DIMENSIONALITY REDUCTION

NOITC DIMENSION

Hands-On Machine Learning with Scikit-Learn and TensorFlow

2022 Jan 17th
Mon 2pm

Kwangwoon University
Machine Learning Study STG Team C

Principal **C**omponent **A**nalysis

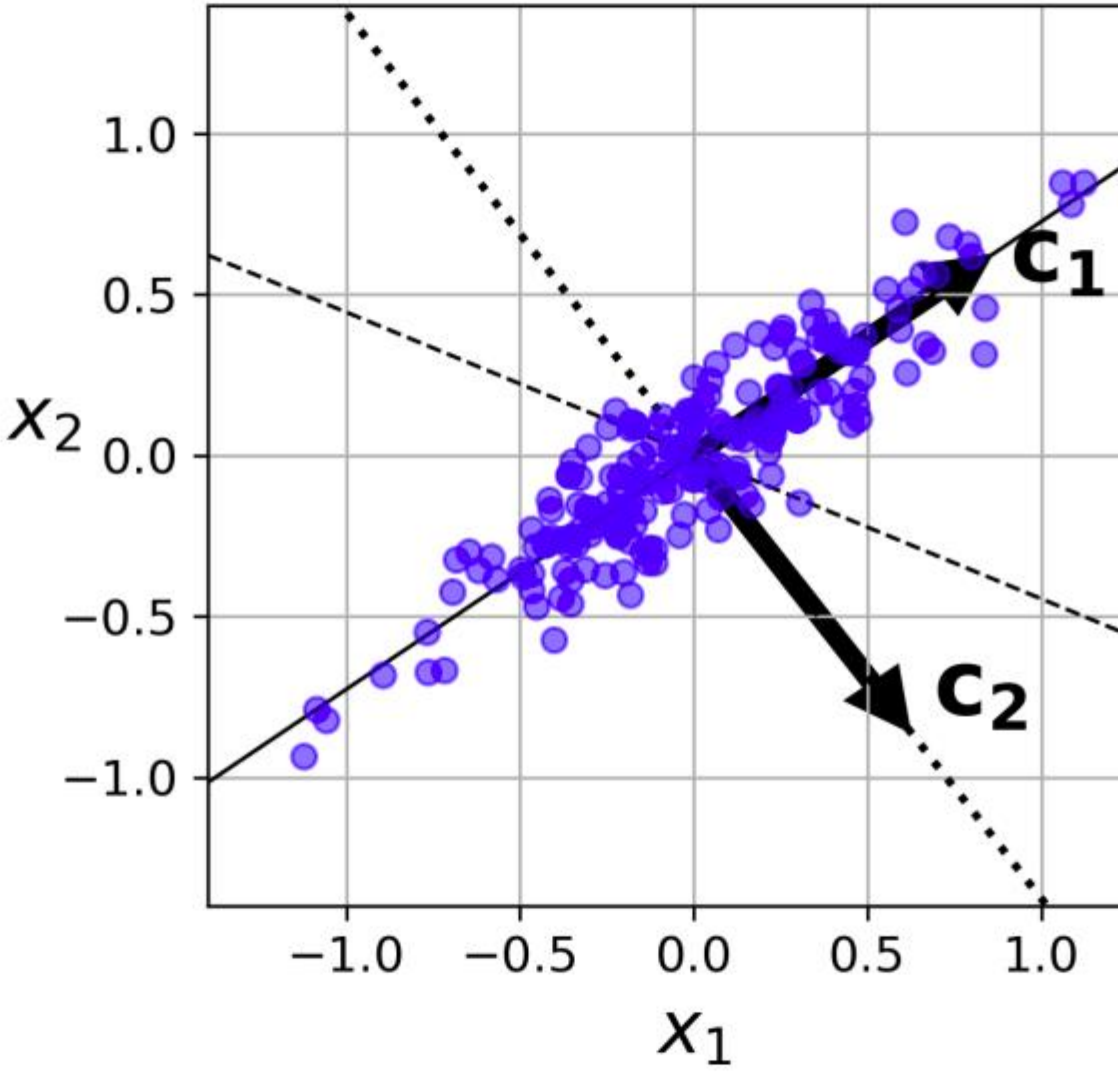
: The **most popular** dimensionality reduction algorithm

