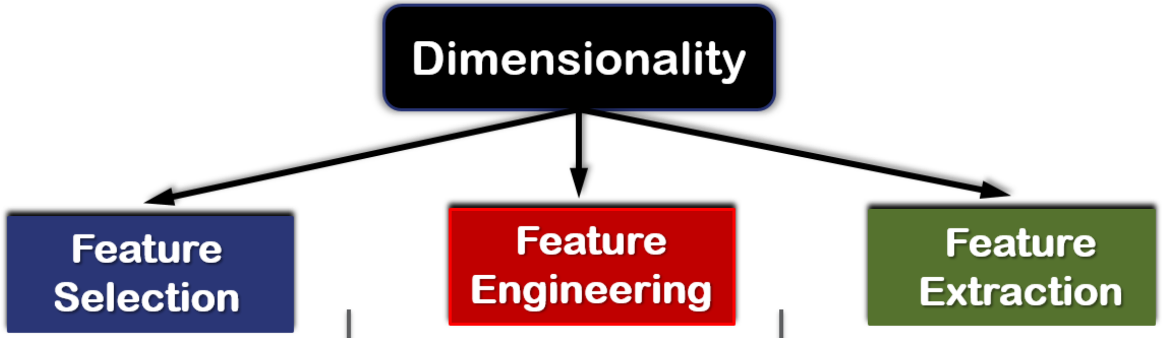
PCA: Dimension Reduction



Select a **subset** of features
Criteria:
- Information gain
- Entropy
- Gini Index
- Correlation
- Domain Knowledge

Learn new **feature functions**
- Cluster Centroids
- Cluster Membership
- Deep Learning
- Domain Knowledge

Extract **a combination** of features
- Multidimensional scaling
- Principal Component Analysis

- Dimensionality and underdetermined problems
- Feature selection vs. feature extraction vs. feature engineering
- Dimension reduction (DR): unsupervised vs. supervised; linear vs. non-linear; orthogonal vs. non-orthogonal
- DR: linear, orthogonal with Principal Component Analysis (PCA)
- Eigenvalue and eigenvectors

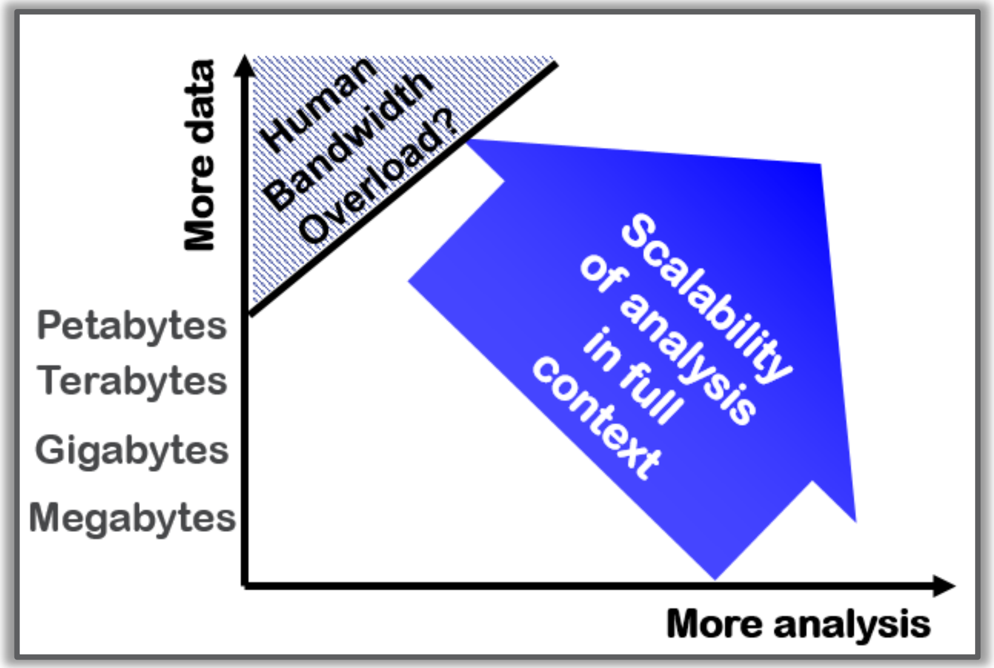
Prof. Nagiza F. Samatova
samatova@csc.ncsu.edu
Department of Computer Science
North Carolina State University

NC STATE Executive Education

February 2019

Dimension Reduction

MOTIVATION



What Analysis Methods to Use?

