#MAHSA_AMINI

MACHINE LEARNING

Electrical Summer Workshops (ESW) 2022

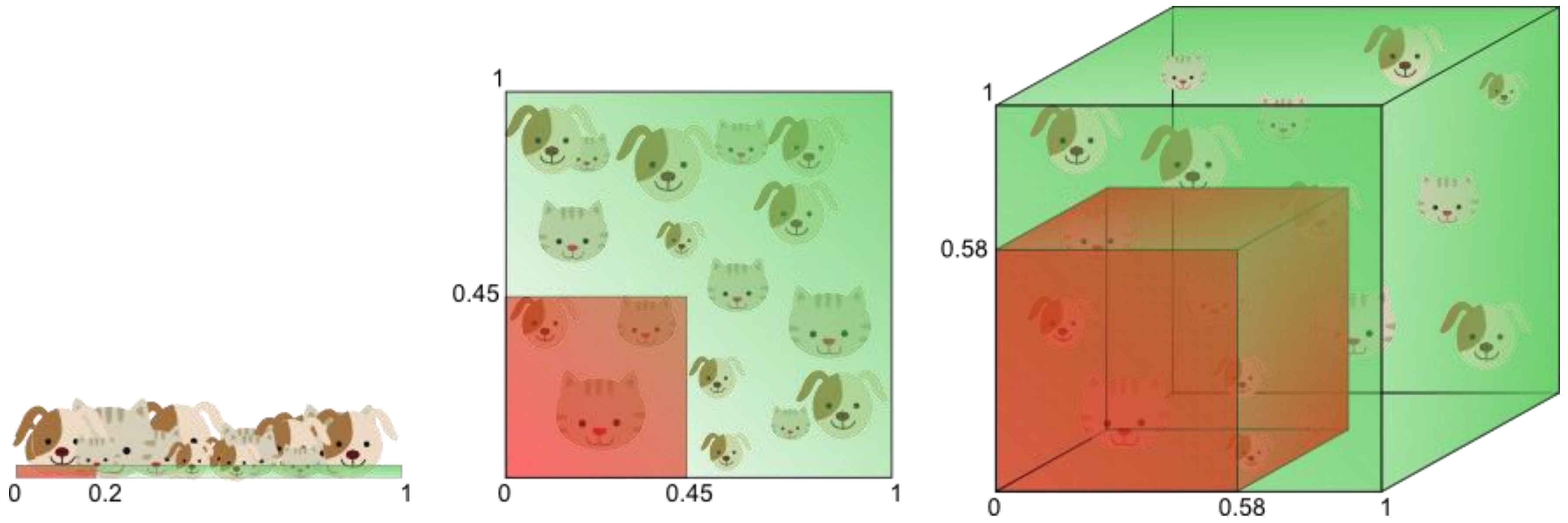
Electrical Engineering Department - Sharif University of Technology

Instructors: *Alireza Gargoori Motlagh – Amir Mirrashid – Ali Nourian*

DIMENSIONALITY REDUCTION

Motivation

- High dimensional data problems:
  - High computation cost and runtime of training
  - Increase the chance of overfitting
  - Correlated variables
  - Curse of dimensionality!



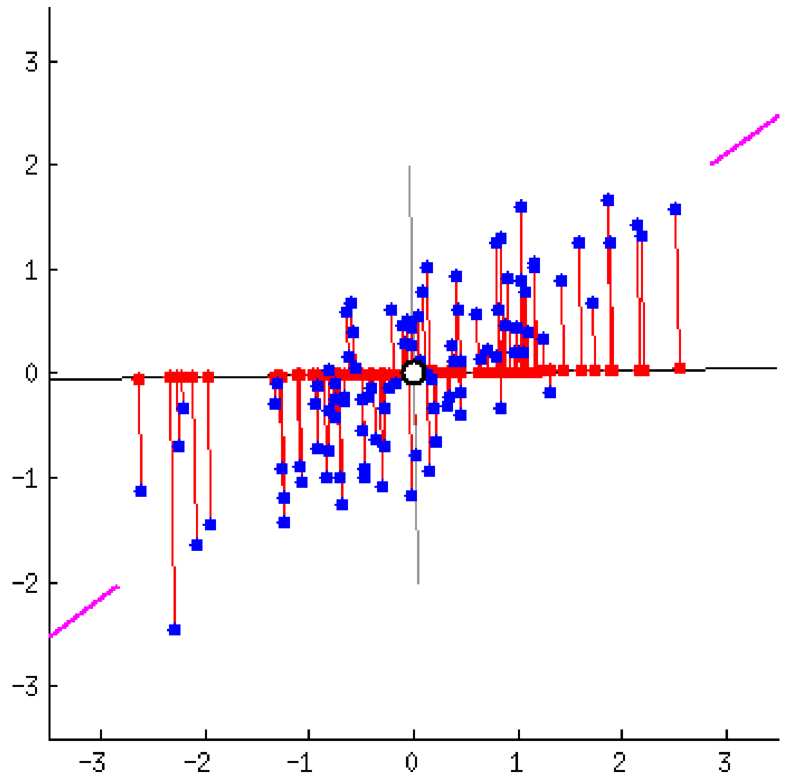
PCA

(PRINCIPAL COMPONENT ANALYSIS)

