality Reduction (& Compression)

- Instead of using all n new feature values (PC's), the last features almost have no variance and therefore their values could be dropped without loosing much information



PCA Algorithm

1. Build the covariance matrix $\frac{1}{m}\sum_{k=1}^{m}\left(\vec{x}^{(k)} - \mu\right)\left(\vec{x}^{(k)} - \mu\right)^T$
2. Find Eigenvectors and Eigenvalues of S by solving $(S - \lambda_i I)\, e_i = 0$
3. Sort Eigenvectors by Eigenvalues
4. Build the matrix A_j using j greatest eigenvectors
5. Transform $x' = A_j(x - \mu)$

- If n, the feature space dimension (also the size of the covariance matrix nxn), is large, then it is often more efficient to only find the top eigenvalues/eigenvectors instead of all of them

Approximation/Reduction/Compression

$$\mathbf{x} = \mu_{\mathbf{x}} + \mathbf{A}^{\mathbf{T}} x' = \mu_{\mathbf{x}} + \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} x'_1 \\ \vdots \\ x'_n \end{pmatrix} =$$

$$\mu_{\mathbf{x}} + x'_1 \begin{pmatrix} a_{11} \\ \\ a_{n1} \end{pmatrix} + x'_2 \begin{pmatrix} a_{12} \\ \\ a_{n2} \end{pmatrix} + \ldots + x'_n \begin{pmatrix} a_{1n} \\ \\ a_{nn} \end{pmatrix} =$$

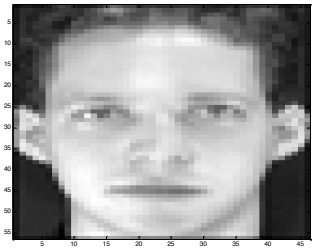
$$\mu_{\mathbf{x}} + \underbrace{x'_1 \vec{e}_1 + x'_2 \vec{e}_2}_{} + \cdots + x'_n \vec{e}_n$$

Use just k coefficients $x'_1 \ldots x'_k$

Example: Images

46

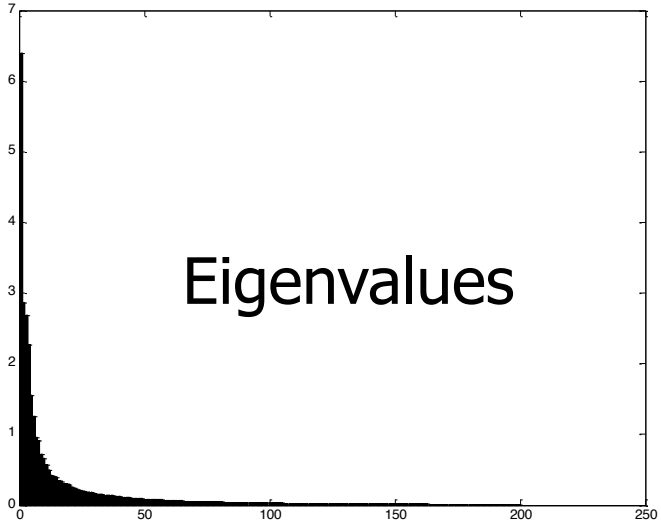
56



→ 2576 dimension vectors

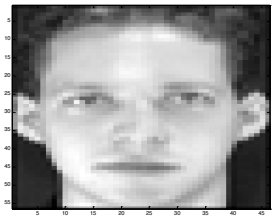
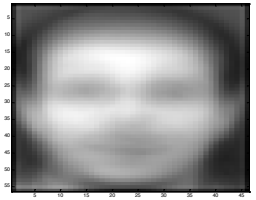
$$\vec{\mathbf{x}} \equiv (x_{1,1}, x_{1,2}, \ldots, x_{1,46}, x_{2,1}, x_{2,2}, \ldots, x_{2,46}, \ldots, x_{56,1}, x_{56,2}, \ldots, x_{56,46})$$

→

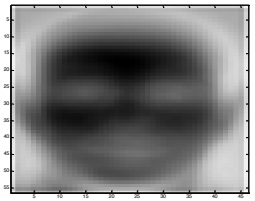
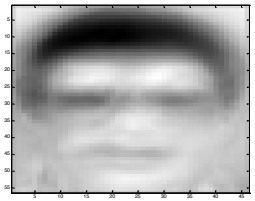
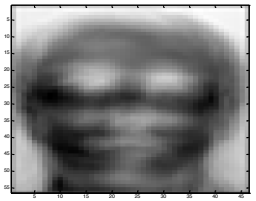


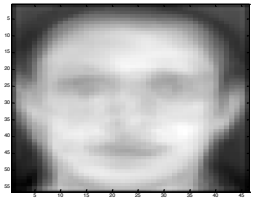
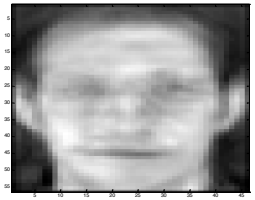
EigenImages

$$\mathbf{x} = \mu_{\mathbf{x}} + \mathbf{A}^{\mathbf{T}} x' = \mu_{\mathbf{x}} + x'_1 \vec{e}_1 + x'_2 \vec{e}_2 + \cdots + x'_n \vec{e}_n$$