Analysis methods fail for a few gigabytes.

Method Complexity:

Calculate means $O(n)$

Calculate Histogram $O(n\,log(n))$

Calculate PCA $O(n \bullet d)$

Clustering algorithms $O(n^2)$

If n =10GB, then what is $O(n)$ or $O(n^2)$ on a teraflop computers?

1GB = 10^9 bytes 1Tflop = 10^{12} op/sec

$O(\ldots)$- in the order of
n – number of rows
d - number of columns

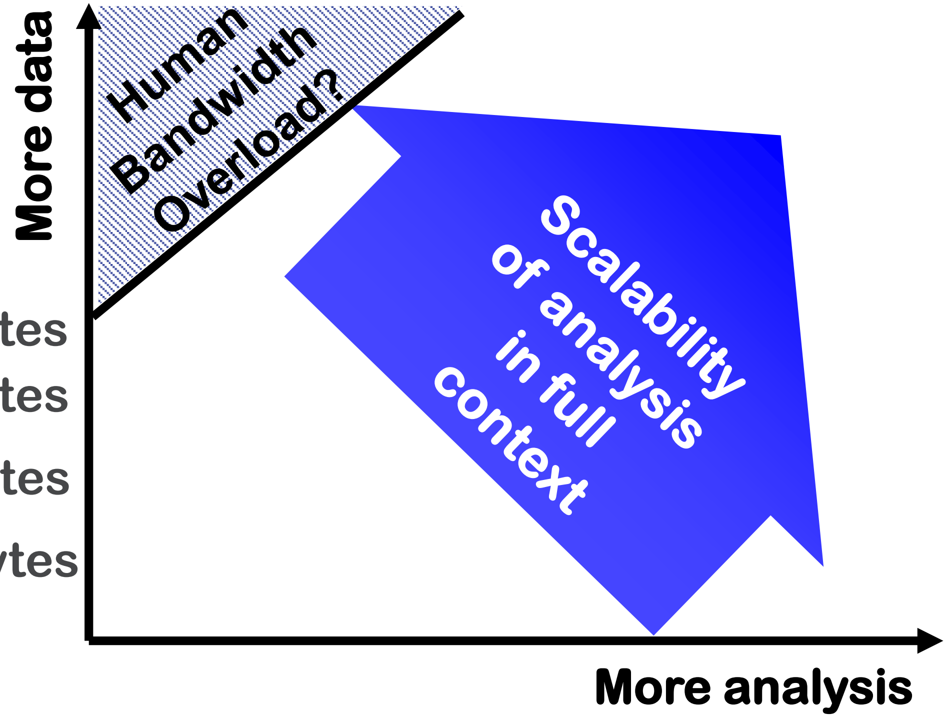
Data size n	Algorithm Complexity		
	n	$nlog(n)$	n^2
100B	10^{-10}sec.	10^{-10} sec.	10^{-8} sec.
10KB	10^{-8} sec.	10^{-8} sec.	10^{-4}sec.
1MB	10^{-6} sec.	10^{-5} sec.	1 sec.
100MB	10^{-4} sec.	10^{-3} sec.	3 hrs
10GB	10^{-2} sec.	0.1 sec.	3 yrs.

For illustration chart assumes 10^{-12} sec. (1Tflop/sec) calculation time per data point

How to Make Sense of Data?

Know Your Limits & Be Smart!

Not humanly possible to browse a petabyte of data.
Analysis must reduce data to quantities of interest.