: **Linear transformation**

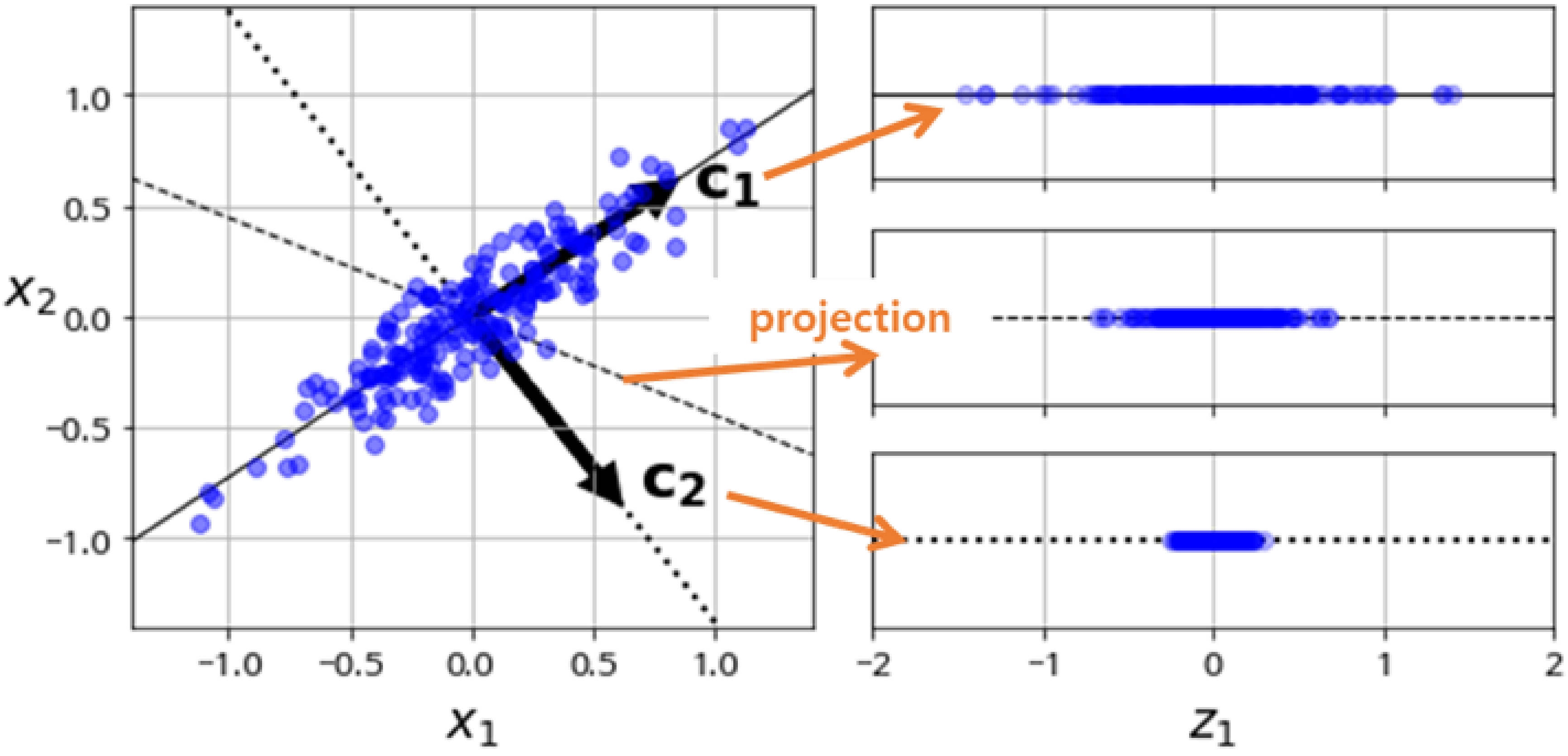
: The point is {**Variance**}

PCA: Preserving the Variance

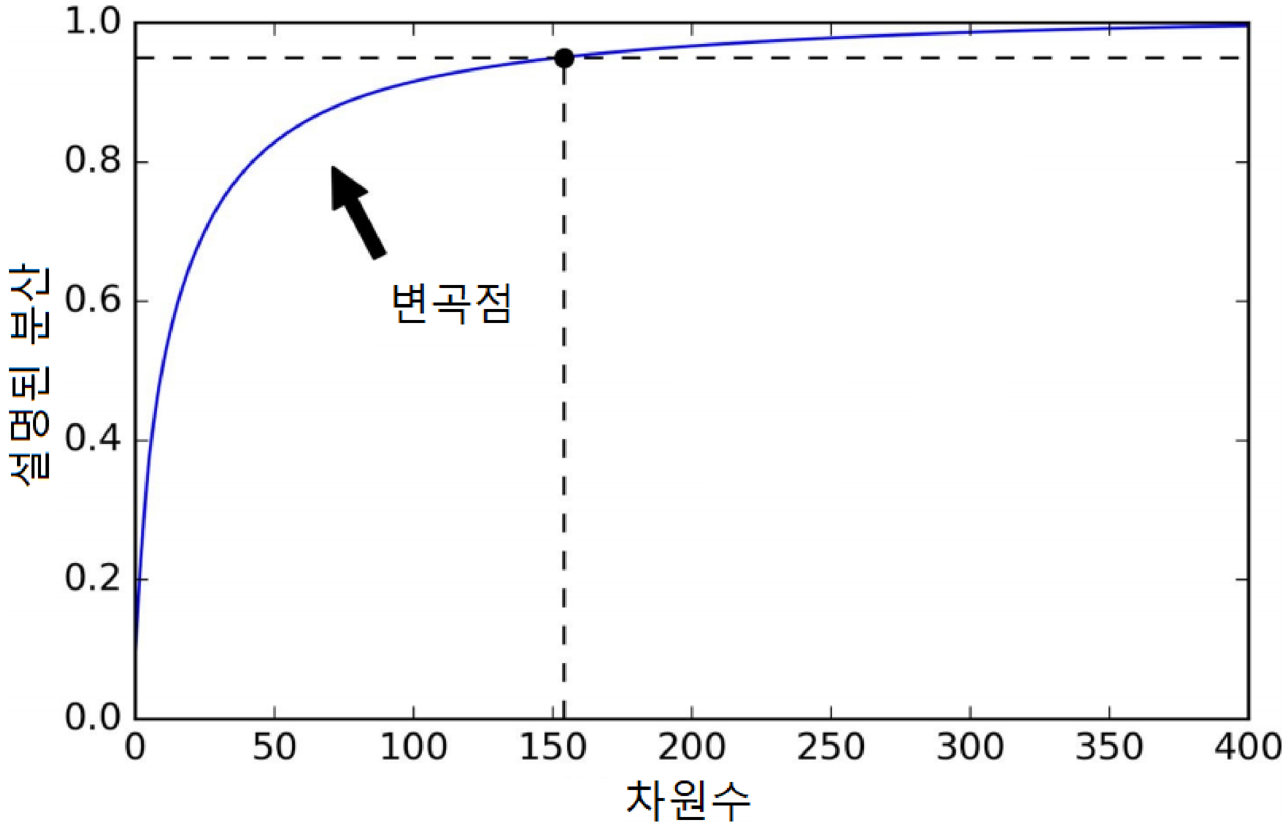
2D scatter plot



PCA: Preserving the Variance

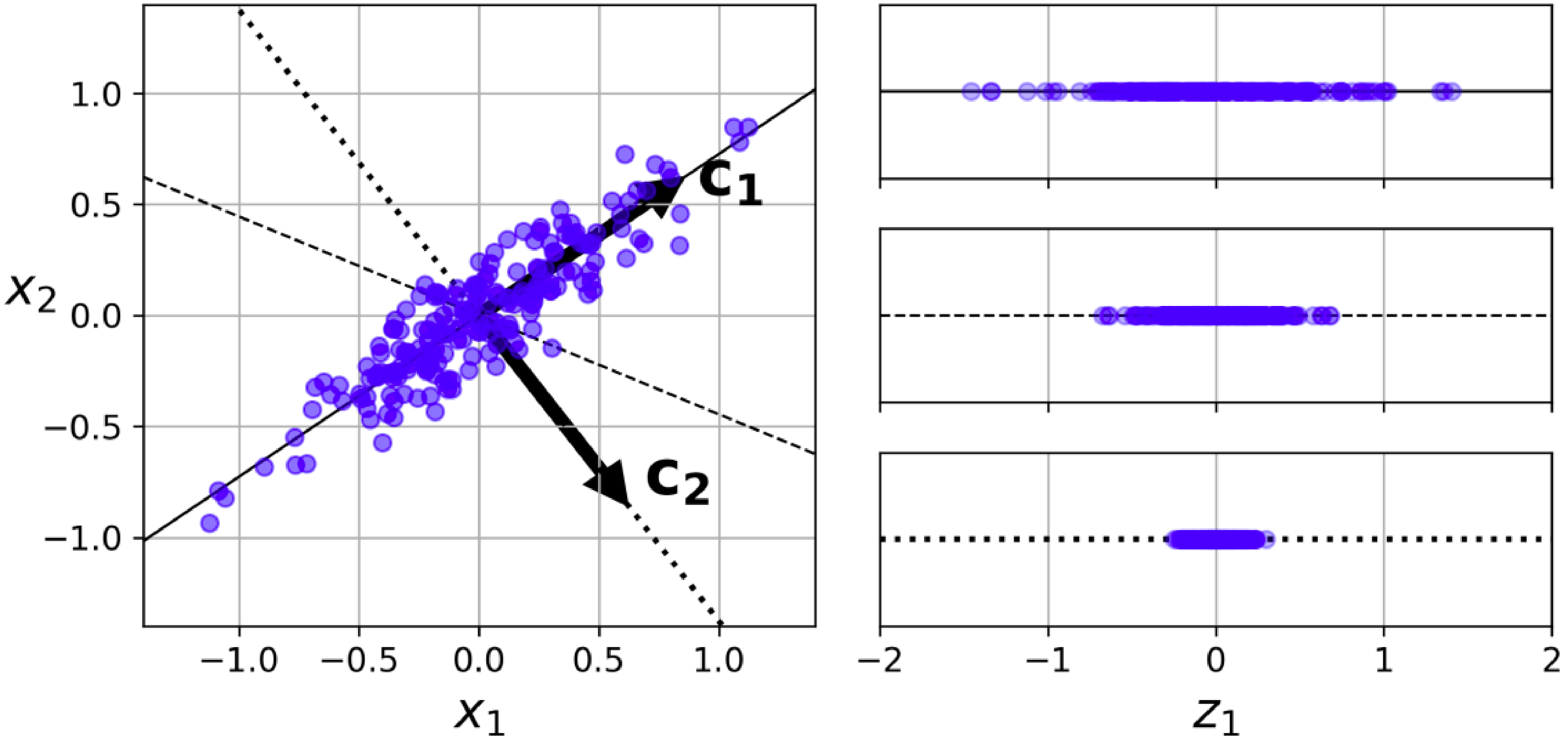


PCA: Preserving the Variance