=
μ_x

+

e_1 e_2 e_3

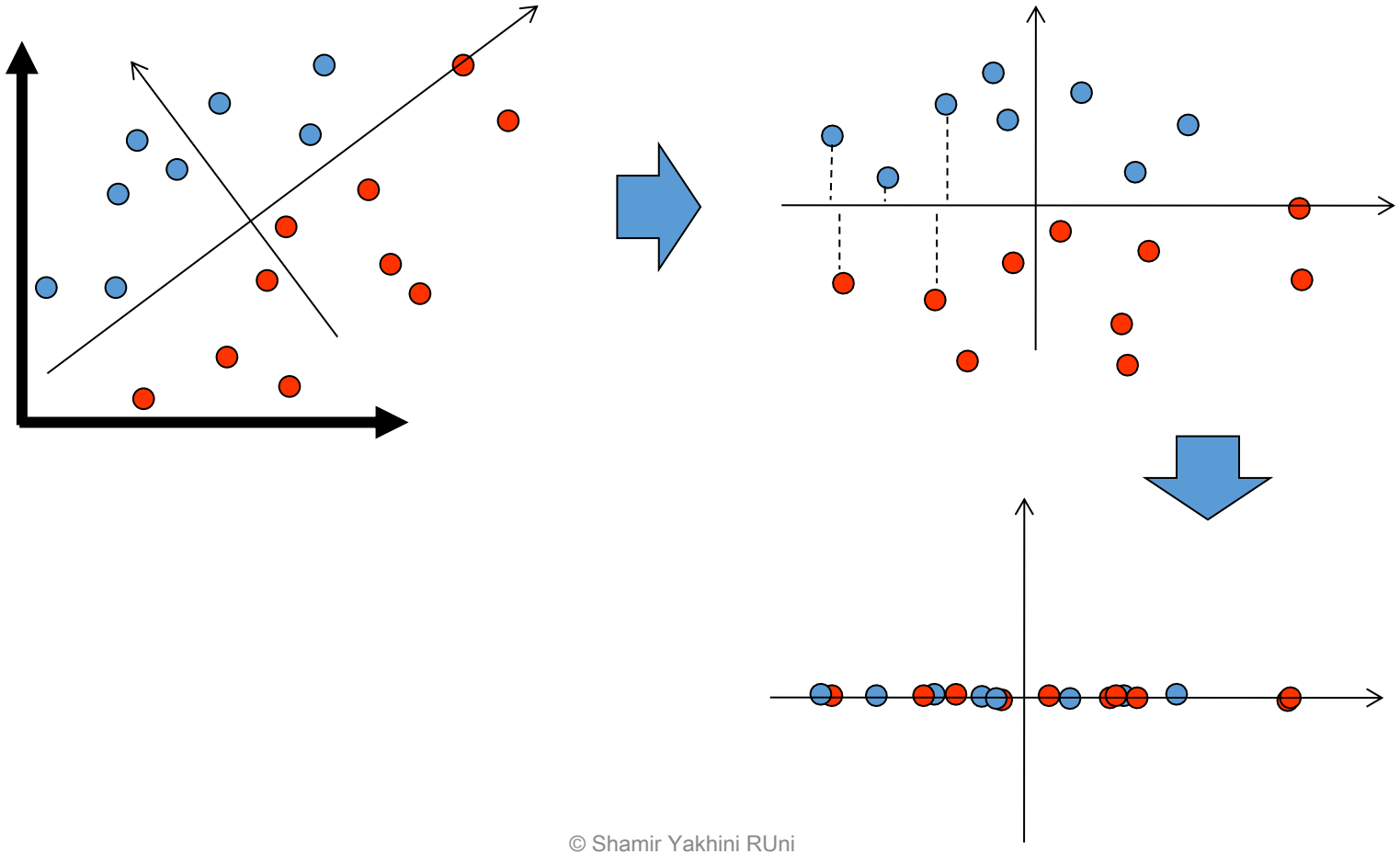
$+ c_1$  $+ c_2$  $+ c_3$  $+ \ldots$

10 ev→ 
50 ev→ 
…
All ev→ 

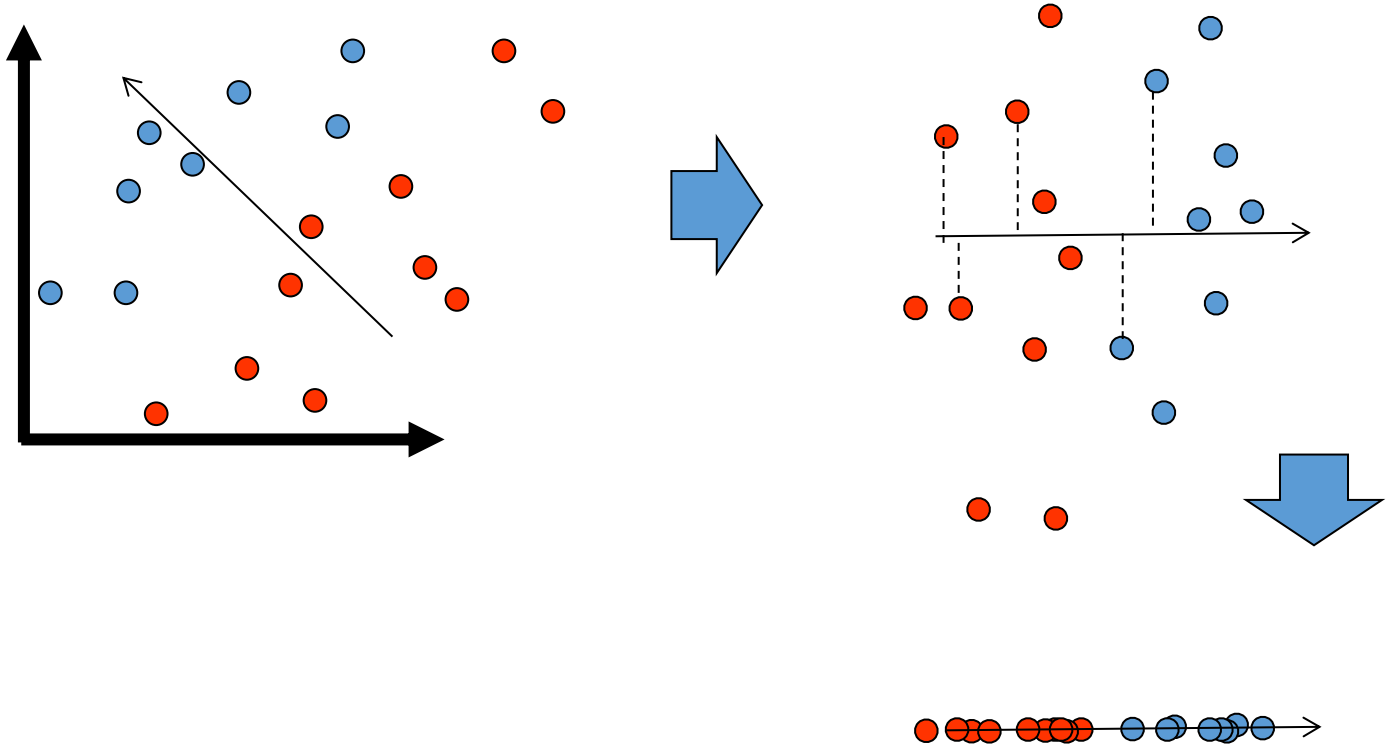
PCA Vs. Discriminant Analysis

- PCA finds components that are useful to represent data.
- Are these components useful to discriminate between data in different classes?
- Not necessarily!
- Instead of principal components analysis we may be interested in discriminant analysis

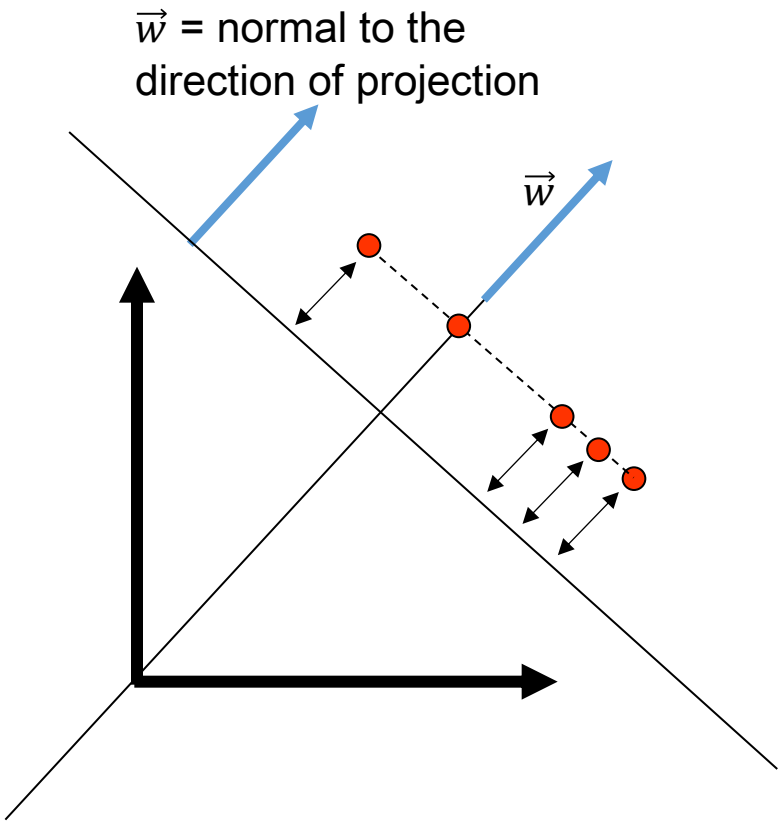
PCA vs. LDA



PCA vs. LDA



Projecting Onto a Line whose Direction is w

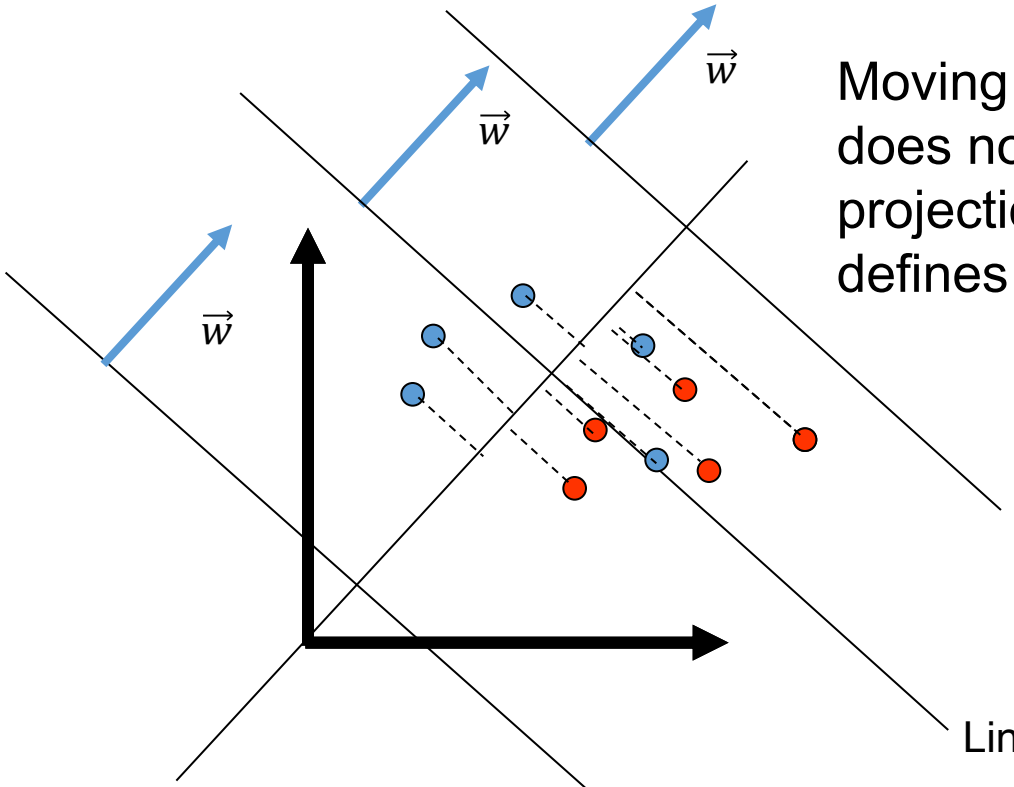


All points with the same distance to the line

$$f(x) = \vec{w}x + w_0 = C$$

for any constant C,
will be projected to the same point on the perpendicular line = the line whose direction is $\vec{w}$.