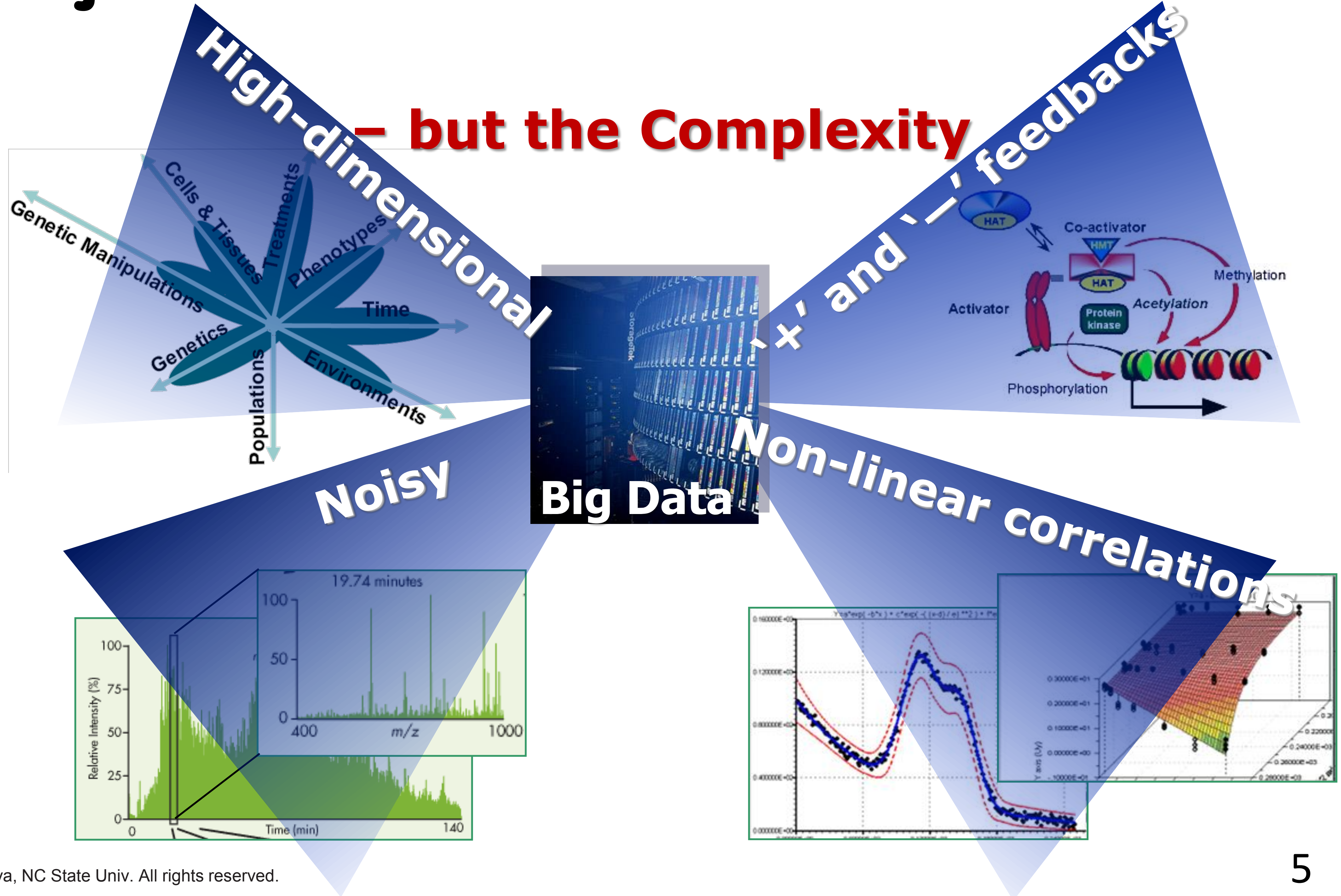


Computations:
Must be smart about which probe combinations to see!

Physical Experiments:
Must be smart about probe placement!

To see 1 percent of a petabyte at 10 megabytes per second takes:
35 8-hour days!

It is not just the Size



High-dimensional Data

- **Text Data**
  - Record/Row: Document ID
  - Dimension/Column: Each word in a collection of text documents
- **Image Data**
  - Record/Row: Image ID
  - Dimension/Column: Each pixel
- **Audio Data**
  - Record/Row: Audio record ID
  - Dimension/Column: Frequency of an audio signal OR a word if audio → text conversion
- **Sales Data**
  - Record/Row: Transaction ID
  - Dimension/Column: Each Product ID

Taxonomy: Dimensionality-aware Methods

FEATURE SELECTION, FEATURE EXTRACTION, FEATURE ENGINEERING

Dimensionality Challenge: Strategies to Cope With

Dimensionality

Feature Selection | **Feature Engineering** | **Feature Extraction**

Select a subset of features

<u>Criteria:</u>

- Information gain
- Entropy
- Gini Index
- Correlation
- Domain Knowledge

Learn new feature functions

- Cluster Centroids
- Cluster Membership
- Deep Learning
- Domain Knowledge

Extract a combination of features

- Multidimensional scaling
- Principal Component Analysis

Motivation: Why to Cope with Dimensionality Issue?