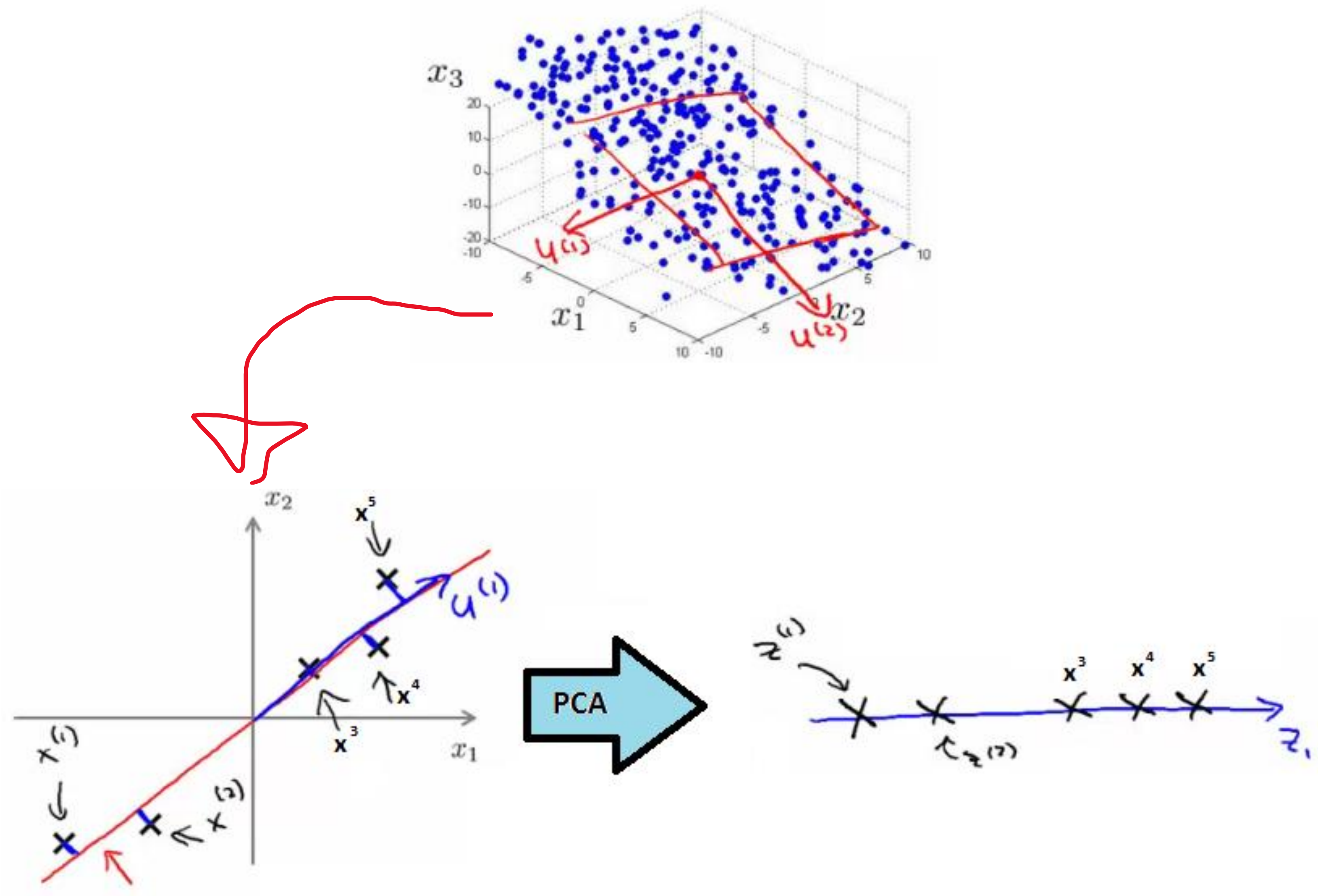


Accuracy / Cost	Low	High
Low	Meaningless	PROFIT!
High	Meaningless	Good

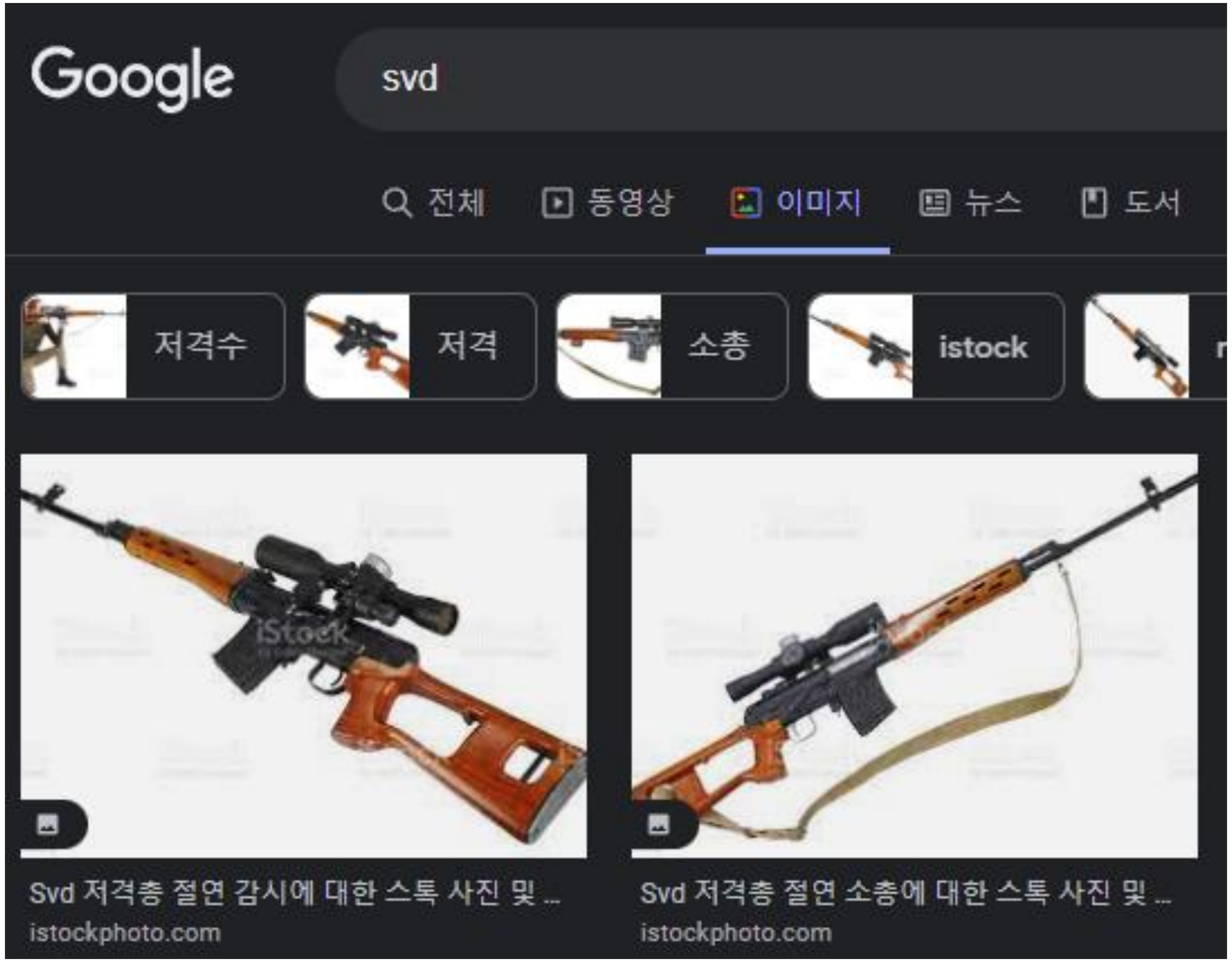
PCA: Preserving the Variance



PCA: Principal Component

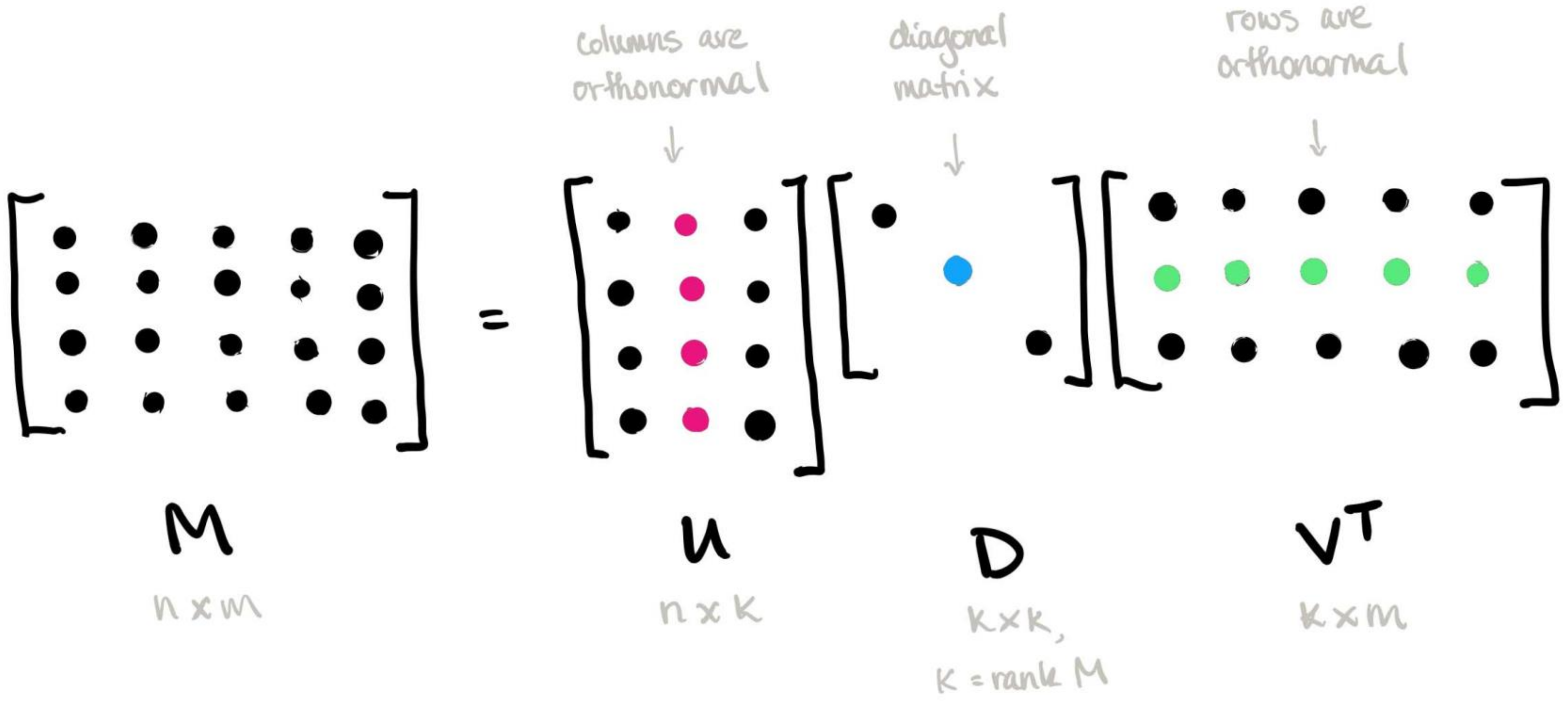


PCA: What is SVD?



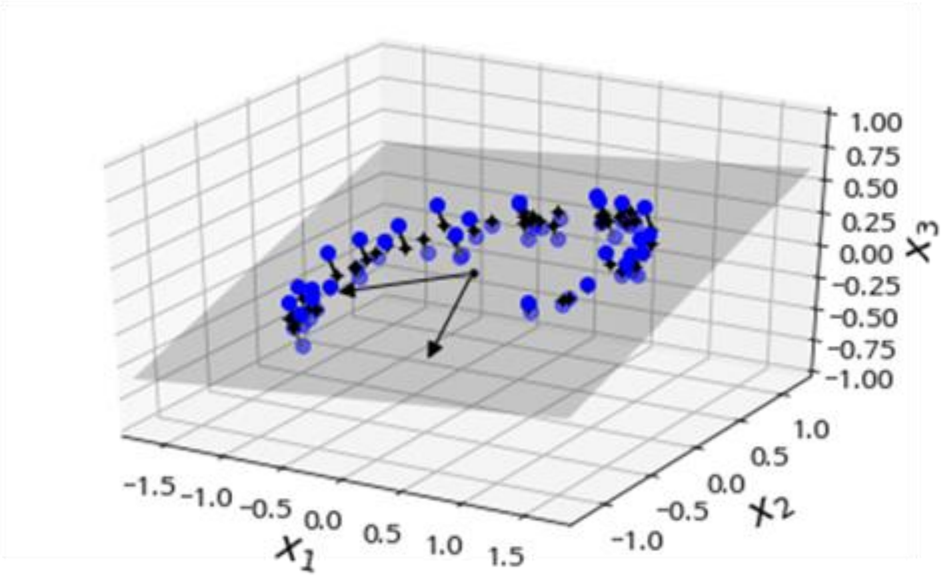
SVD?