Projecting Onto a Line



Moving the line f(x) in parallel does not change the order of projection points so $\vec{w}x$ defines the projection

Line $\mathrm{f(x)} = \vec{w}x + w_2 = 0$

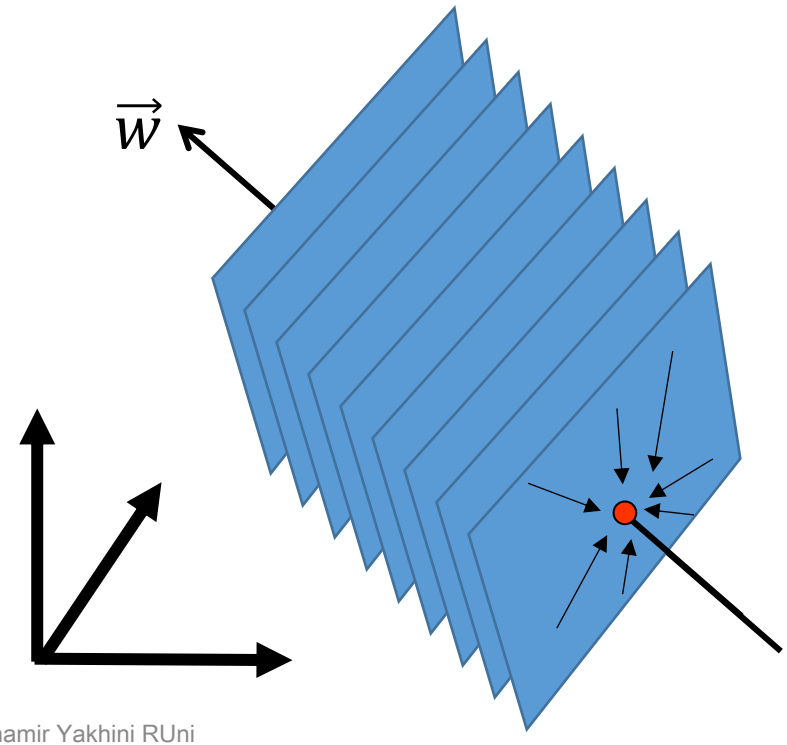
Line $\mathrm{f(x)} = \vec{w}x + w_0 = 0$

Line $\mathrm{f(x)} = \vec{w}x + w_1 = 0$

Illustration in 3D

- The projection line is defined by the normal to all the planes $\vec{w}x$
- All points on such planes will be projected to the same point
- The “order” of the planes will be preserved on this line

- Same in N-dimensions



Projecting to a Line

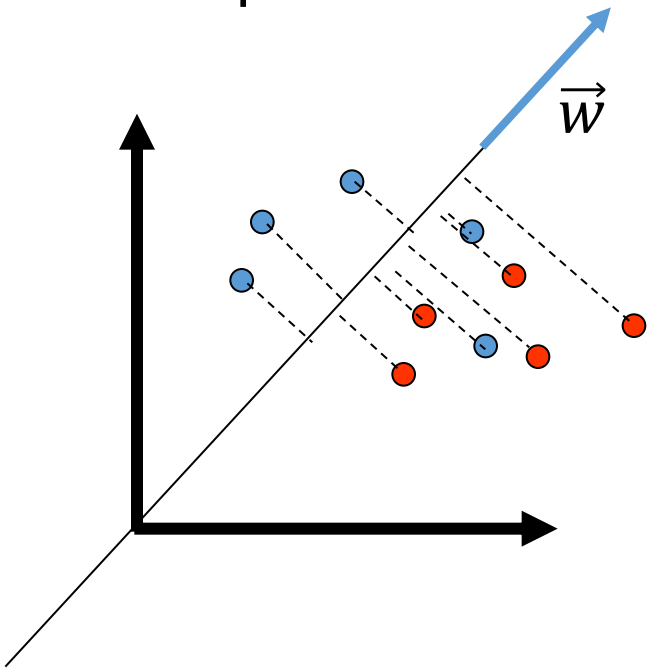
- We project from n dimension to a line in 1D using the equation

$$y = \vec{w}x$$

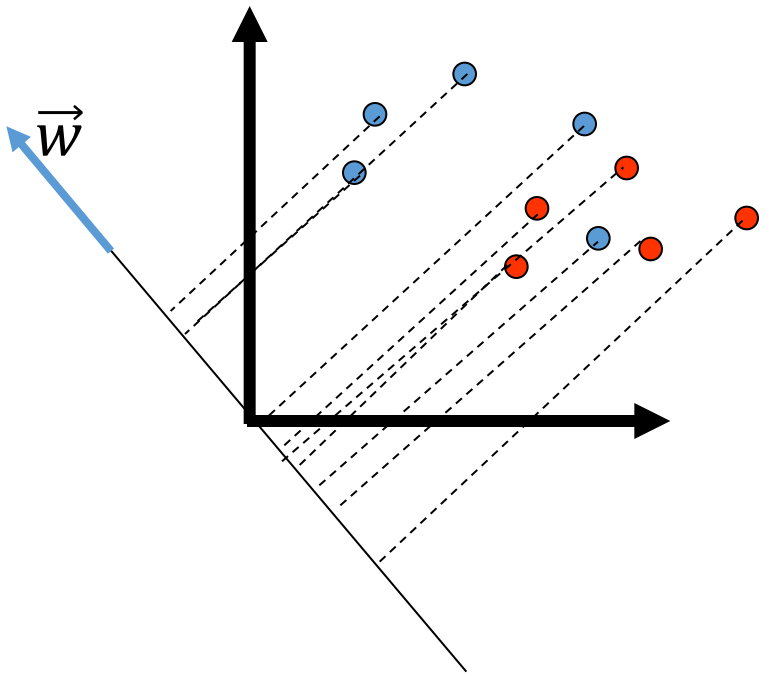
- Even if the samples are well separated in n-dimension they will usually produce a mixture on the projected line
- By rotating the line direction (i.e. $\vec{w}$), we might find an orientation where the projected samples are well separated

Line Projection Examples

Worse separation



Better separation



Maximize Distance Between Classes

- Find the line direction that maximizes the separation between the classes and classify in 1D!
- First step: define the line direction by maximizing the distance between cluster (class) means

$$\mu_0 = \frac{1}{n_0}\sum_{x \in D_0} x \qquad \mu_1 = \frac{1}{n_1}\sum_{x \in D_1} x$$

n-dimensional space

$$\tilde{\mu}_0 = \frac{1}{n_0}\sum_{y \in Y_0} y = \frac{1}{n_0}\sum_{x \in D_0} w^T x = w^T \mu_0$$

$$\tilde{\mu}_1 = \frac{1}{n_1}\sum_{y \in Y_1} y = \frac{1}{n_1}\sum_{x \in D_1} w^T x = w^T \mu_1$$

1-dimensional space

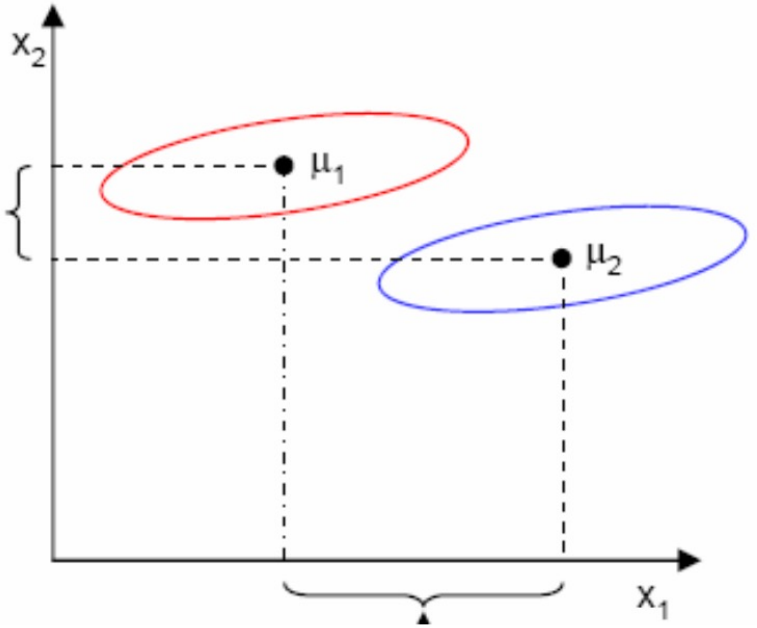
$$\Rightarrow \quad \text{maximize}: \quad \left(\tilde{\mu}_0 - \tilde{\mu}_1\right)^2 = \left(w^T\left(\mu_0 - \mu_1\right)\right)^2$$

$$\Rightarrow \quad \text{maximize} \quad w^T\left(\mu_0 - \mu_1\right)\left(\mu_0 - \mu_1\right)^T w = w^T S_B w$$

Tradeoff

- Is this enough?
- We also want the classes to be compact!