- **Not all the measured variables are important for understanding the underlying "interesting" phenomena:**
  - could complicate the process of data analysis
- **Decrease the computational cost for other data mining tasks:**
  - Proximity measure calculations: O(d) → O(k), d>>k
- **Reduce the noise in the data:**
  - improve signal to noise ratio
- **Improve the accuracy of predictive models**
  - better designed/engineered features capable of capturing non-linear signals in the data
- **Reduce collinearity among variables/features**
  - Critical for regression models

What is Dimensionality Reduction (DR)?

Objective: To transform data from a high-dimensional representation to a low-dimensional representation, while "best" preserving the information.

Given dataset with n objects:

$$\boldsymbol{X} \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{m}} \xrightarrow{\text{DR}} \boldsymbol{X}' \in \mathbb{R}^{\boldsymbol{n}\times\boldsymbol{p}}$$

m dimensions

p dimensions

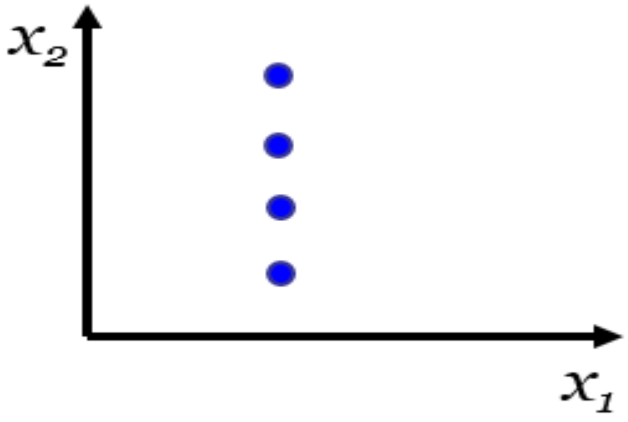
where $p << m$

Text Mining Example:

- $m = 50,000$ words → $p \approx 100$ extracted features

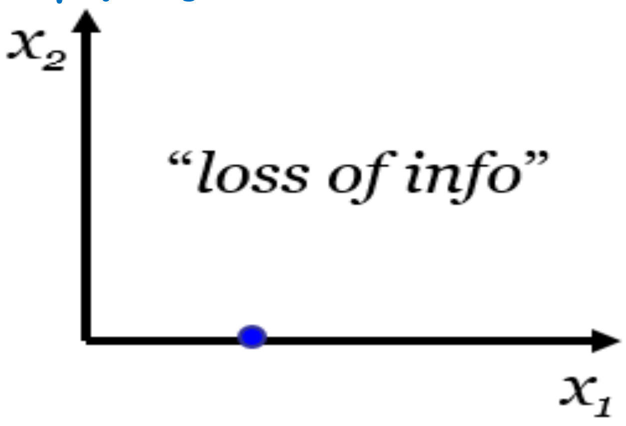
Example 1: p=2→d=1

Original data, $p = 2$



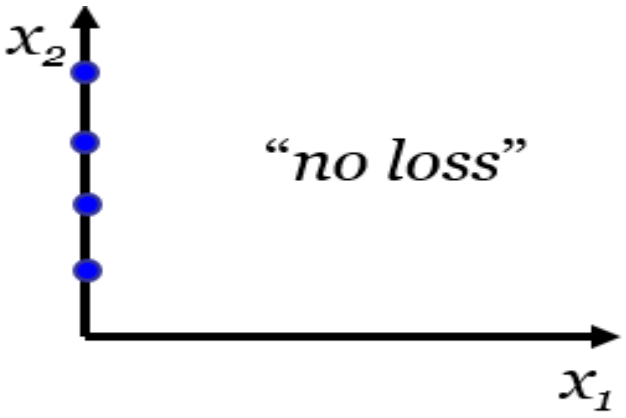
$$X_{4\times 2}$$

x_1 -projection, k=1



$$X'_{4\times 2} = X_{4\times 2} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}$$