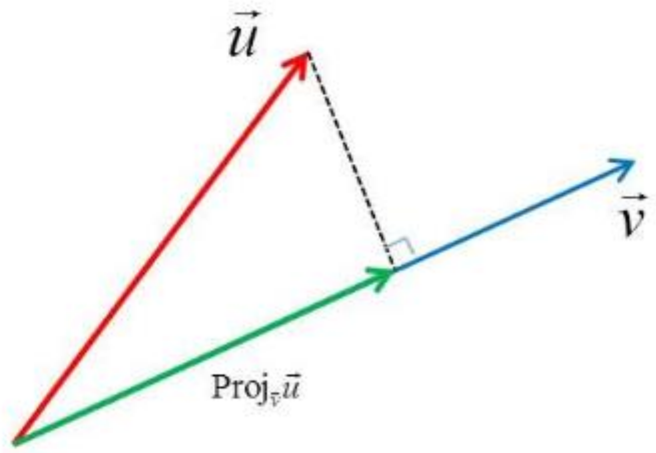
Principal Components

- Goal: Trying to find the directions with the most variance (after projection).

- The direction with the most variance is called *first principal component (PC1)*, the direction with the second highest variance and *orthogonal* to PC1 is called *second principal component (PC2)* and so on.



Projection Along a Direction



$$proj_v\boldsymbol{u} = \left(\frac{\boldsymbol{u}.\boldsymbol{v}}{||\boldsymbol{v}||^2}\right)\boldsymbol{v}$$

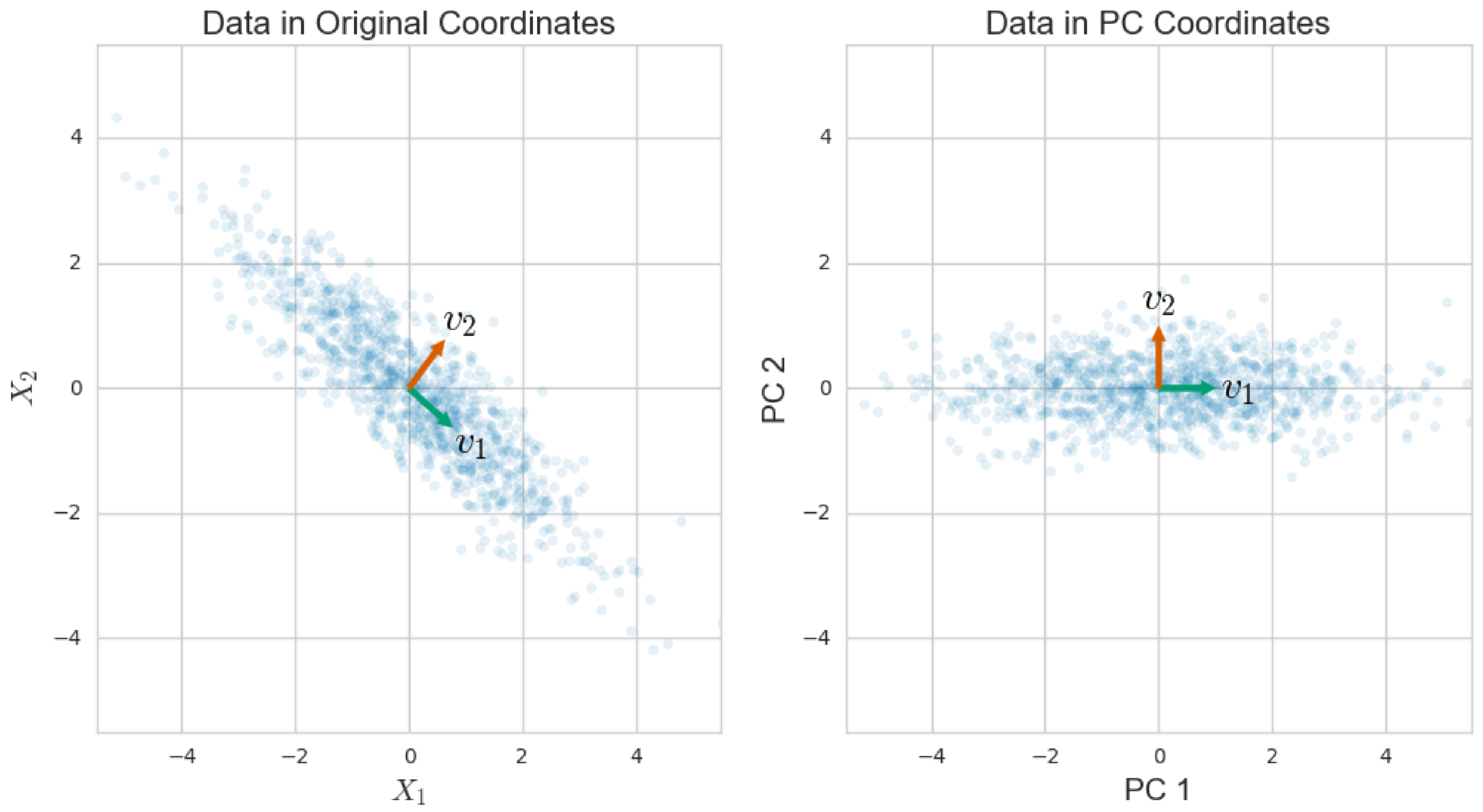
Since we care about the direction itself, we can consider norm of v to be one:

$$||\boldsymbol{v}||_{\mathbf{2}} = \boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{v} = 1$$

→ So the principal component score is:

$$PC\ score = \langle \boldsymbol{u}\ , \boldsymbol{v}\rangle = \boldsymbol{u}^{\boldsymbol{T}}\boldsymbol{v}$$

PC Transform



How to find PC1, PC2, … ?

The projected samples are: $\boldsymbol{v^T x^{(i)}}$ with the mean: $\boldsymbol{v^T \overline{x}}$

Considering all the data matrix $\boldsymbol{X}$:

$$var(\boldsymbol{Xv}) = cov(\boldsymbol{Xv}, \boldsymbol{Xv}) = \mathbb{E}\left[\left(\boldsymbol{Xv} - \mathbb{E}(\boldsymbol{Xv})\right)\left(\boldsymbol{Xv} - \mathbb{E}(\boldsymbol{Xv})\right)^T\right] =$$
$$\boldsymbol{v^T}\mathbb{E}\left[\left(\boldsymbol{X} - \mathbb{E}(\boldsymbol{X})\right)\left(\boldsymbol{X} - \mathbb{E}(\boldsymbol{X})\right)^T\right)\boldsymbol{v} = \boldsymbol{v^T C v}$$

where $\boldsymbol{C} = cov(\boldsymbol{X})$ is a symmetric matrix as follows:

$$\boldsymbol{C} = \begin{bmatrix} var(\boldsymbol{X_1}) & \cdots & cov(\boldsymbol{X_1}, \boldsymbol{X_p}) \\ \vdots & \ddots & \vdots \\ cov(\boldsymbol{X_p}, \boldsymbol{X_1}) & \cdots & var(\boldsymbol{X_p}) \end{bmatrix}$$

So our problem would be:

$$\max_{\boldsymbol{v}} \quad \boldsymbol{v^T C v}$$
$$s.t. \; \left||\boldsymbol{v}|\right|_2 = 1$$

Finding PCs

$$\max_{v} \quad \boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{C}\boldsymbol{v}$$
$$s.t. \left|\left|\boldsymbol{v}\right|\right|_2 = 1$$

using lagrange multipliers:

$$\mathcal{L}(\boldsymbol{v}, \lambda) = \boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{C}\boldsymbol{v} + \lambda(1 - \boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{v})$$

$$\nabla\mathcal{L}_{\boldsymbol{v}} = \frac{\partial\mathcal{L}(\boldsymbol{v}, \lambda)}{\partial\boldsymbol{v}} = 0 \ \rightarrow 2\boldsymbol{C}\boldsymbol{v} - 2\lambda\boldsymbol{v} = 0 \rightarrow \boldsymbol{C}\boldsymbol{v} = \lambda\boldsymbol{v}$$

Since $\boldsymbol{C}\boldsymbol{v} = \boldsymbol{\lambda}\boldsymbol{v}$, v is the unit eigenvector of covariance matrix with the eigenvalue λ.

$$\rightarrow var(\boldsymbol{X}\boldsymbol{v}) = \boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{C}\boldsymbol{v} = \boldsymbol{v}^{\boldsymbol{T}}\lambda\boldsymbol{v} = \lambda\boldsymbol{v}^{\boldsymbol{T}}\boldsymbol{v} = \lambda$$

$\Rightarrow$ the first k-components would be the first k *unit eigenvectors* corresponding to the biggest k *eigenvalues* of the covariance matrix of $\mathbf{X}$.