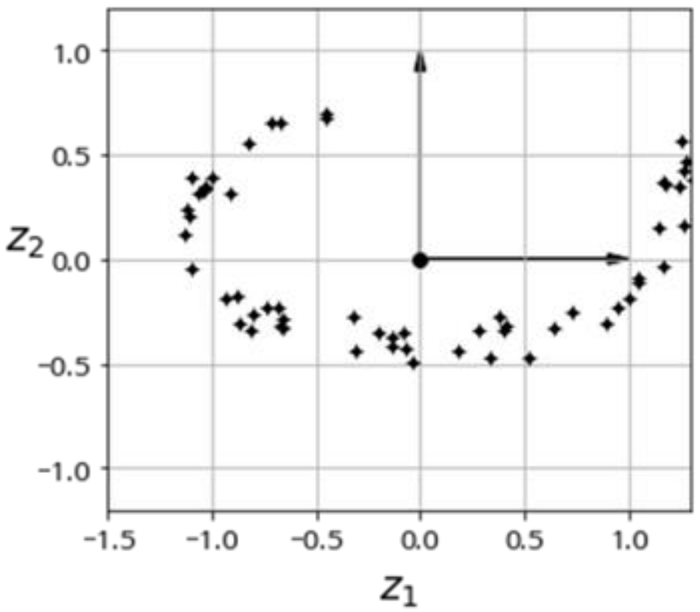
PCA: Singular Value Decomposition



S ingular
V alue
D ecomposition

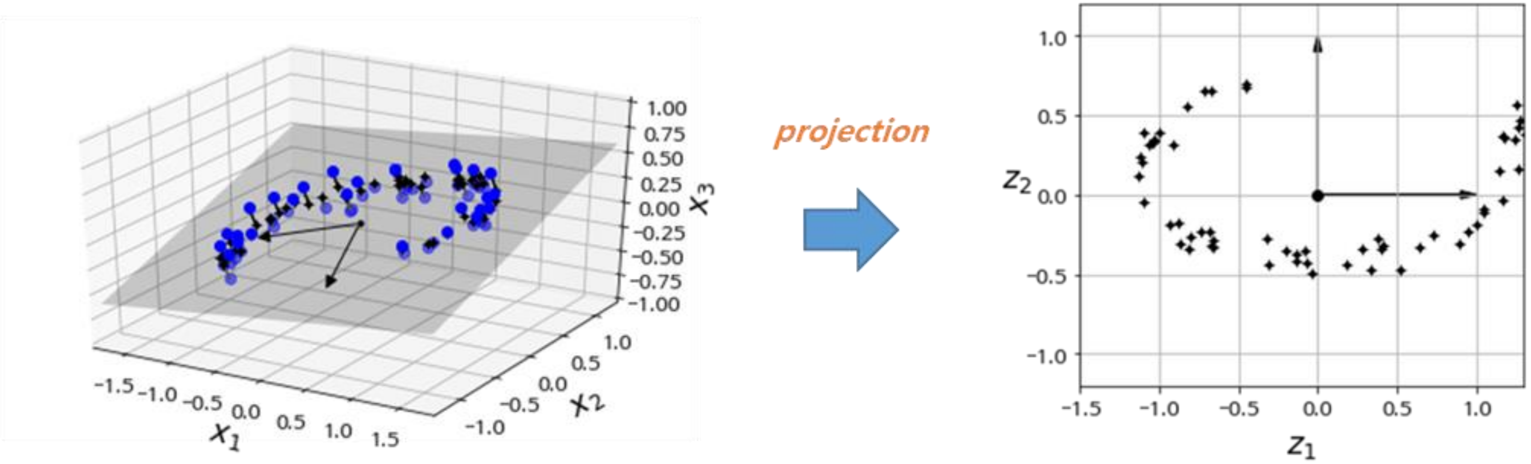
PCA: Projecting Down to *d* Dimensions





3D to **2D** Conversion

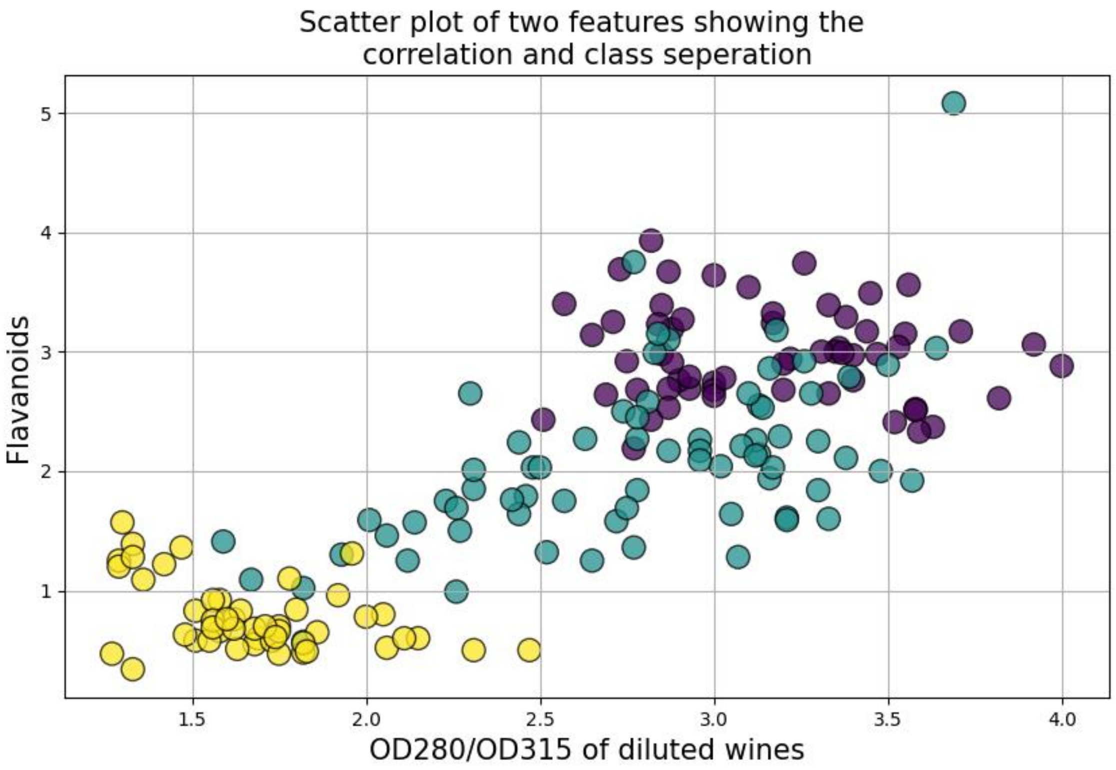
PCA: Projecting Down to *d* Dimensions

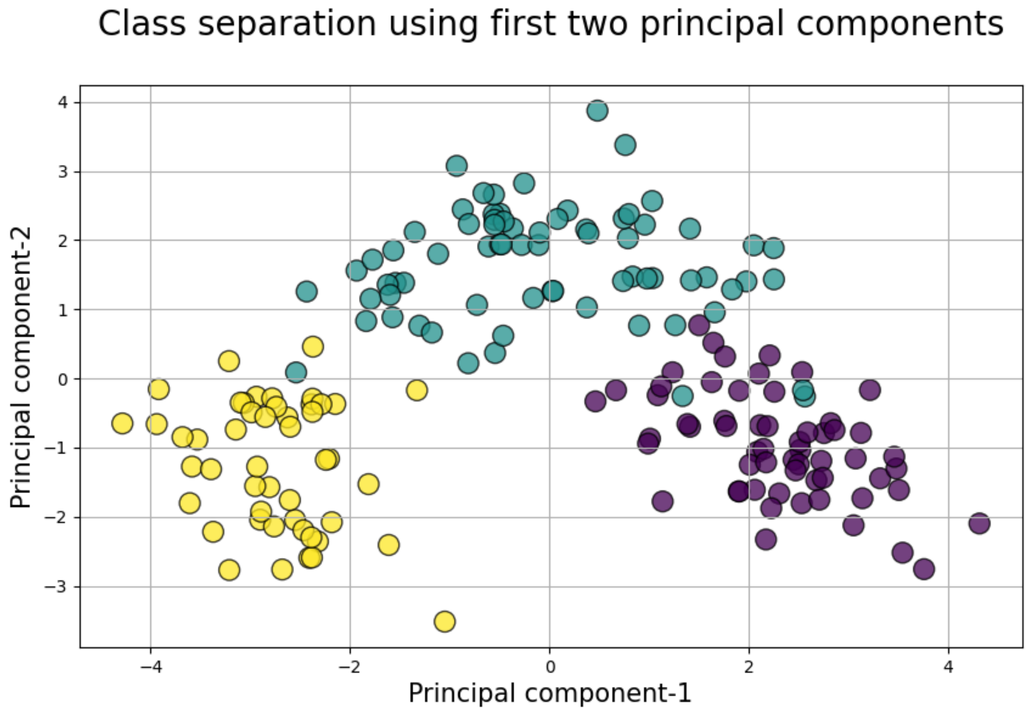


$$\mathbf{X}_{d\text{-proj}} = X \cdot W_d$$

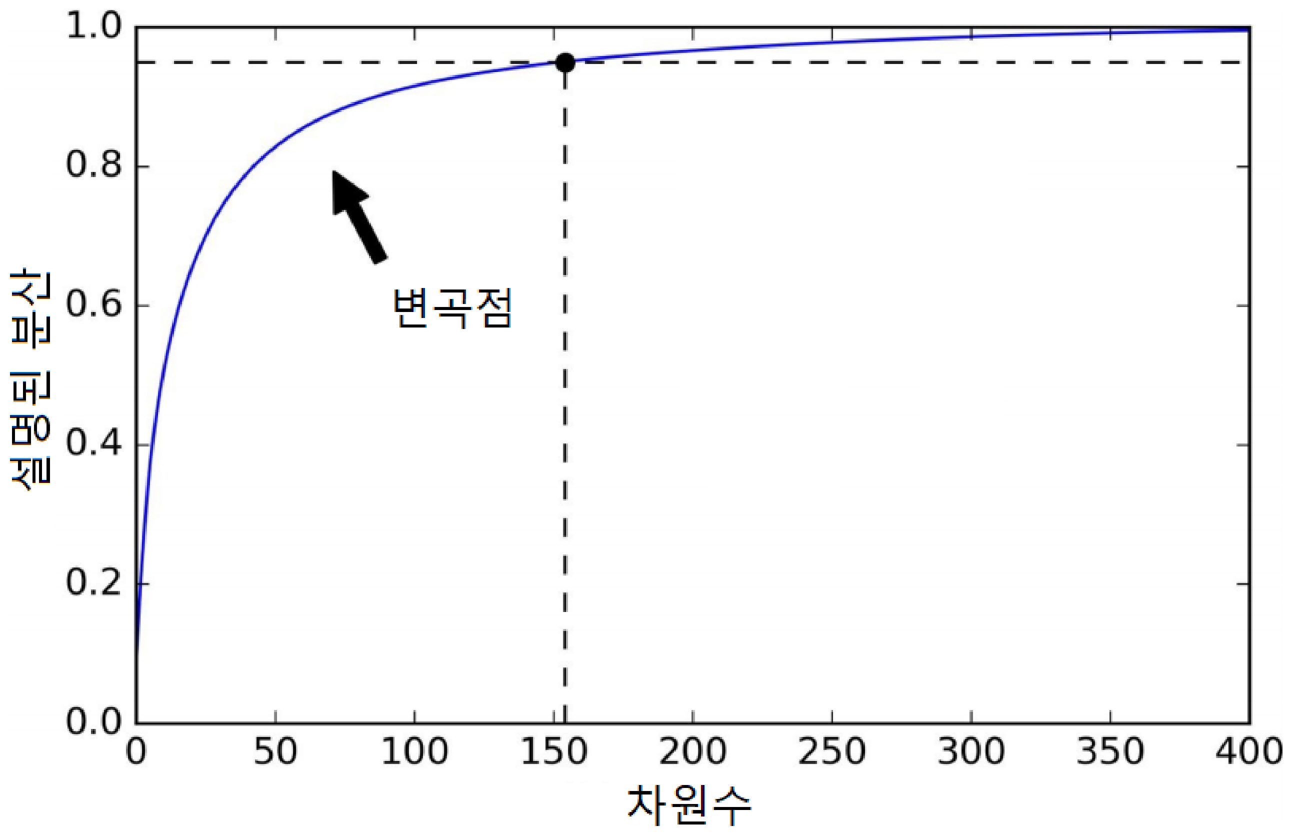
3D to **2D** Conversion

PCA: Projecting Down to d Dimensions

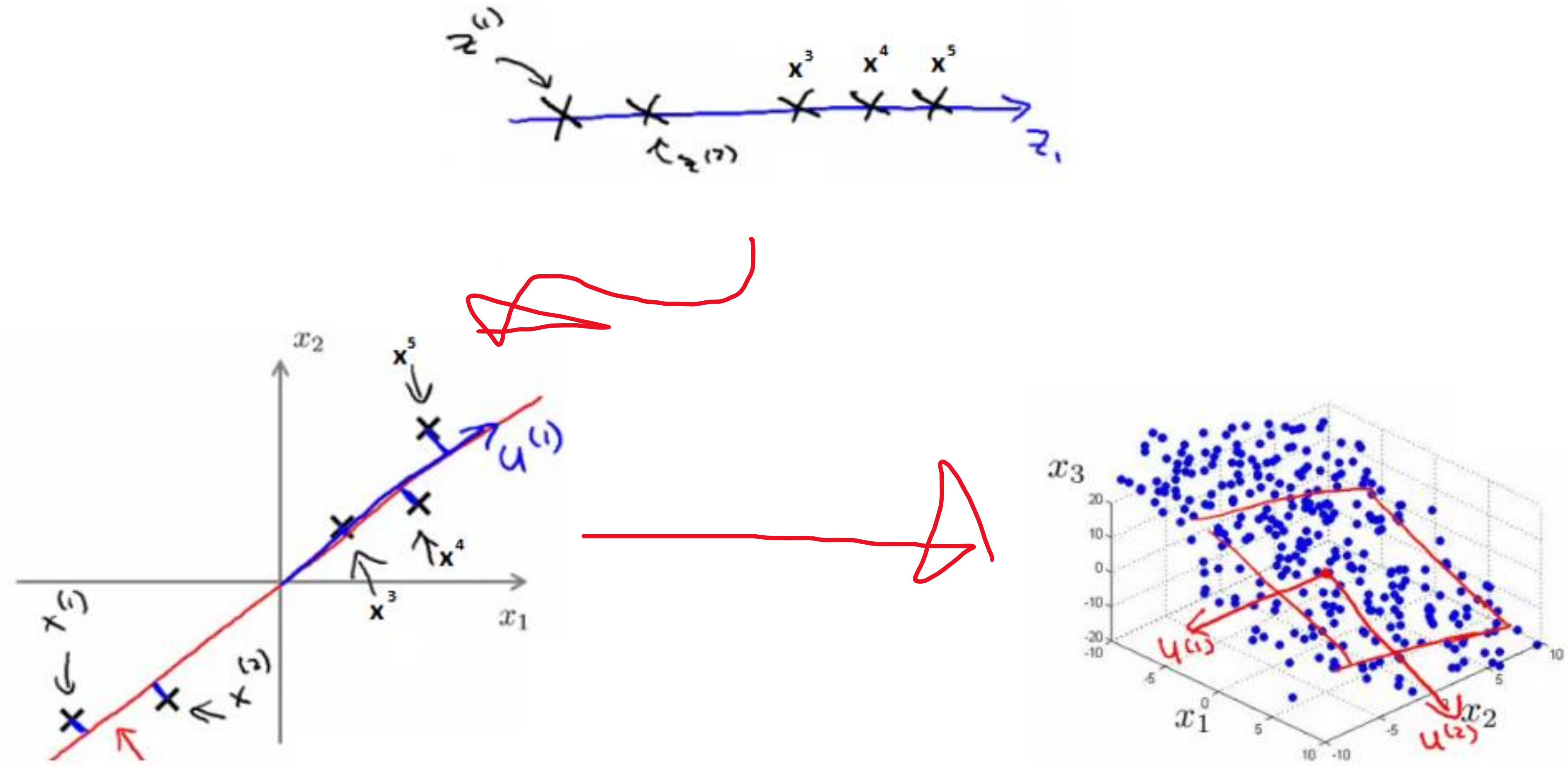




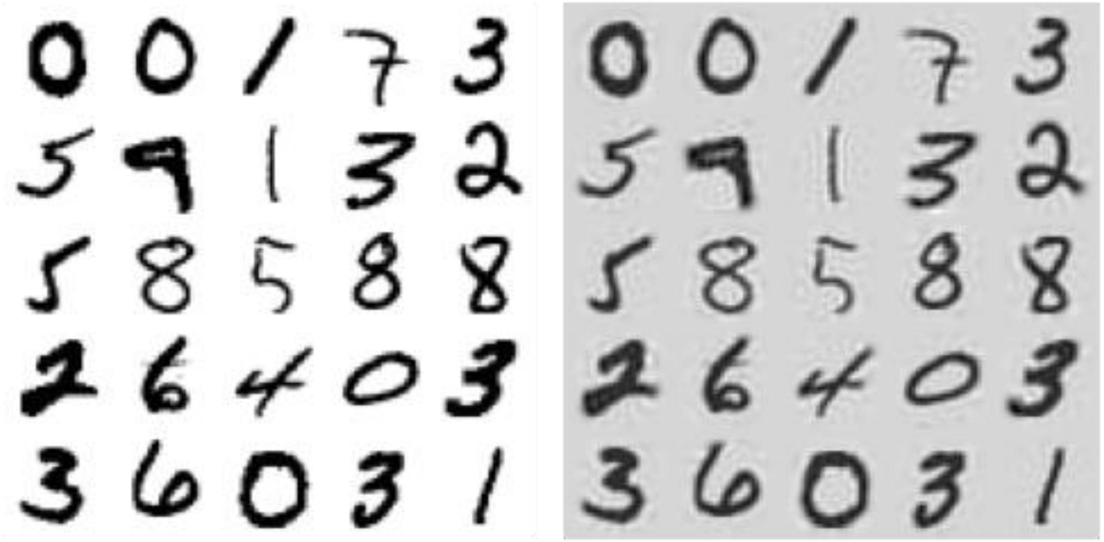
PCA: Compression