x_2 -projection, $k = 1$



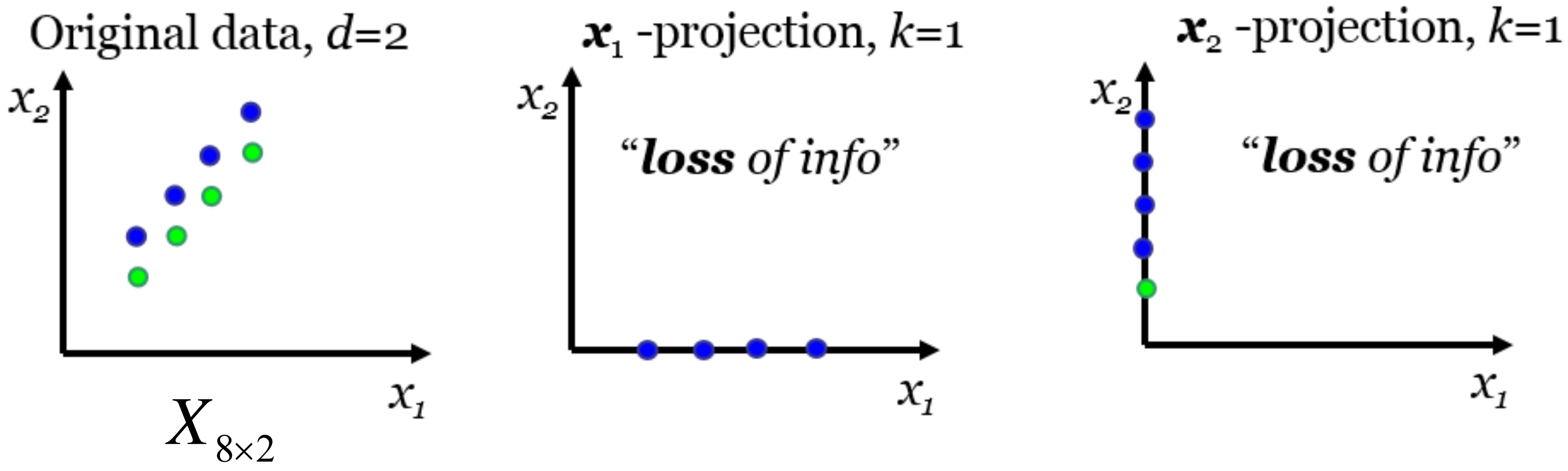
$$X'_{4\times 2} = X_{4\times 2} \cdot \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

$$X'_{m\times d} = X_{m\times d} \cdot P_{d\times d}$$

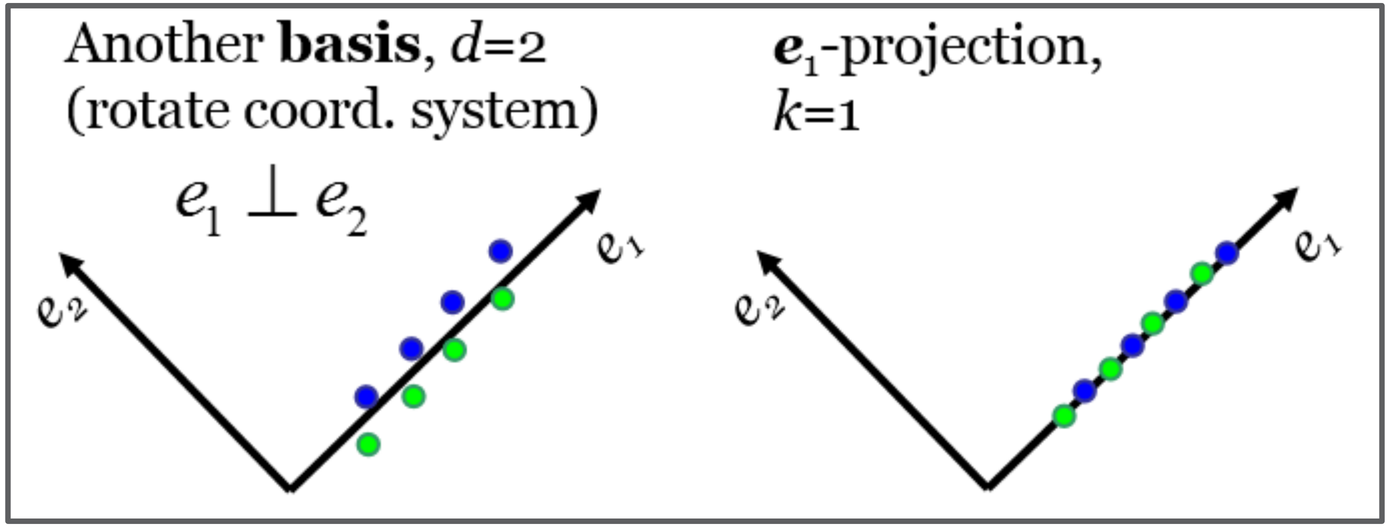
$$\boldsymbol{P}_{d\times d} = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & 0 & \ldots & 0 \\ \ldots & \ldots & \ldots & \ldots \\ 0 & 0 & 0 & 1 \end{bmatrix} - \text{projection matrix; some diagonal elements are 0}$$

Which projection is "better"?