Proportion of Variance Explained (PVE)

Explained variance can be represented as a **function of ratio of related eigenvalue and sum of eigenvalues of all eigenvectors.**

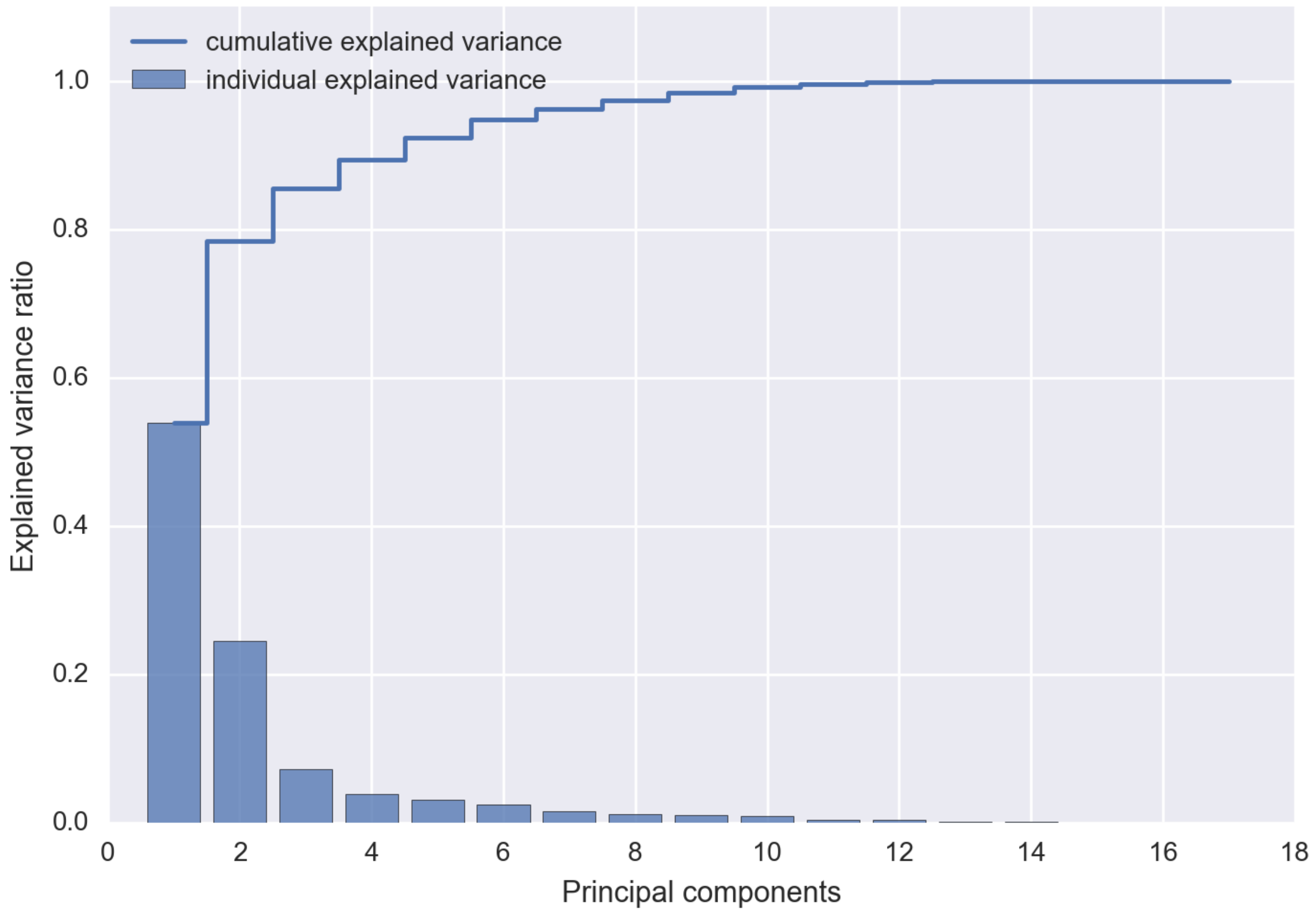
The total variance of X can be explained using all the eigenvectors of covariance matrix:

$$tot_{var}(\boldsymbol{X}) = tr\big(cov(\boldsymbol{X})\big) = \sum_{i=1}^{p} \lambda_i$$

So the explained variance of the k-th eigenvector is:

$$PVE(k) = \frac{\lambda_k}{\sum_{i=1}^{p} \lambda_i}$$

Choosing the new dimension using PVE



Transform into PC coordinates

- $\boldsymbol{C}$ with eigenvalue decomposition:

$$\boldsymbol{C} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{\boldsymbol{T}} = [\boldsymbol{v_1} \quad \boldsymbol{v_2} \quad \cdots \quad \boldsymbol{v}_{\boldsymbol{p}}] \begin{bmatrix} \lambda_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \lambda_p \end{bmatrix} \begin{bmatrix} \boldsymbol{v_1^T} \\ \boldsymbol{v_2^T} \\ \vdots \\ \boldsymbol{v_p^T} \end{bmatrix}$$