PCA: Compression

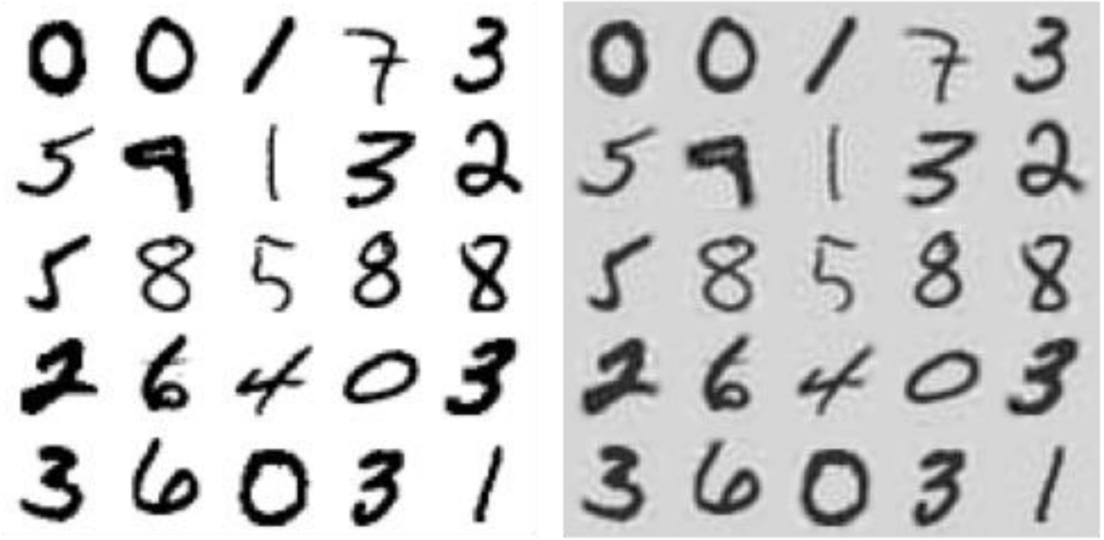


PCA: Compression



Recovered MNIST data
From **154-d** to **784-d**

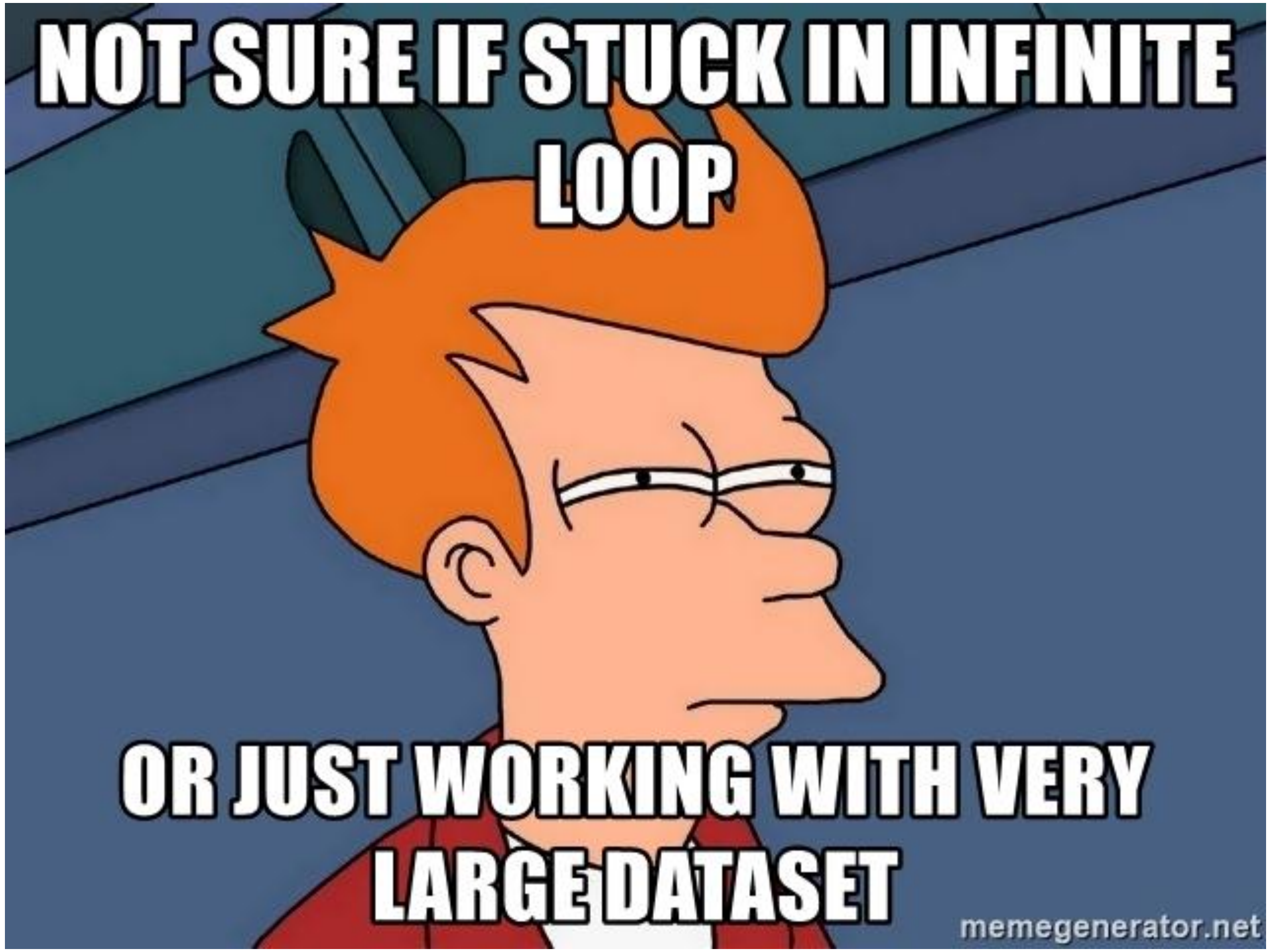
PCA: Compression



Recovered MNIST data
From **154-d** to **784-d**

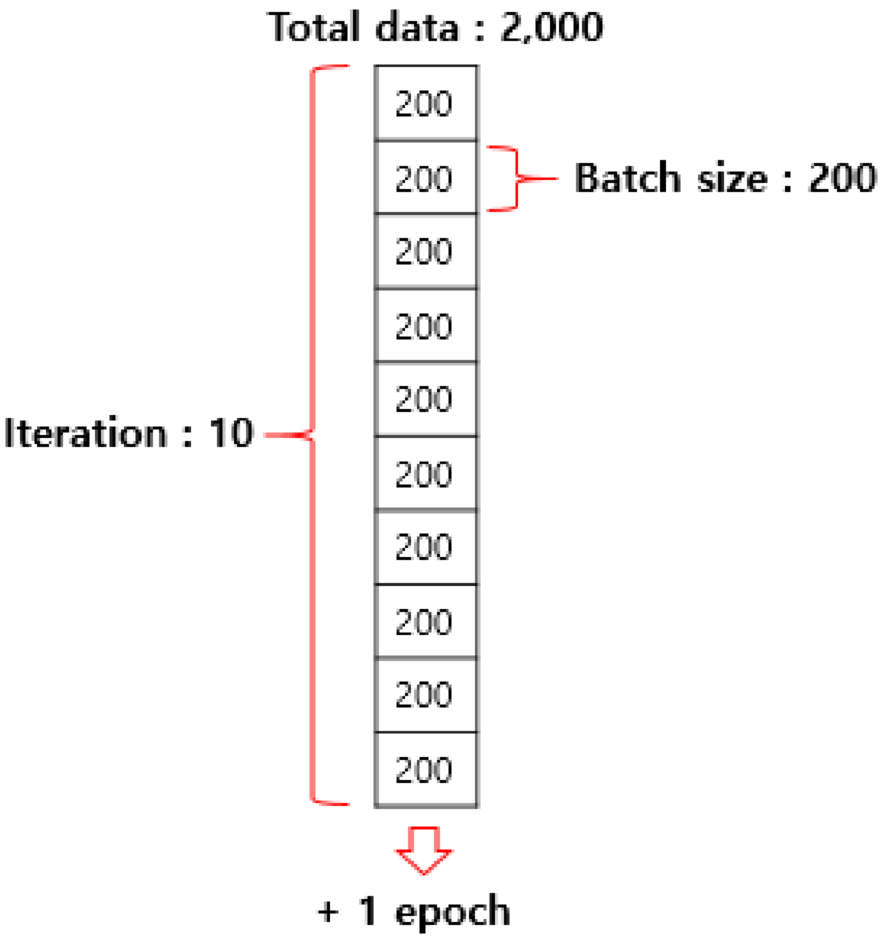
$$X_{recovered} = X_{d\text{-}proj} \cdot W^{T}_{d}$$

PCA: Incremental, Random and Kernel approach



Limited resources

PCA: Incremental, Random and Kernel approach



Mini batch method

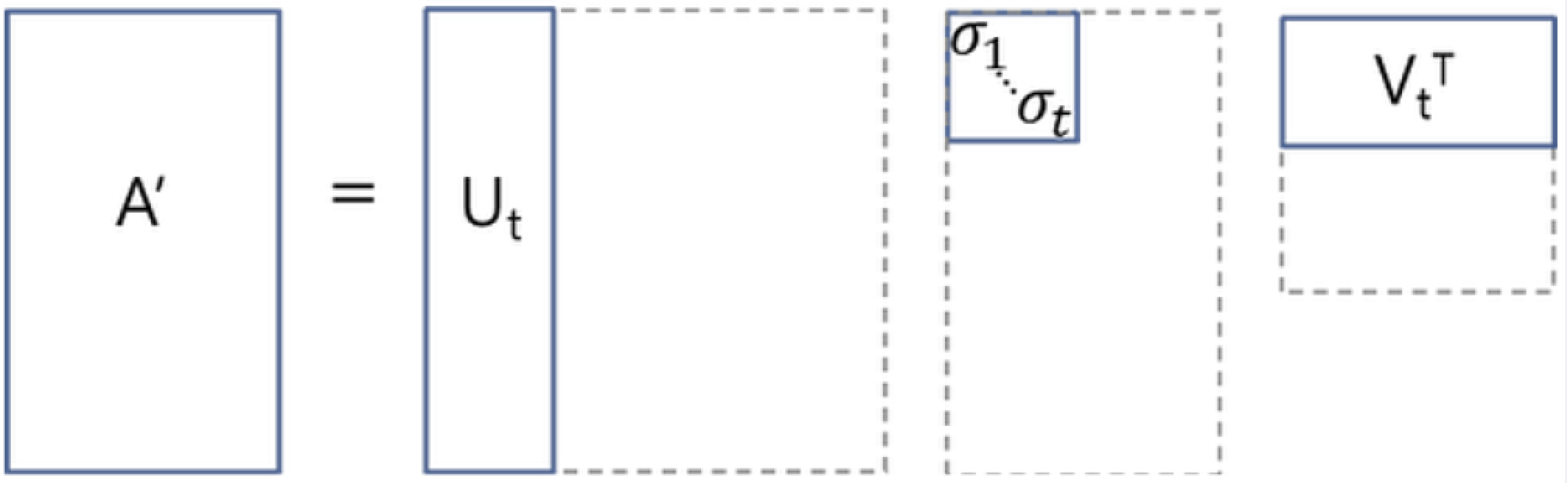
PCA: Incremental, Random and Kernel approach

```
def _fit_truncated(self, X, n_components, svd_solver):