Example 2: Linear, Orthogonal Projection



Is there a "better" projection?



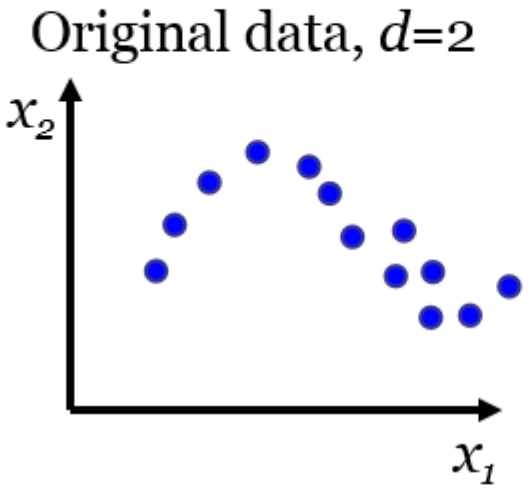
Projection:

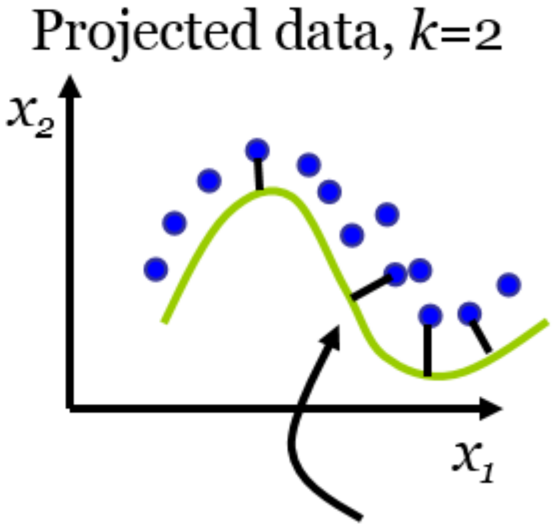
- **Linear,** $\boldsymbol{e}_1$ – line
- **Orthogonal**