This axis is better for class compactness



This axis is good for large distance between classes mean

Minimize Difference within Each Class

$$\forall \text{i minimize } \tilde{s}_i^2 = \sum_{y \in Y_i} \left(y - \tilde{\mu}_i \right)^2 = \sum_{x \in D_i} \left(w^T x - w^T \mu_i \right)^2 =$$

$$= w^T \left(\sum_{x \in D_i} \left(x - \mu_i \right)\left(x - \mu_i \right)^T \right) w$$

$$\Rightarrow \text{minimize} \sum_{i=0,1} \left[w^T \left(\sum_{x \in D_i} \left(x - \mu_i \right)\left(x - \mu_i \right)^T \right) w \right]$$

$$\Rightarrow \text{minimize } w^T \sum_{i=0,1} \left[\left(\sum_{x \in D_i} \left(x - \mu_i \right)\left(x - \mu_i \right)^T \right) \right] w = w^T S_C w$$

Scatter Matrices

Define the between class scatter :

$$S_B = (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T$$

Define the within class scatter : $S_C = S_0 + S_1$

where $S_i = \sum_{x \in D_i} (x - \mu_i)(x - \mu_i)^T$

$$\text{maximize } (\tilde{\mu}_0 - \tilde{\mu}_1)^2 \Rightarrow \text{maximize } w^T (\mu_0 - \mu_1)(\mu_0 - \mu_1)^T w$$

$$\Rightarrow \text{maximize } w^T S_B w$$

$$\forall \text{i minimize } w^T \left(\sum_{x \in D_i} (x - \mu_i)(x - \mu_i)^T \right) w \Rightarrow \text{minimize } w^T S_C w$$

Optimization Objective

- We want to find w that will maximize J(w)

$$\max_{\mathrm{w}} J(\mathrm{w}) = \frac{(\tilde{\mu}_0 - \tilde{\mu}_1)^2}{\tilde{s}_0^2 + \tilde{s}_1^2} = \frac{w^T S_B w}{w^T S_C w}$$

Between class scatter

Within class scatter

- To do that we differentiate and equate to 0
- Note: this can be viewed as $\frac{(mean_0 - mean_1)^2}{var_0 + var_1}$
- What is this different from Clustering?

Differentiating

$$\frac{d}{dw}J(w) = \frac{d}{dw}\frac{w^T S_B w}{w^T S_C w} = 0$$

$$\Rightarrow (w^T S_C w)\frac{d}{dw}(w^T S_B w) - (w^T S_B w)\frac{d}{dw}(w^T S_C w) = 0$$

$$\Rightarrow (w^T S_C w)2S_B w - (w^T S_B w)2S_C w = 0$$

- Dividing by $2w^T S_C w$ we get:

$$\Rightarrow \frac{w^T S_C w}{w^T S_C w}S_B w - \frac{w^T S_B w}{w^T S_C w}S_C w = 0$$

$$\Rightarrow S_B w - \mathrm{J(w)S_C w} = 0$$

$$\Rightarrow S_C^{-1} S_B w - \mathrm{J(w)w} = 0$$

$$\Rightarrow S_C^{-1} S_B w = \mathrm{J(w)w}$$

Solution

This problem is called the generalized eigenvector problem

$S_C^{-1} S_B w = \lambda \mathrm{w}$ where $\lambda = \mathrm{J}(\mathrm{w})$

The solution w is an eigenvector of $S_C^{-1} S_B$ and can be found as:

$\mathrm{w} = S_C^{-1}(\mu_0 - \mu_1)$

This is called **Fisher linear discriminant** (even though it is a direction of projection and not the actual discriminant function)

Ronald Fisher

Fisher Linear Discriminant

1. Define the within class scatter (compactness): $S_C = S_0 + S_1$

$$S_i = \sum_{x \in D_i} (x - \mu_i)(x - \mu_i)^T$$

2. Solve $w = S_C^{-1}(\mu_0 - \mu_1)$

3. Find the best threshold (which means find w_0)

Decide: $w^T x > threshold$

$\Rightarrow$ Use: $w^T x - w_0 > 0$

Extending to C classes

- The within class scatter remains the same but now we use C classes in the summation:
- $S_C = \sum_{i\ classes} S_i = \sum_{i\ classes} \left(\sum_{x \in D_i} (x - \mu_i)(x - \mu_i)^T \right)$
- The between class scatter is measured as the (weighted) difference of each mean from the global mean:

$$S_B = \sum_{i\ classes} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

- Where N_i is the number of instances in class i and the global mean is $\mu = \sum_{x \in D} x$

LDA Algorithm (multi-class multi-dimension reduction)

1. Define the within class scatter (compactness) : $S_C = \sum_i S_i$

$$S_i = \sum_{x \in D_i} (x - \mu_i)(x - \mu_i)^T$$

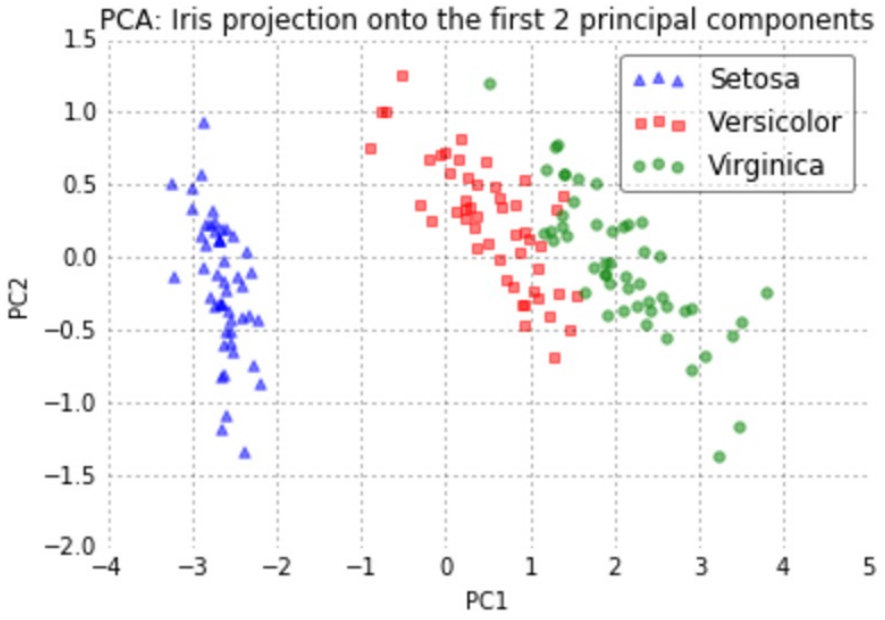
2. Define the between class scatter :

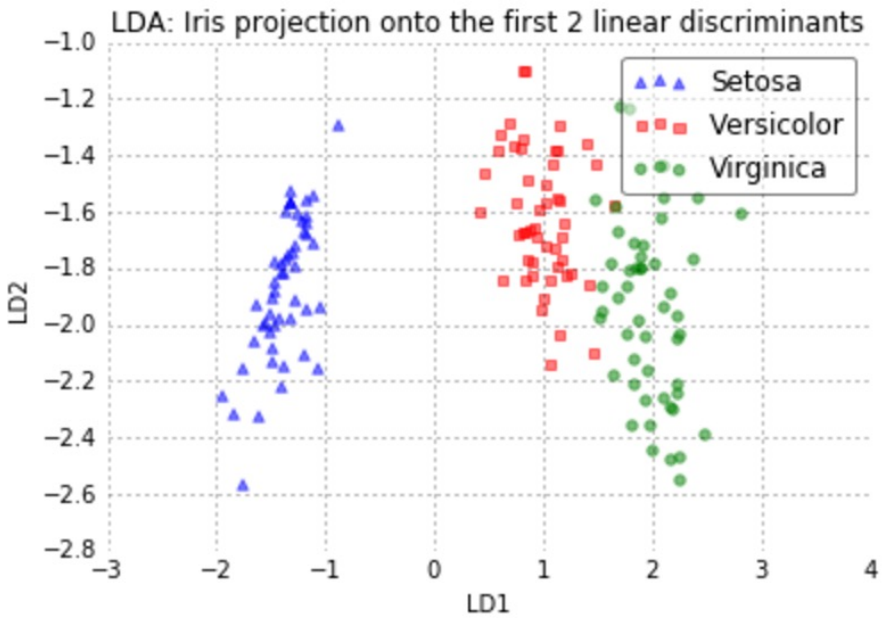
$$S_B = \sum_i N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

3. *F*ind eigenvectors of $S_C^{-1} S_B$
4. *S*ort them and transform instances to a subspace
5. Find classifier in this subspace

PCA vs. LDA

- Where the PCA accounts for the most variance in the whole dataset, the LDA gives us the axes that account for the most variance between the individual classes.