    """Fit the model by computing truncated SVD (by ARPACK or randomized)
    on X.
    """

    n_samples, n_features = X.shape
```

Random PCA == Truncated SVD

PCA: Incremental, Random and Kernel approach

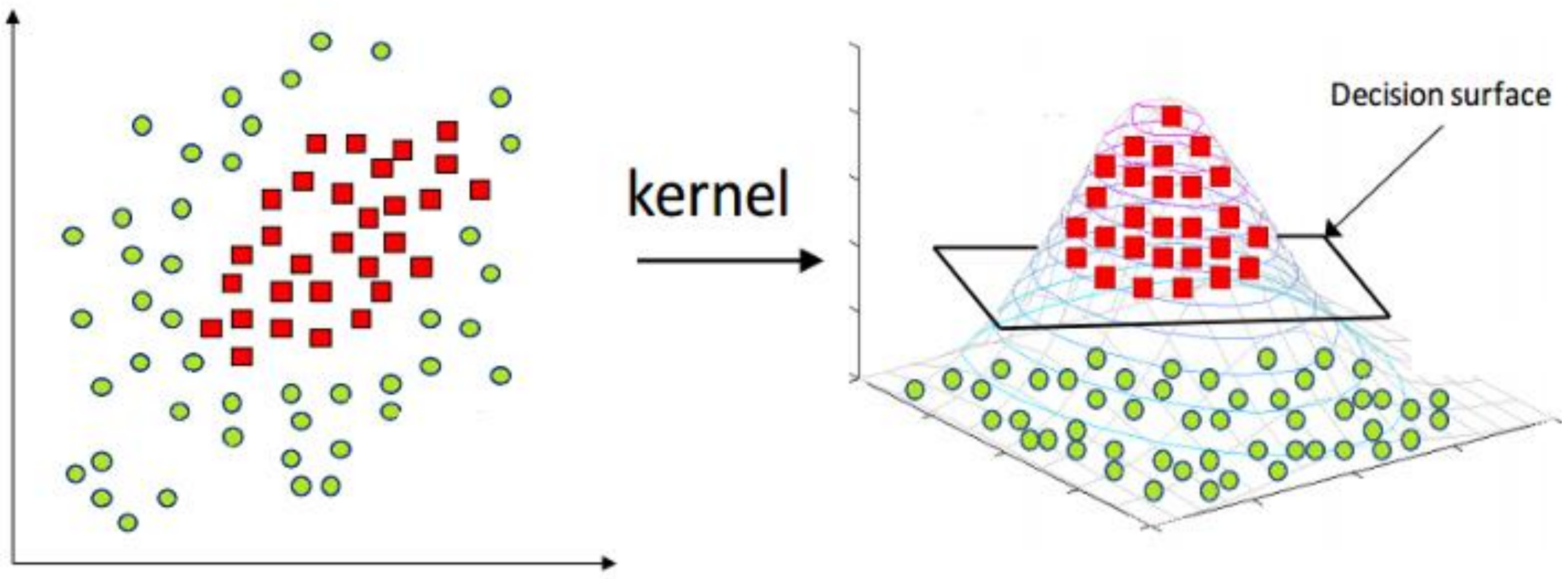


Time complexity comparison

PCA with full SVD: O(m * n^2) + O(n^3)

✓ PCA with Truncatted SVD: O(m * d^2) + O(d^3)

PCA: Incremental and Random approach



Welcome back, **Kernel Trick!**

PCA: Kernel approach

$$\mathbf{x} = (x_1, x_2, x_3)^T$$
$$\mathbf{y} = (y_1, y_2, y_3)^T$$

Assume that we need to map x and y to **9-d space**.