- After choosing the k (with PVE or any other method), the rotation matrix $\boldsymbol{W}$ with the first k eigenvectors would be:

$$\boldsymbol{W} = [\boldsymbol{v_1} \quad \boldsymbol{v_2} \quad \cdots \quad \boldsymbol{v_k}]$$

- Samples are transformed from the original feature space into new PC coordinates as:

$$\boldsymbol{x}_{\boldsymbol{new}}^{(\boldsymbol{i})} = \boldsymbol{y}^{(i)} = \boldsymbol{W}^T \boldsymbol{x}^{(i)}$$

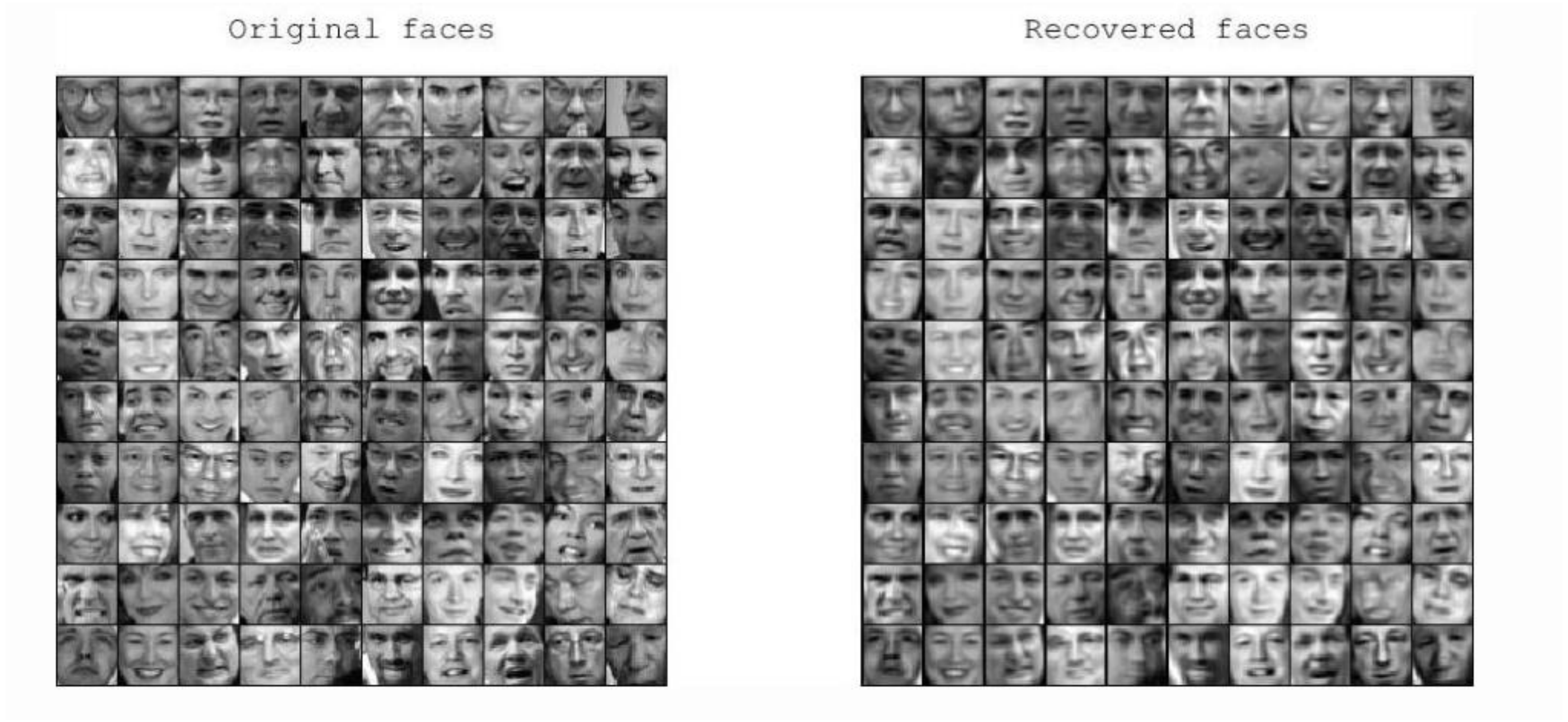
- Considering all the samples, we can write:

$$\boldsymbol{X}_{\boldsymbol{new}} = \boldsymbol{W}^{\boldsymbol{T}} \boldsymbol{X}$$

Reconstruction using PCA

$$PCA\ reconstuction = PC\ scores \times eigenvectors^T + mean$$

$$\hat{\boldsymbol{x}}^{(\boldsymbol{i})} = \sum_{j=1}^{k} y_j^{(i)} \boldsymbol{v_j} + \overline{\boldsymbol{x}}$$