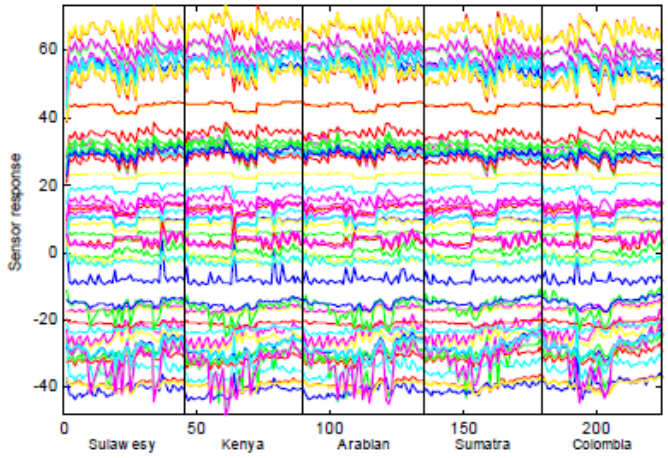


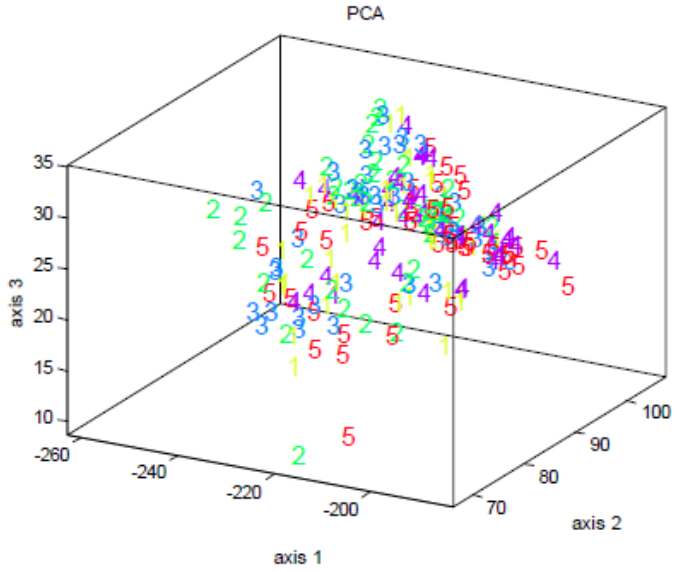


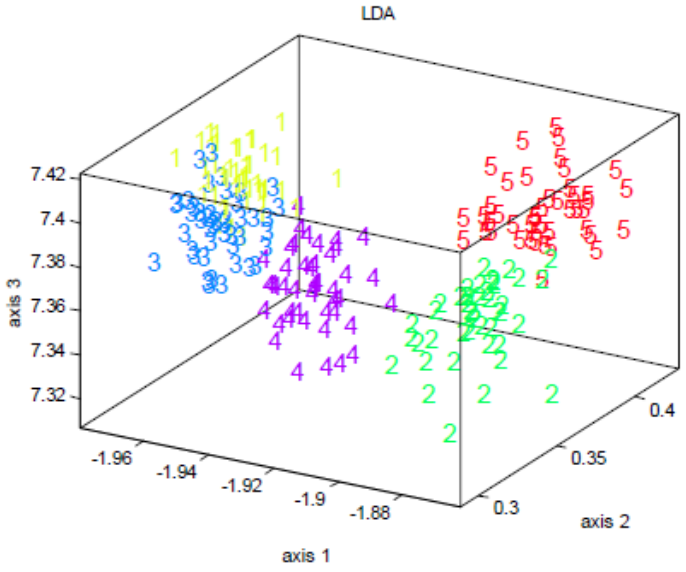
Coffee and LDA

LDA Vs. PCA: Coffee discrimination with a gas sensor array

- **These figures show the performance of PCA and LDA on an odor recognition problem**
  - Five types of coffee beans were presented to an array of chemical gas sensors
  - For each coffee type, 45 "sniffs" were performed and the response of the gas sensor array was processed in order to obtain a 60-dimensional feature vector
- **Results**
  - From the 3D scatter plots it is clear that LDA outperforms PCA in terms of class discrimination
  - This is one example where the discriminatory information is not aligned with the direction of maximum variance

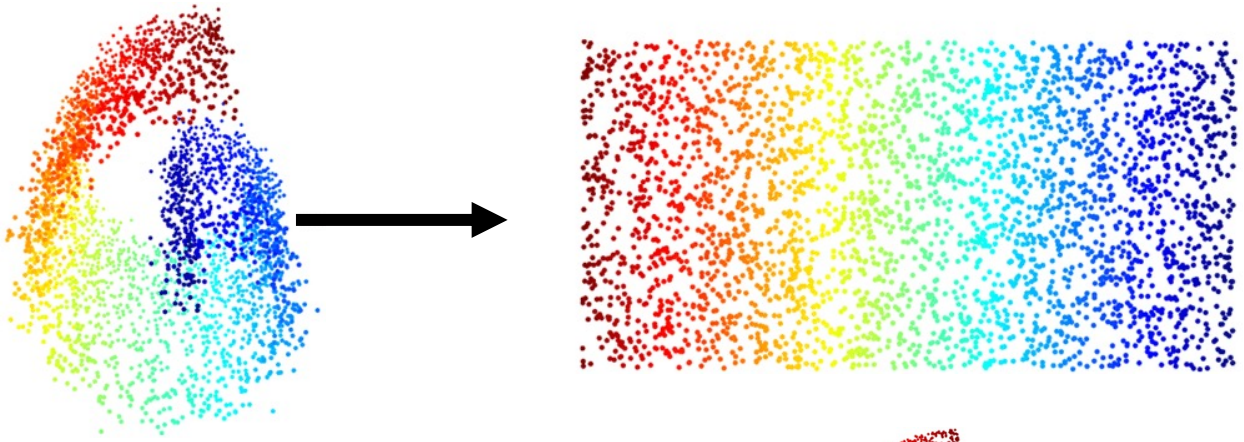






Non-Linearity

- Both PCA and LDA are **linear** transformations of the data
- Sometimes we want to map the data in a non-linear fashion…



”Swiss Role”

Dimensionality Reduction (Mapping) as Optimization

We want to map high-dimensional data to a lower dimension preserving some properties by defining a cost function and minimizing/maximizing it

Find a mapping

$$\mathcal{X} = \{x_1, x_2, \dots, x_m \in \mathbb{R}^h\} \rightarrow \mathcal{Y} = \{y_1, y_2, \dots, y_m \in \mathbb{R}^l\}$$

where $l \ll h$, while $\min_{\mathcal{Y}} C(\mathcal{X}, \mathcal{Y})$

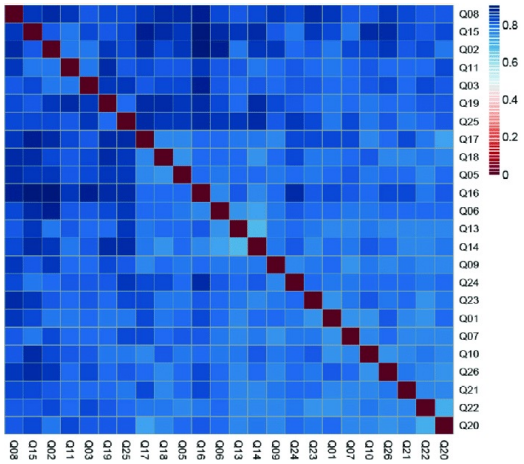
PCA – optimizes informaion preservation (linear)

LDA – optimizes class seperability (linear)