PCA: Kernel approach

$$\mathbf{x} = (x_1, x_2, x_3)^T$$
$$\mathbf{y} = (y_1, y_2, y_3)^T$$

Assume that we need to map x and y to **9-d space**.

PCA: Kernel approach

$$\phi(\mathbf{x}) = (x_1^2, x_1x_2, x_1x_3, x_2x_1, x_2^2, x_2x_3, x_3x_1, x_3x_2, x_3^2)^T$$
$$\phi(\mathbf{y}) = (y_1^2, y_1y_2, y_1y_3, y_2y_1, y_2^2, y_2y_3, y_3y_1, y_3y_2, y_3^2)^T$$

Time complexity: $O(n^2)$

PCA: Kernel approach

$$\phi(\mathbf{x})^T\phi(\mathbf{y}) = \sum_{i,j=1}^{3} x_i x_j y_i y_j$$

Time complexity: O(n)

PCA: Kernel approach

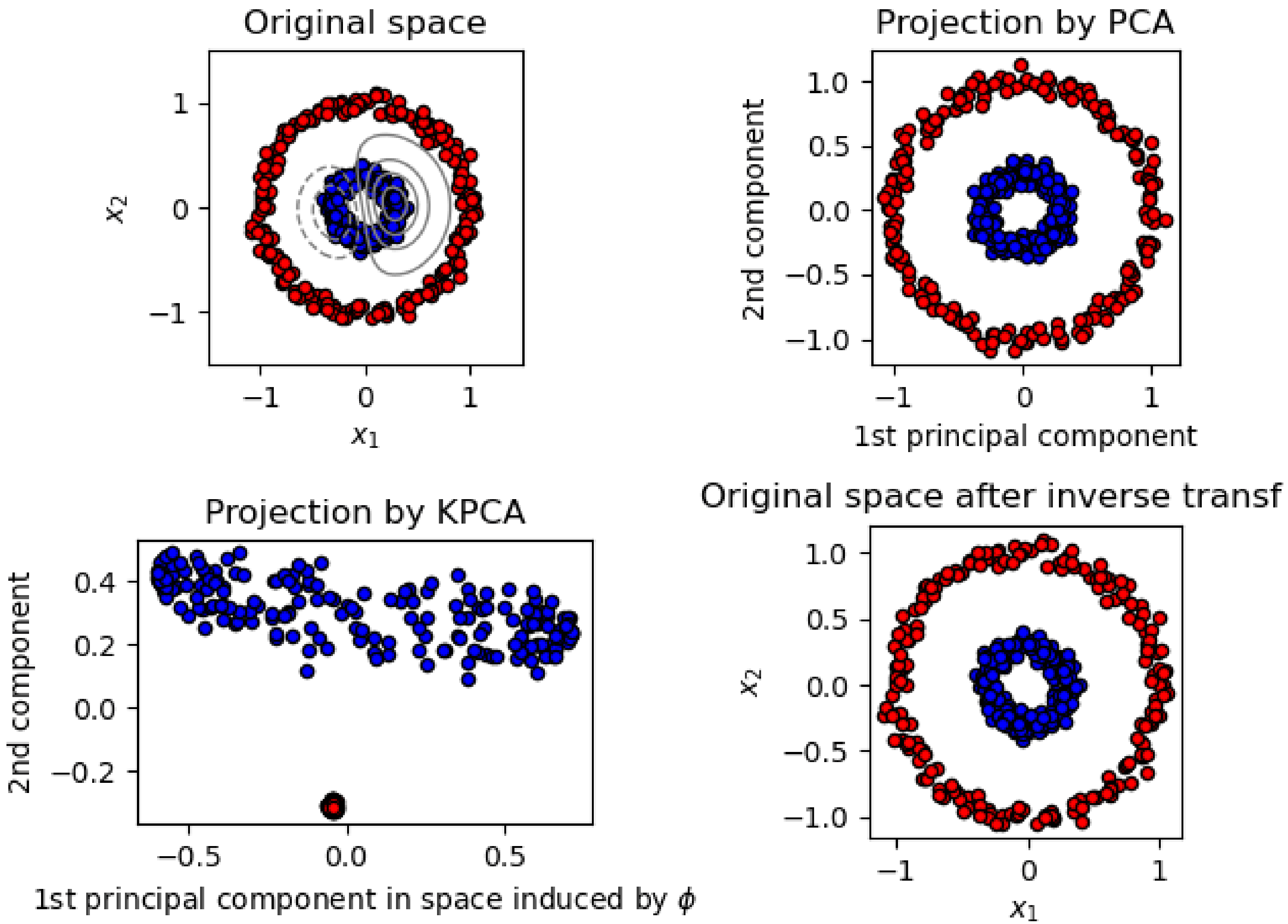
$$k(\mathbf{x}, \mathbf{y}) = (\mathbf{x}^T \mathbf{y} + 1)^d$$

Polynomial

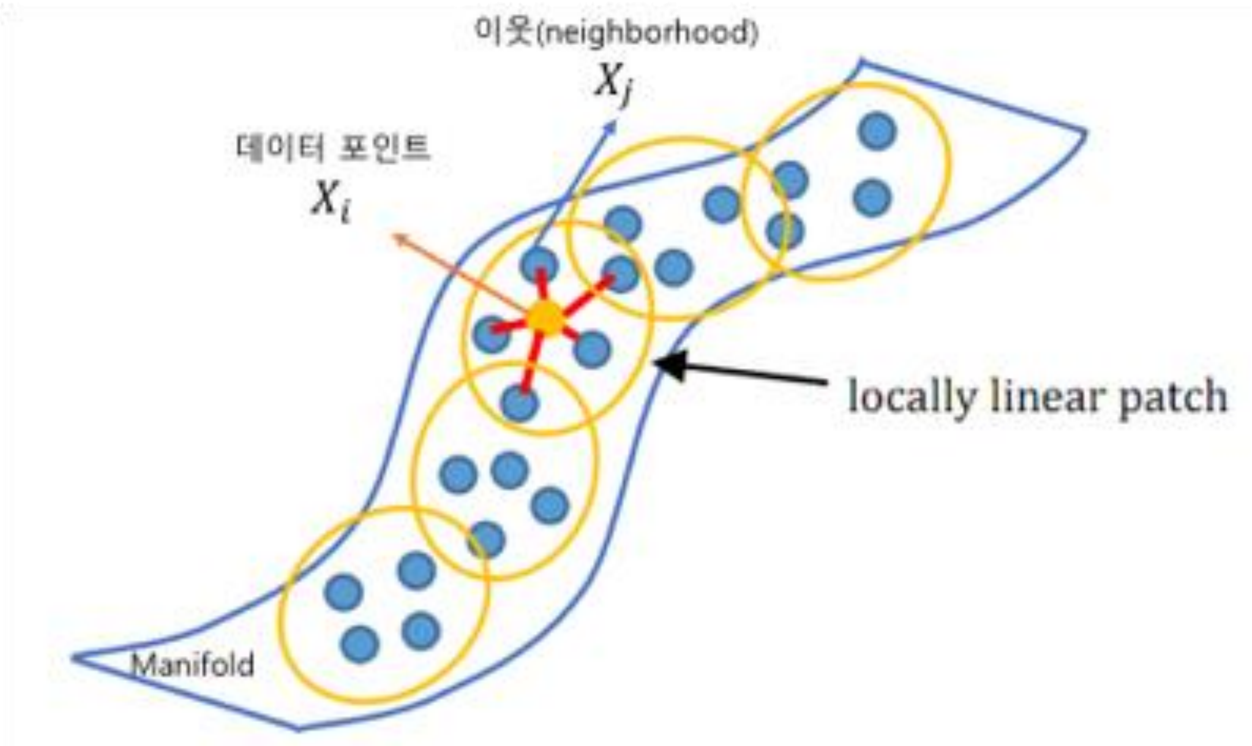
$$k(\mathbf{x}, \mathbf{y}) = e^{-\gamma \|\mathbf{x} - \mathbf{y}\|^2}, \gamma > 0$$