KERNEL PCA

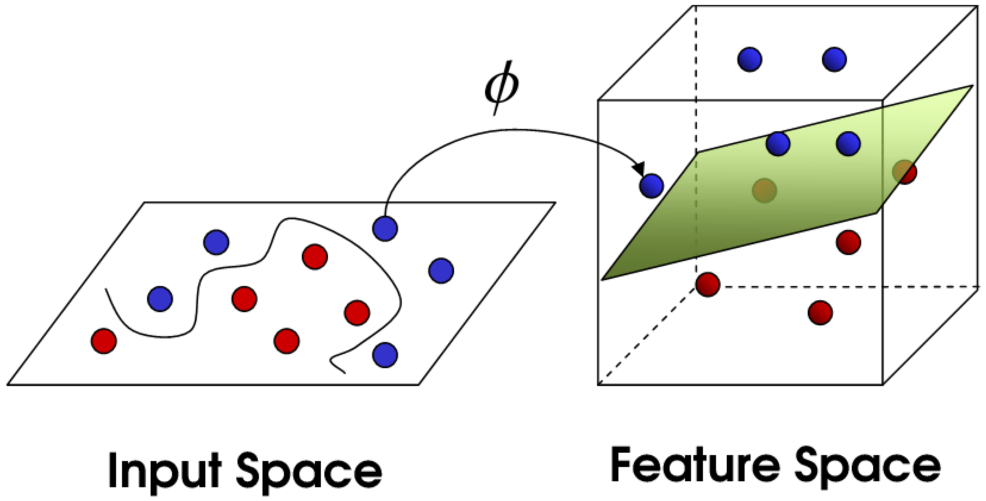
(K-PCA)

Making PCA non-Linear

- Suppose that instead of using the samples themselves, we wanted to go to some different feature space $\phi\left(x^{(i)}\right) \in \mathbb{R}^p$

- In the new feature space, then we can do PCA.
- This result will be non-linear in the original data space!

- Similar to SVM kernels idea!



Kernels & Inner Products

- Suppose the zero-mean samples, then covariance matrix would be:

$$\boldsymbol{C} = \boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}$$

- Also, the inner product matrix is:

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{x}^{(\mathbf{1})^{\boldsymbol{T}}}\boldsymbol{x}^{(\mathbf{1})} & \cdots & \boldsymbol{x}^{(\mathbf{1})^{\boldsymbol{T}}}\boldsymbol{x}^{(\boldsymbol{N})} \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}^{(\boldsymbol{N})^{\boldsymbol{T}}}\boldsymbol{x}^{(\mathbf{1})} & \cdots & \boldsymbol{x}^{(\boldsymbol{N})^{\boldsymbol{T}}}\boldsymbol{x}^{(\boldsymbol{N})} \end{bmatrix} = \boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}$$

where $K_{i,j}$ is the inner product of i^{th} & j^{th} samples.

This can be generalized into other kernels, such as RBF kernel as follows:

$$\boldsymbol{K} = \begin{bmatrix} \langle \phi(x^{(1)}), \phi(x^{(1)}) \rangle & \cdots & \langle \phi(x^{(1)}), \phi(x^{(N)}) \rangle \\ \vdots & \ddots & \vdots \\ \langle \phi(x^{(N)}), \phi(x^{(1)}) \rangle & \cdots & \langle \phi(x^{(N)}), \phi(x^{(N)}) \rangle \end{bmatrix}$$

where $K_{i,j} = K\left(x^{(i)}, x^{(j)}\right)$

Covariance Matrix & Inner Products

- Suppose $\boldsymbol{u}$ is an eigenvector of $\boldsymbol{K}$:

$$\boldsymbol{K}\boldsymbol{u} = \mu\boldsymbol{u} \rightarrow \boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{u} = \mu\boldsymbol{u}$$

then:

$$\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{X}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{u} = \mu\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{u} \rightarrow \boldsymbol{C}\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{u} = \mu\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{u}$$

- We only need inner products!

To find the PC coefficients for a sample x:

$$y = \boldsymbol{x}^{\boldsymbol{T}}\left(\boldsymbol{X}^{\boldsymbol{T}}\boldsymbol{u}\right) = (\boldsymbol{X}\boldsymbol{x})^{\boldsymbol{T}}\boldsymbol{u} = [\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{x}^{(\mathbf{1})} \quad \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{x}^{(\mathbf{2})} \quad \ldots \quad \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{x}^{(\boldsymbol{N})}]\boldsymbol{u}$$

- If K-PCA, we always have to use normalized kernel matrix!