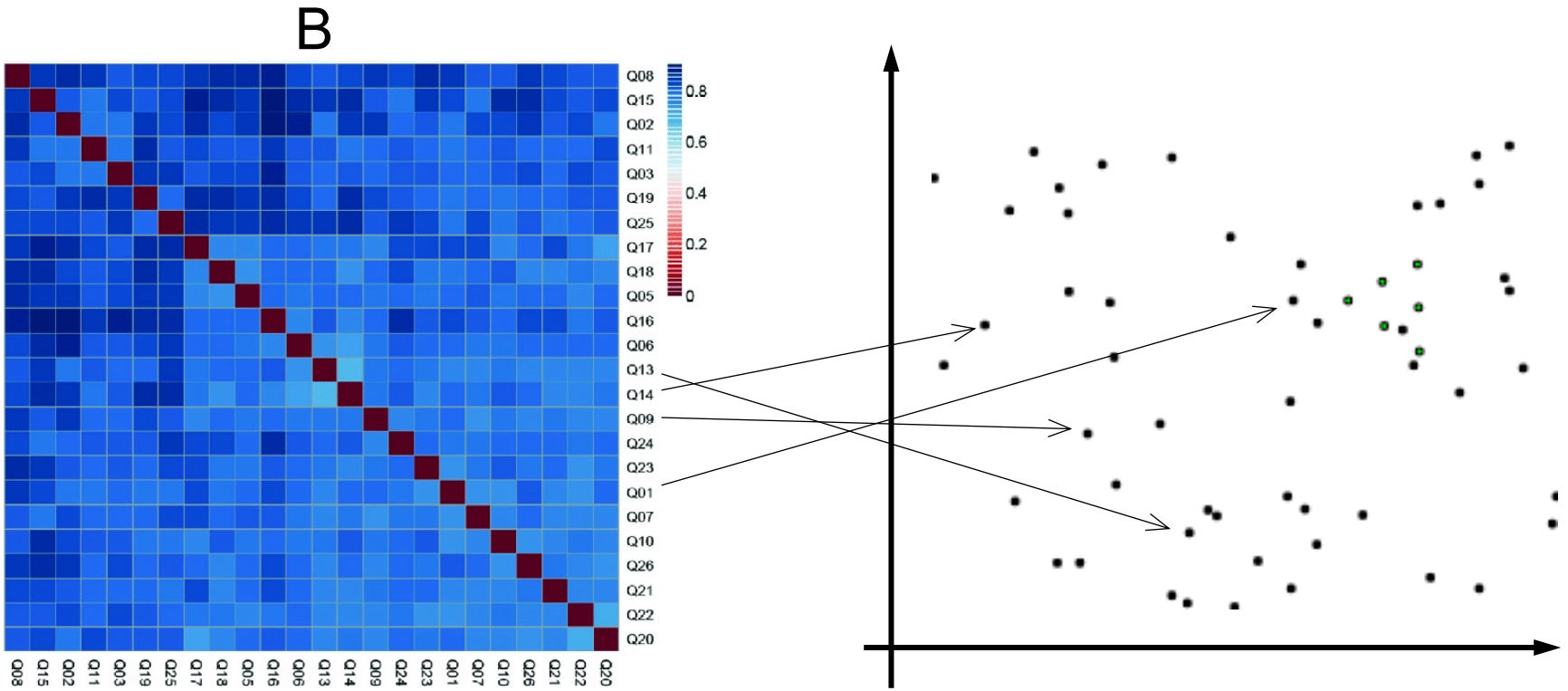
Next: **MDS** – preserves instance distances (non-linear)

Multi Dimensional Scaling (MDS)

- We have the (dis)similarity scores between a set of instances – i.e. a similarity matrix.
- The scores are not necessarily Euclidean distances, and in fact not necessaritly define a metric
- **Our goal** is to map the instances into points in Euclidean space where the pairwize Euclidean distances will reflect as much as possible the similarity scores in the matrix



From Distances to Positions



How can this be done?

Stress Optimization

Given similarity score matrix $B = \{b_{ij}\}$ between all items in D we map each item i to point x_i such that the following function is **minimized**:

$$Stress_D\{x_1, x_2, \dots, x_n\} = \left(\sum_{\{i,j\}} \left(b_{ij} - \|x_i - x_j\|\right)^2\right)^{\frac{1}{2}}$$

Similarity in B

Distance in Space

Some versions of this problem can also be solved using eigenvector decomposition, and some using other types of optimization

Metric vs. Non-Metric

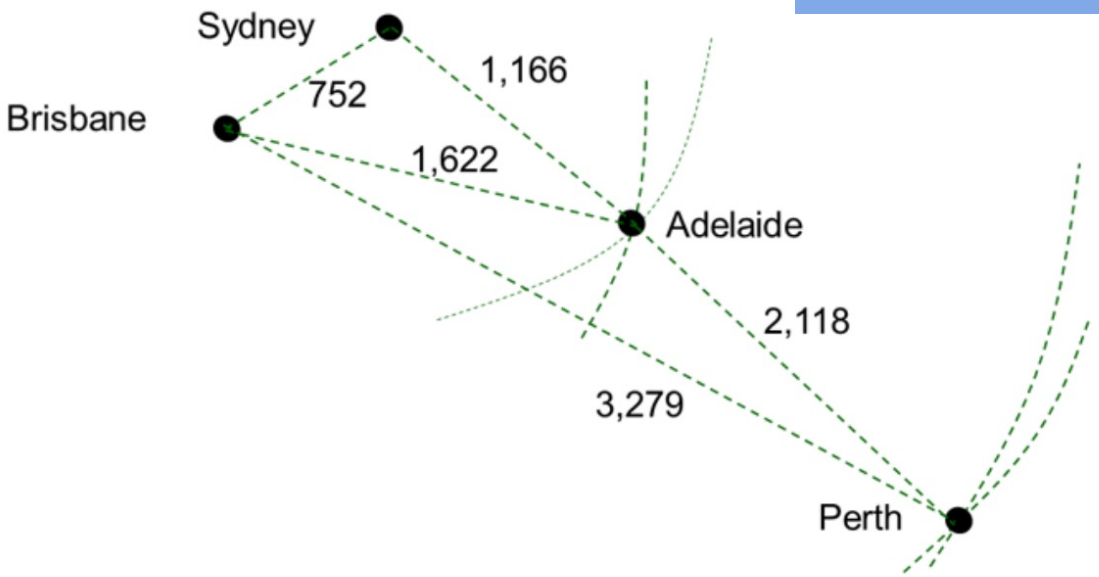
- In case the distances in B are metric, then we map to lower dimension and try to **match** the distances – i.e. bring the stress as close to zero as much as possible

- In case the dissimilarities are non-metric, then we just want to **preserve the rank order** – i.e.:

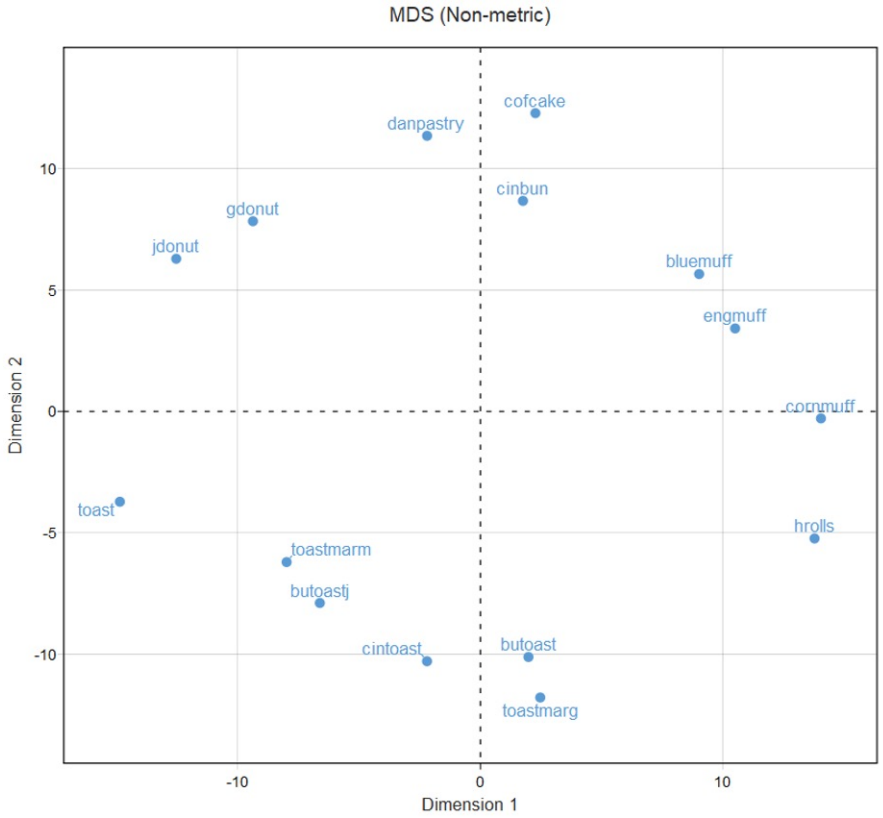
  If $b_{ij} < b_{pq}$ then $d(x_i, x_j) \leq d(x_p, x_q)$