Radial Basis

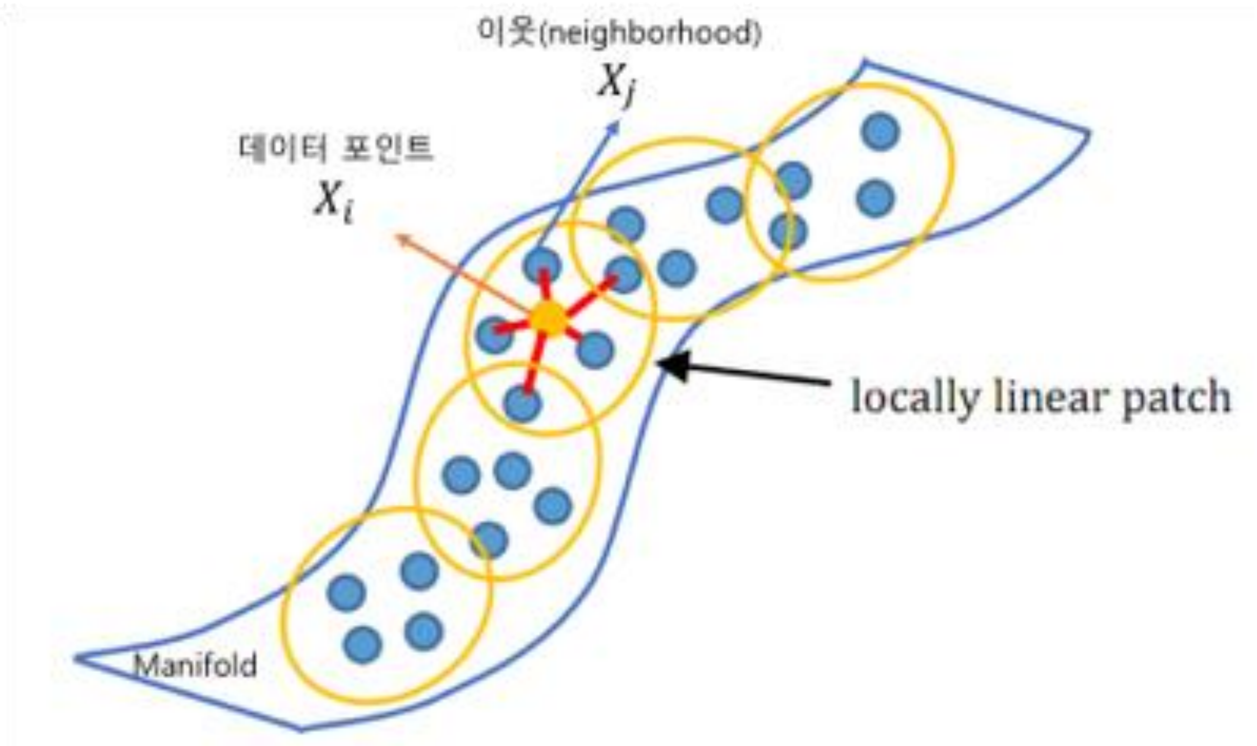
PCA: Kernel approach



LLE: Non-linear Dimension Reduction



LLE: Non-linear Dimension Reduction

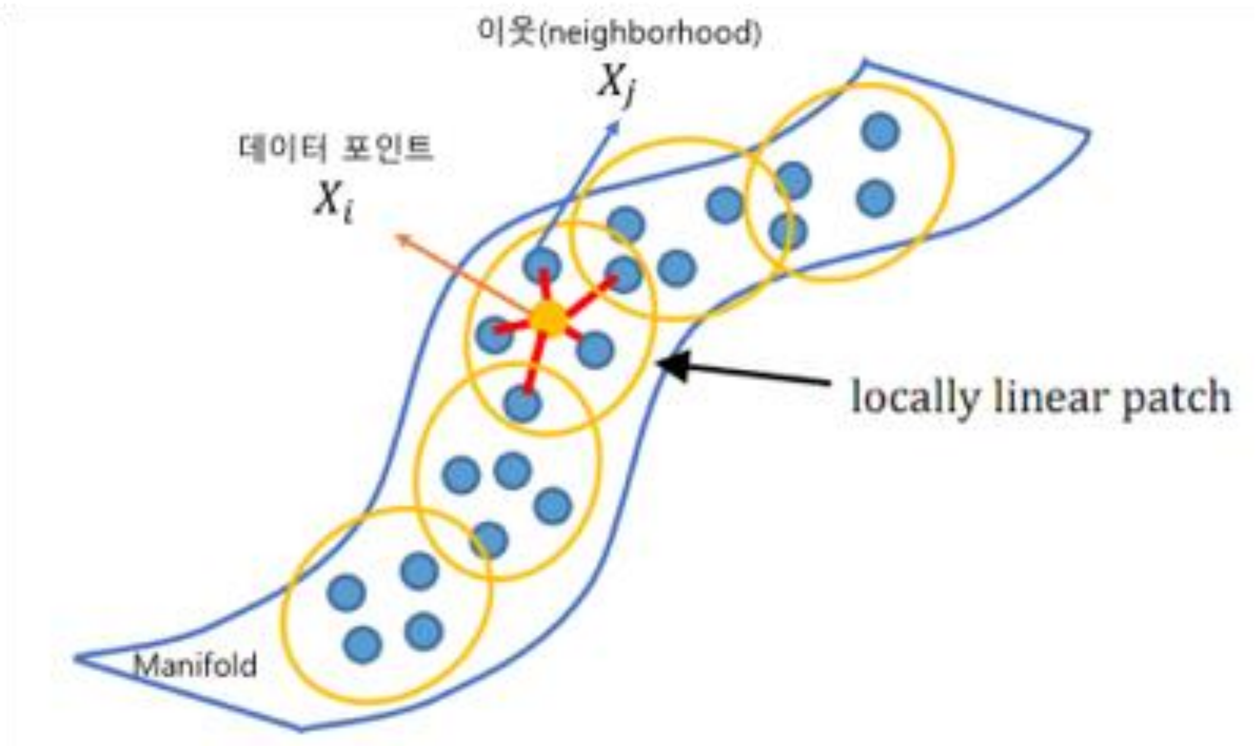


$$\sum_{j=1}^{k} w_{ij}\vec{x}_j \approx \vec{x}_i$$

$$\min \quad \varepsilon_i(\mathbf{w}) = \left\| \vec{x}_i - \sum_{\substack{j=1 \\ j \neq i}}^{k} w_{ij}\vec{x}_j \right\|^2$$

$$\text{s.t.} \quad \sum_{\substack{j=1 \\ j \neq i}}^{k} w_{ij} = 1$$

LLE: Non-linear Dimension Reduction



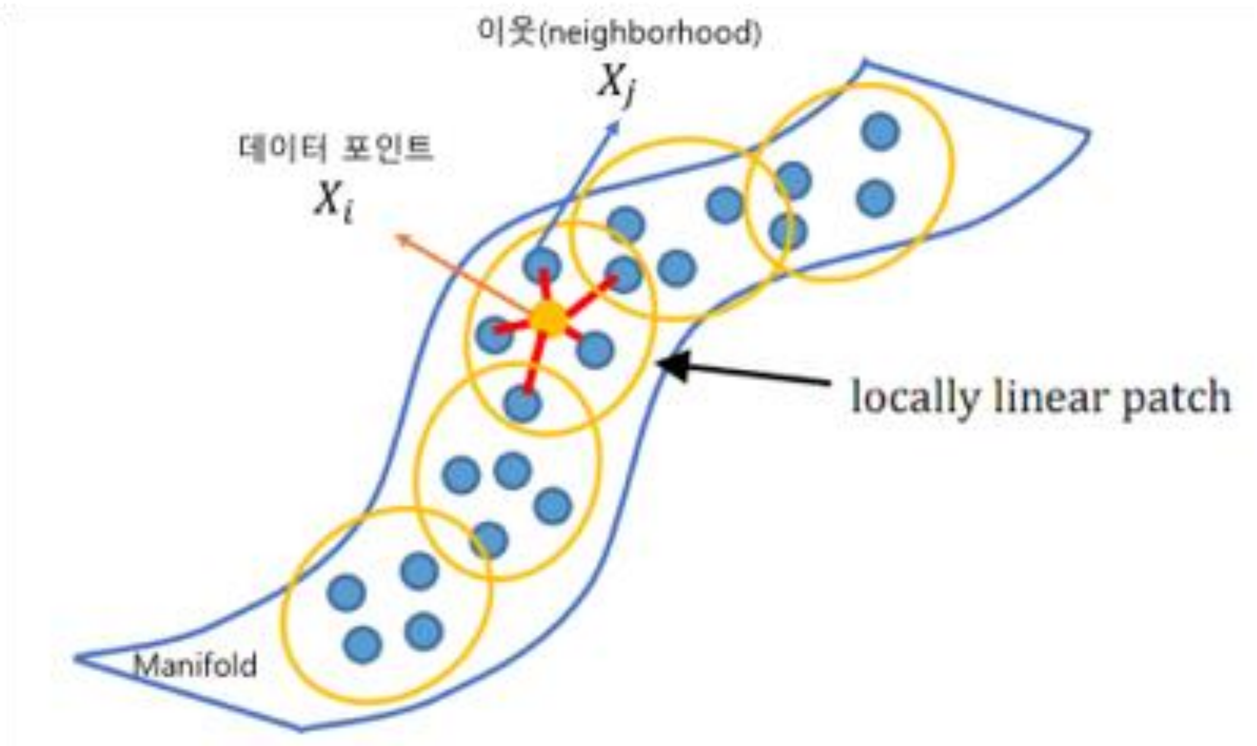
$$\min \quad \varepsilon_i(\mathbf{w}) = \left\| \vec{x_i} - \sum_{\substack{j=1 \\ j\neq i}}^{k} w_{ij} \vec{x_j} \right\|^2$$

$$\text{s.t.} \quad \sum_{\substack{j=1 \\ j\neq i}}^{k} w_{ij} = 1$$

$$\min \quad \varepsilon_i(\mathbf{w}) = \left\| \vec{x_i} - \sum_{\substack{j=1 \\ j\neq i}}^{k} w_{ij} \vec{x_j} \right\|^2 = \mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i$$

$$\text{s.t.} \quad \sum_{\substack{j=1 \\ j\neq i}}^{k} w_{ij} = 1 = \mathbf{1}^T \mathbf{w}_i$$

LLE: Non-linear Dimension Reduction



$$\min \quad \varepsilon_i(\mathbf{w}) = \left\| \vec{x_i} - \sum_{\substack{j=1 \\ j\neq i}}^{k} w_{ij}\vec{x_j} \right\|^2 = \mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i$$

$$\text{s.t.} \quad \sum_{\substack{j=1 \\ j\neq i}}^{k} w_{ij} = 1 = \mathbf{1}^T \mathbf{w}_i$$

Lagrangian Function L

$$L(\mathbf{w}_i, \lambda) = \mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i - \lambda \left(\mathbf{1}^T \mathbf{w}_i - 1\right)$$

Partial differentiation

$$\begin{aligned} \frac{\partial L}{\partial \mathbf{w}_i} &= \left(\mathbf{G}_i + \mathbf{G}_i^T\right)\mathbf{w}_i - \lambda \mathbf{1}, \quad (\mathbf{G}_i = \mathbf{G}_i^T) \\ &= 2\mathbf{G}_i \mathbf{w}_i - \lambda \mathbf{1} \\ &= 0 \end{aligned}$$

$$\therefore \mathbf{w}_i = \frac{\lambda}{2}\mathbf{G}_i^{-1}\mathbf{1}$$

LLE: Non-linear Dimension Reduction

$$\min \quad \Phi(\mathbf{Y}) = \sum_{i=1}^{m} \left\| \vec{y}_i - \sum_{\substack{j=1 \\ j \neq i}}^{k} w_{ij} \vec{y}_j \right\|^2$$

O(m log(m) n log(k)) for finding the *k* nearest neighbors

O(mnk^3) for optimizing the weights

O(d**m^2**) for constructing the low-dimensional representations

Other Dimensionality Reduction Techniques

- **MDS**: Multidimensional Scaling
- **Isomap**
- **t-SNE**: t-Distributed **Stochastic** Neighbor Embedding
- **LDA**: **Linear** Discriminant Analysis(Actually a classification algorithm)

Done!

수고하셨습니다

Jiwoon Lee @metr0jw
https://metr0jw.studio/