$$e_1 \perp e_2$$

$\boldsymbol{e}_1$, $\boldsymbol{e}_2$ – eigenvectors of X

Example 3: Non-linear projection

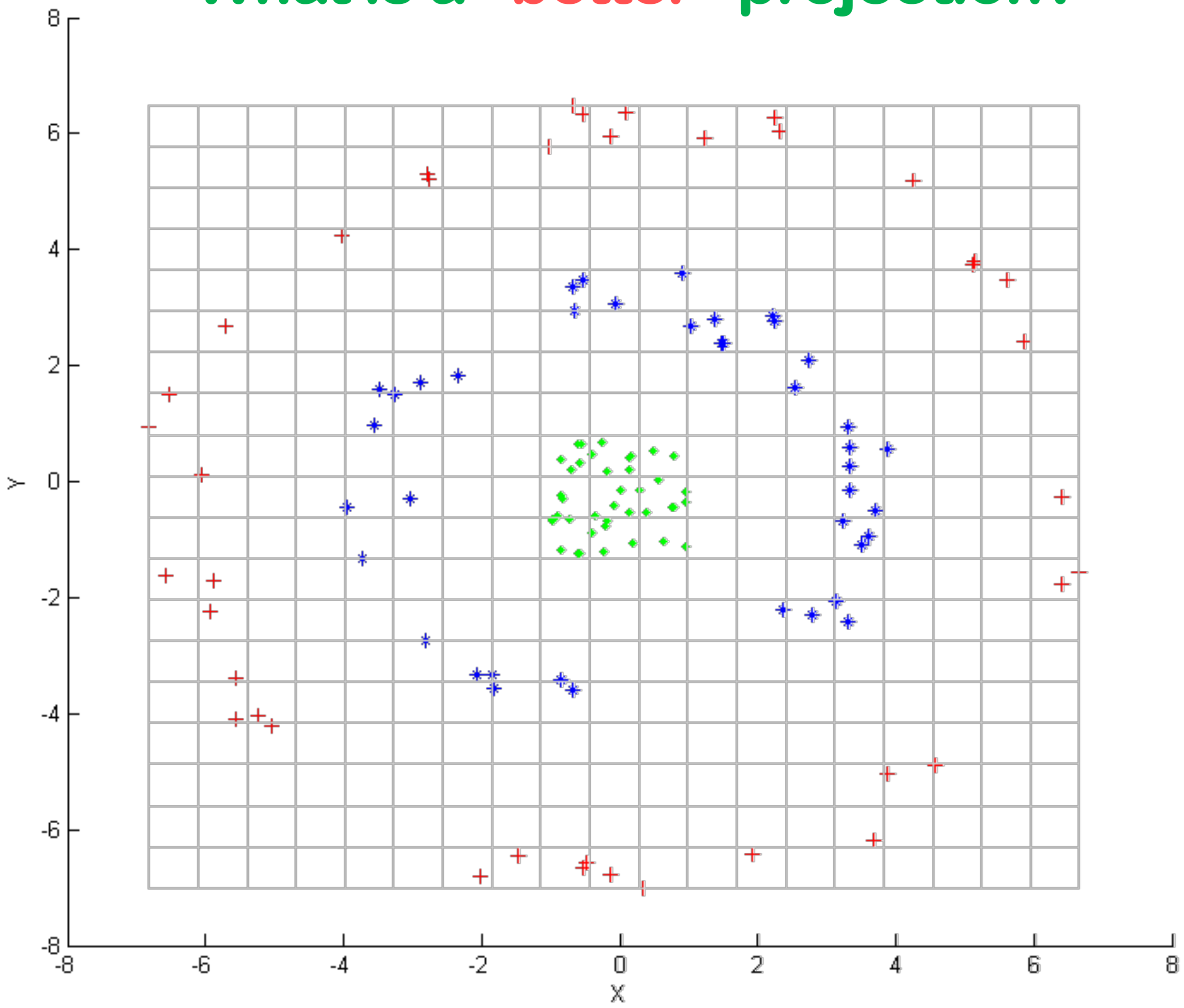




NON-linear projection

Example 4: Projection for Labeled Data

What is a "better" projection?



Ideal for machine learning (ML) algorithm:

- Points within the same group come from a Gaussian distribution (spherical shapes)
- Points from different groups are linearly separable

"better" means:

- it bests, ideally, linearly separates different groups of data, i.e.
- points from the same group are closer to each other and are farther away from the points in different groups

Summary: Taxonomy of Dimension Reduction

- **Linear vs. NON-linear**
- **Orthogonal vs. non-orthogonal**
- **Unsupervised (unlabeled data) vs. supervised (labeled data)**

Linear dimension reduction:
- Interpretable in original space
- Preserves non-linearity for visualization
- Orthogonal projections
- Non-orthogonal projections
- Favored for structure discovery