Toy Metric Example

Adelaide	1,166		
Brisbane	752	1,622	
Perth	3,279	2,118	3,606
	Sydney	Adelaide	Brisbane



Toy Non-Metric Example



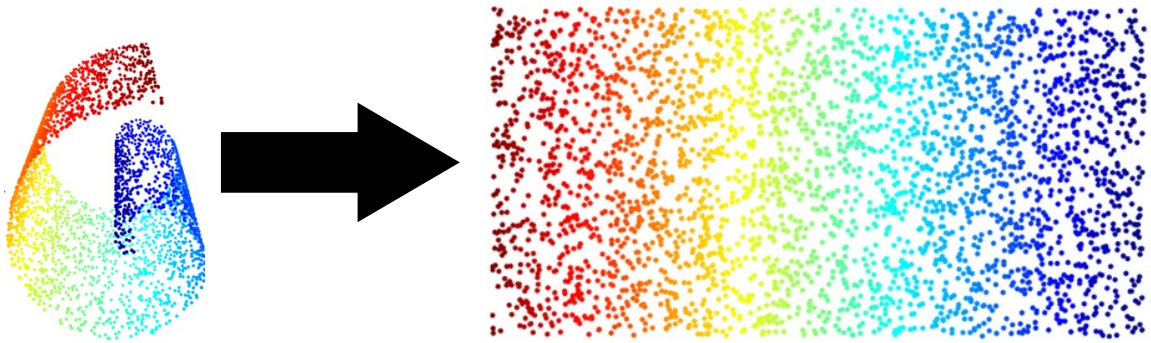
	toast	butoast	engmuff	jdonut	cintoast	bluemuff	hrolls	toastmarm	butoastj	toastmarg	cinbun	danpastry	gdonut	cofcake
butoast	15													
engmuff	25	15												
jdonut	3	24	22											
cintoast	14	3	17	22										
bluemuff	24	17	2	21	19									
hrolls	28	8	4	27	18	8								
toastmarm	7	7	20	11	6	18	23							
butoastj	8	6	21	12	5	19	22	2						
toastmarg	16	2	16	25	4	18	9	8	7					
cinbun	26	17	10	17	12	7	18	20	19	18				
danpastry	21	25	11	5	19	10	22	17	16	26	2			
gdonut	20	18	24	2	23	22	25	11	12	17	4	11		
cofcake	16	22	11	13	21	7	21	21	20	23	6	7	11	
cornmuff	27	11	3	26	16	4	5	25	24	12	12	16	24	16

Lower number indicates the pair of baked breakfast goods are highly similar.

The breakfast data comes from Green, Paul E. and Vithala R. Rao (1972), Applied Multidimensional Scaling: A Comparison of Approaches and Algorithms. New York: Holt, Rinehart and Winston.

Local vs. Global Structure

- Many times you do want to visualize your data
- You want to map it to low dimension (2,3) to see it
- Mapping from high dimension to low dimension looses a lot.
- You cannot preserve global structure but maybe you can preserve local structure? i.e. – neighboring data
- Swiss Role Example:



Stochastic Neighbor Embedding

- Converts high dimensional Euclidean distances to conditional probabilities that represent similarities
- Similarity in high dimensional space:

$$p_{j|i} = \frac{\exp(-\left\|x_i - x_j\right\|^2/\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/\sigma_i^2)}$$

- Similarity in low dimensional space:

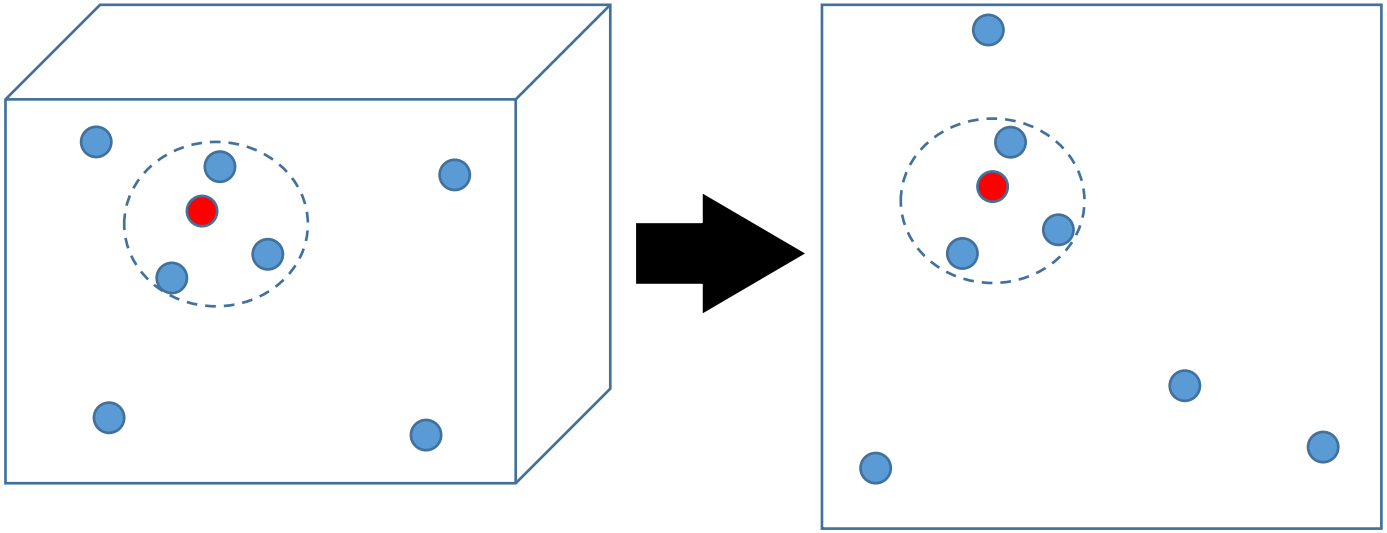
$$q_{j|i} = \frac{\exp(-\left\|y_i - y_j\right\|^2/\sigma_i^2)}{\sum_{k\neq i}\exp(-\|y_i - y_k\|^2/\sigma_i^2)}$$

- Cost function:

$$C = \sum_i KL(P_i||Q_i) = \sum_i\sum_j p_{j|i}\log\frac{p_{j|i}}{q_{j|i}}$$

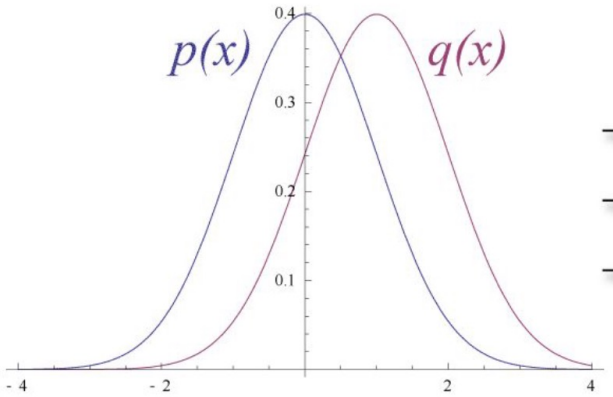
Key Idea

- Since the probabilities depend on the distance – local neighborhood affects much more the mapping to low dimension
- Local structure would be preserved while global not necessarily

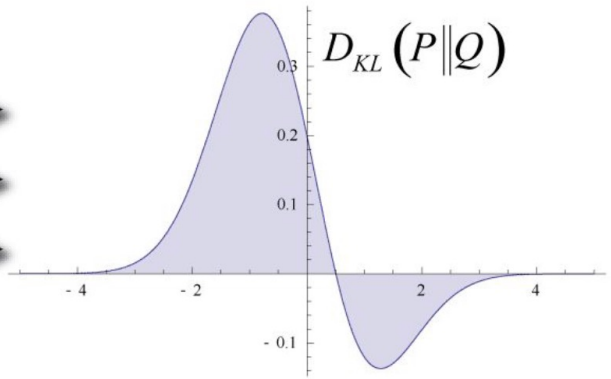


KL Divergence

- Measures the similarity betwee two probability distributions
- Asymetric: $D_{KL}(P||Q) \neq D_{KL}(Q||P)$



Original Gaussian PDF's



KL Area to be Integrated

Optimization

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$$

- The gradient has a simple form:

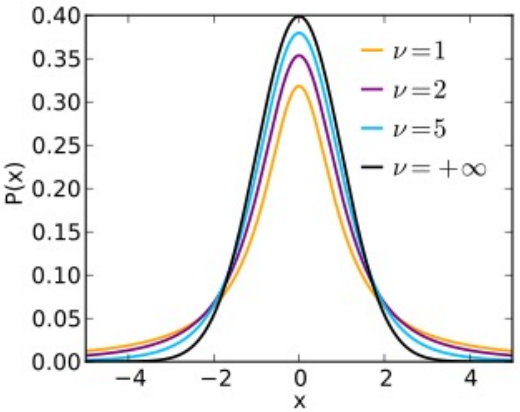
$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

T-distributed Stochastic Neighbor Embedding

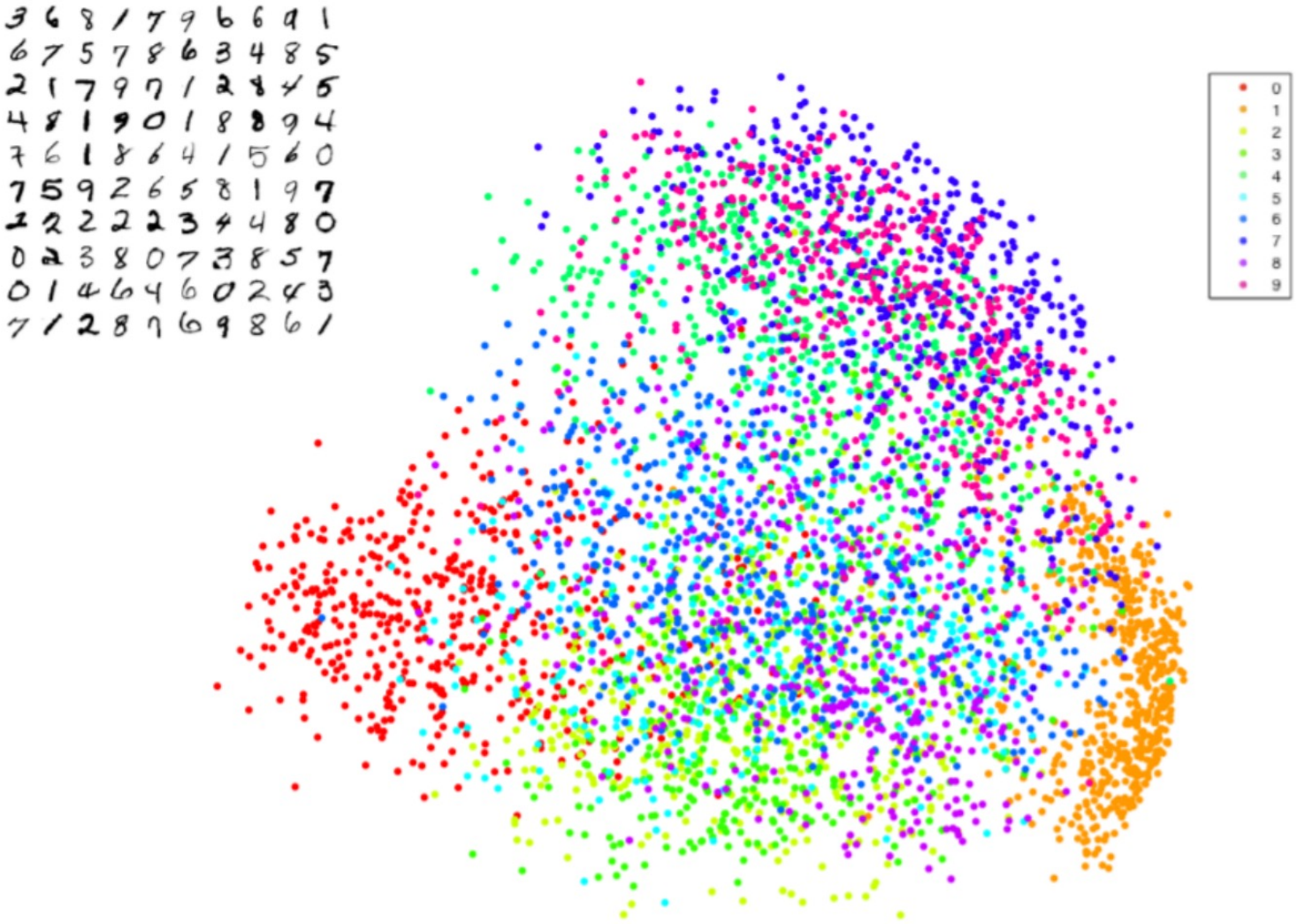
- Changed the distribution in low dimension to t-distribution which is a heavier tail distribution than Gaussian.

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

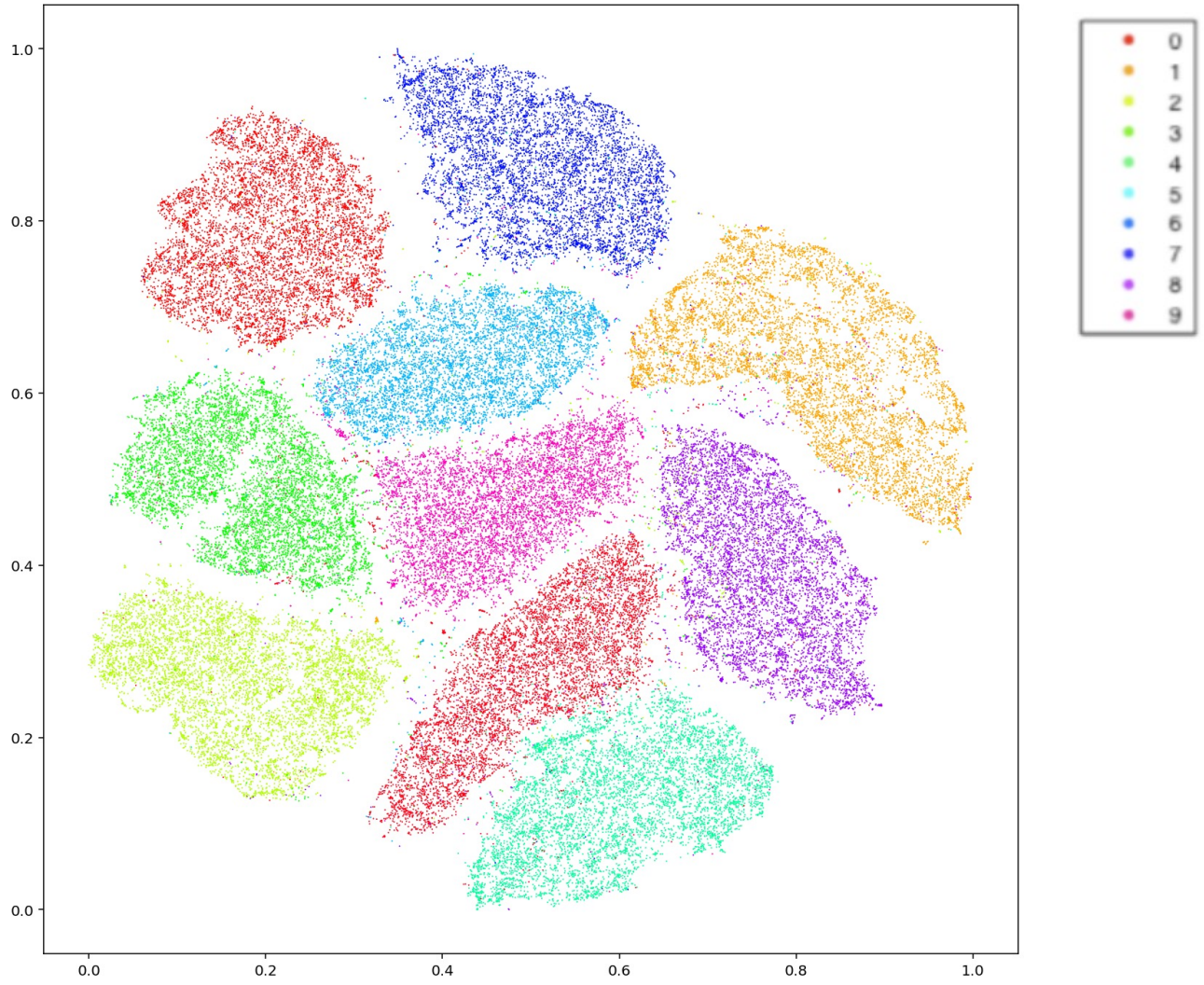
$$f(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi}\,\Gamma(\frac{\nu}{2})}\left(1 + \frac{t^2}{\nu}\right)^{-(\nu+1)/2}$$



PCA for MNIST



tSNE of M-NIST Dataset



Summary

Mapping from high-dim to low-dim

Linear:

PCA – optimizes informaion preservation

LDA – optimizes class seperability

Non-Linear

MDS – preserves instance distances

tSNE – preserves local distances/structure

Many many more…