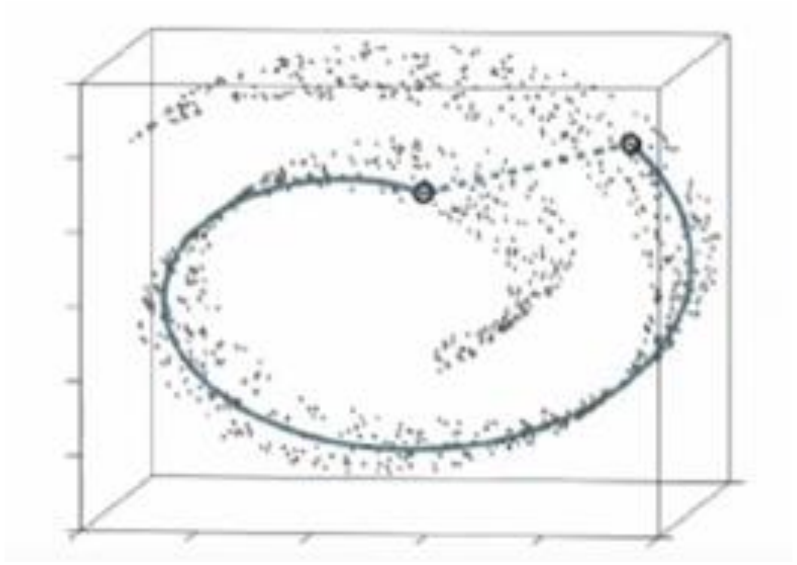
T-SNE

(T-DISTRIBUTED STOCHASTIC NEIGHBOR EMBEDDING)

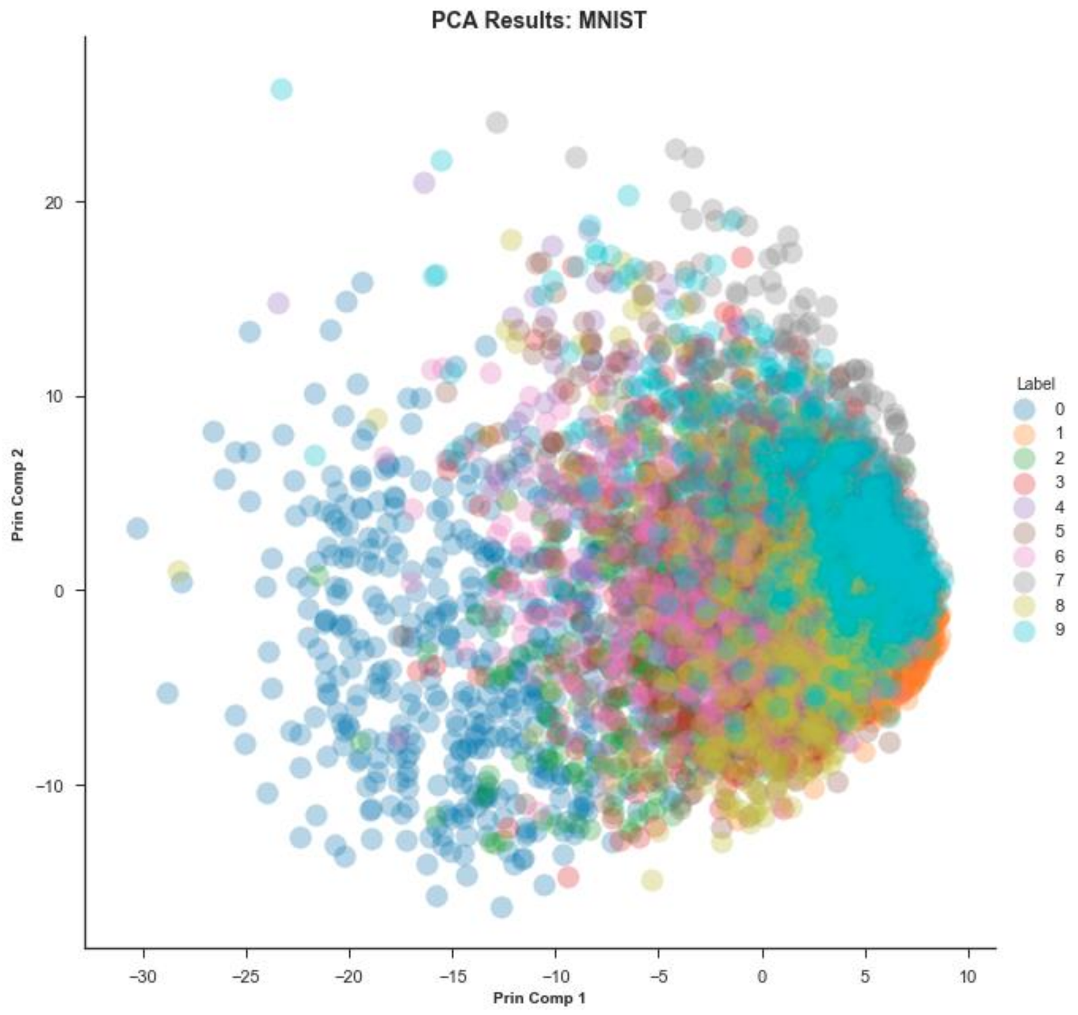
t-SNE: Motivation

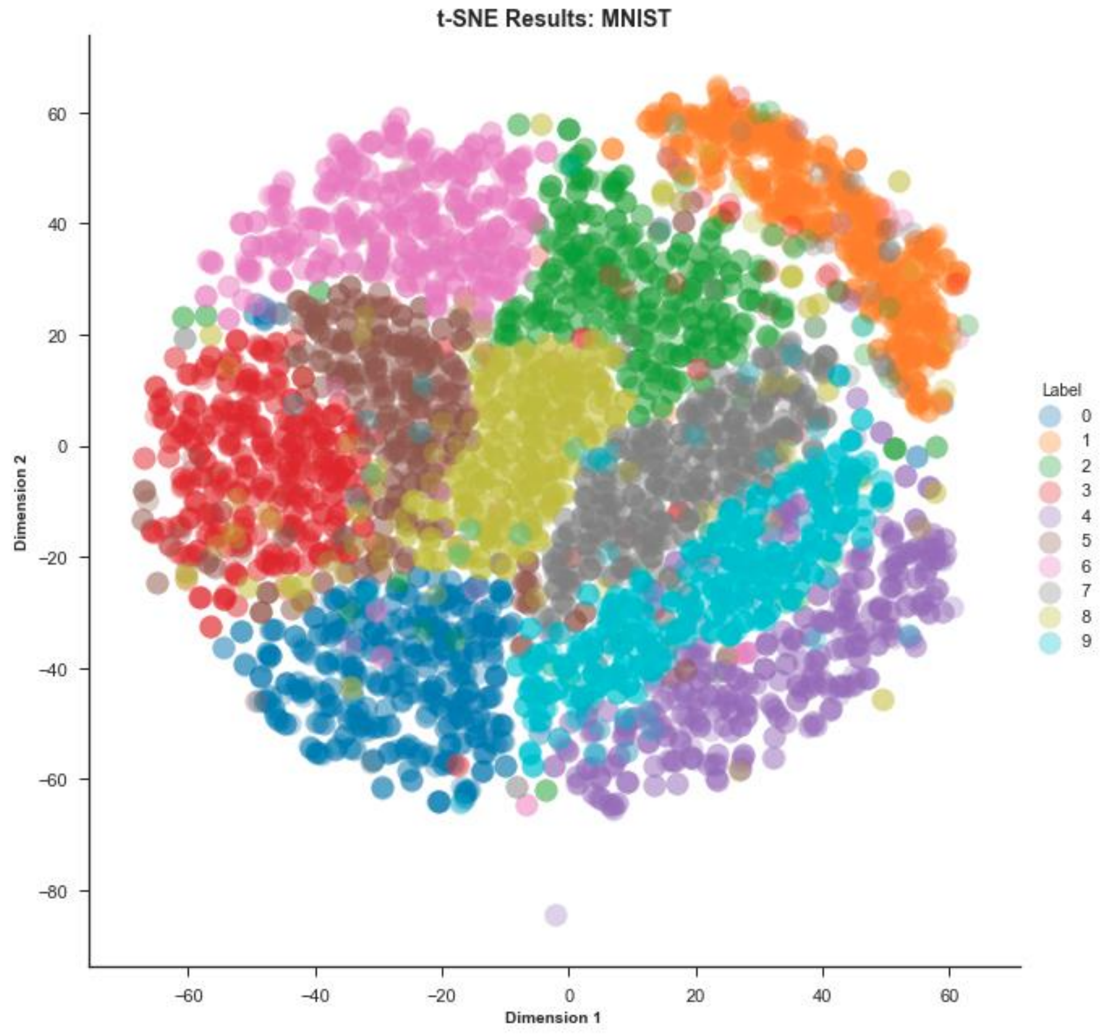
t-Distributed Stochastic Neighbor Embedding (t-SNE) *(L.J.P. van der Maaten and G.E. Hinton, 2008)* is a technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.

The t-SNE algorithm calculates a similarity measure between pairs of instances in the high dimensional space and in the low dimensional space. It then tries to optimize these two similarity measures using a cost function (KL divergence).



PCA vs t-SNE


PCA Results: MNIST
Prin Comp 1
Prin Comp 2
Label
0
1
2
3
4
5
6
7
8
9


t-SNE Results: MNIST
Dimension 1
Dimension 2
Label
0
1
2
3
4
5
6
7
8
9