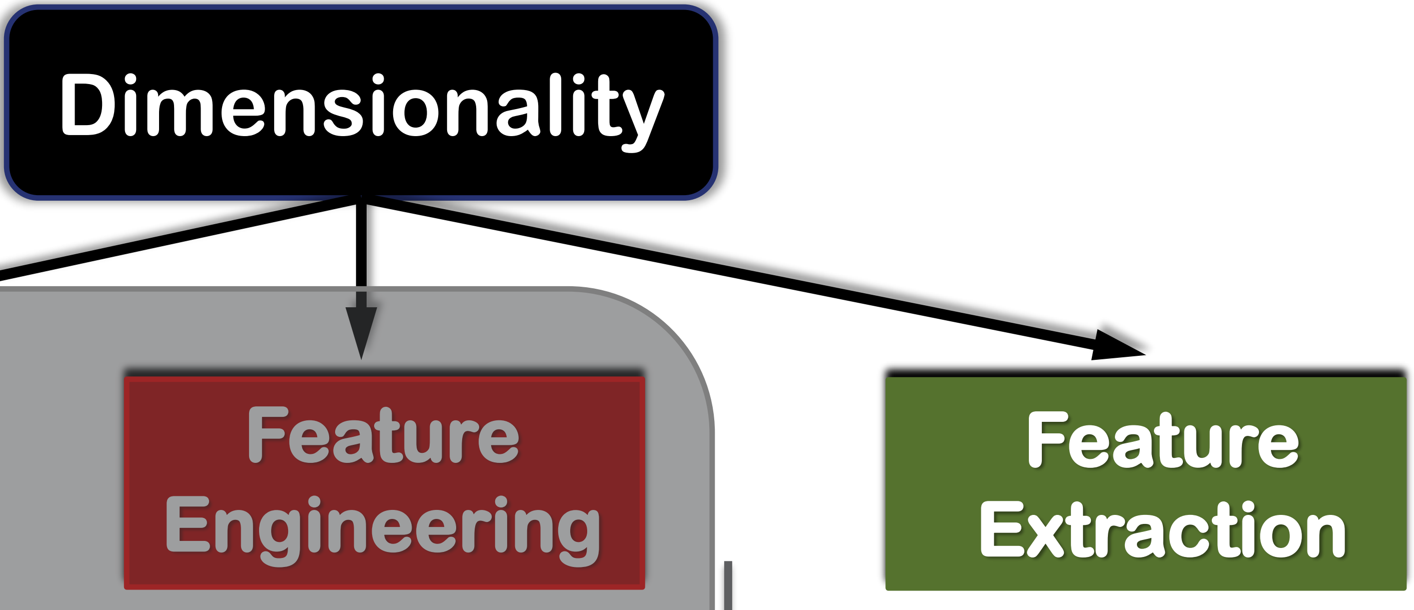
Non-linear dimension reduction:
- Lower-dimensional representation
- Interpretable w.r.t. Non-linear transformation
- 1 to 3 orders of magnitude more computation
- Favored for prediction or classification

- A lower-dimensional representation that contains the essence of the high dimensional data
- Blessing of dependence/correlation that saves us from the curse of dimensionality

Linear Orthogonal Dimension Reduction

PCA: PRINCIPAL COMPONENT ANALYSIS

Dimensionality Challenge: Strategies to Cope With



Select a subset of features

Criteria:

- Information gain
- Entropy
- Gini Index
- Correlation
- Domain Knowledge

Learn new feature functions

- Cluster Centroids
- Cluster Membership
- Deep Learning
- Domain Knowledge

Extract a combination of features

- Multidimensional scaling
- Principal Component Analysis

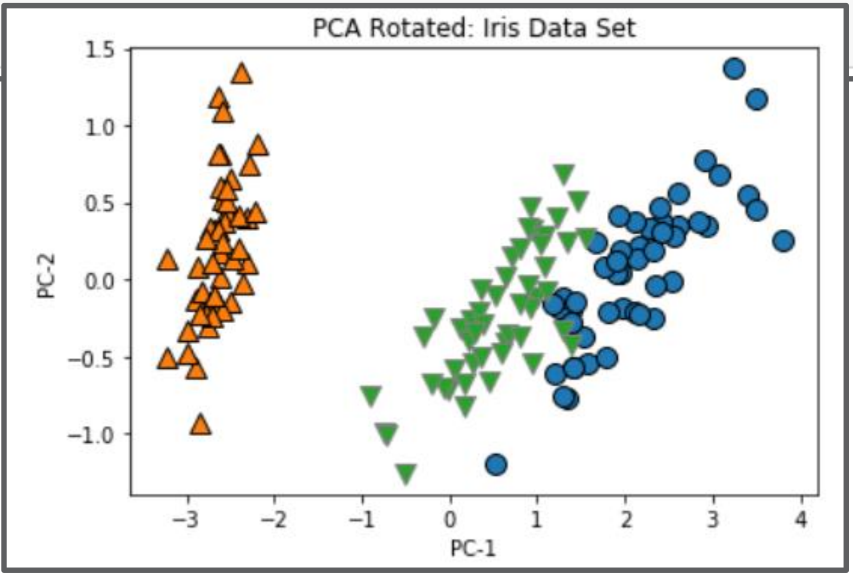
What is “better” projection: $d \rightarrow k$ ($k<d$)?

- **Many definitions are possible**
- **Definition 1:**
  - Projection that maximizes the VARIANCE of the original d-dimensional data upon its projection onto the target k-dimensions ($k < d$)

Python Code Example: Iris Data

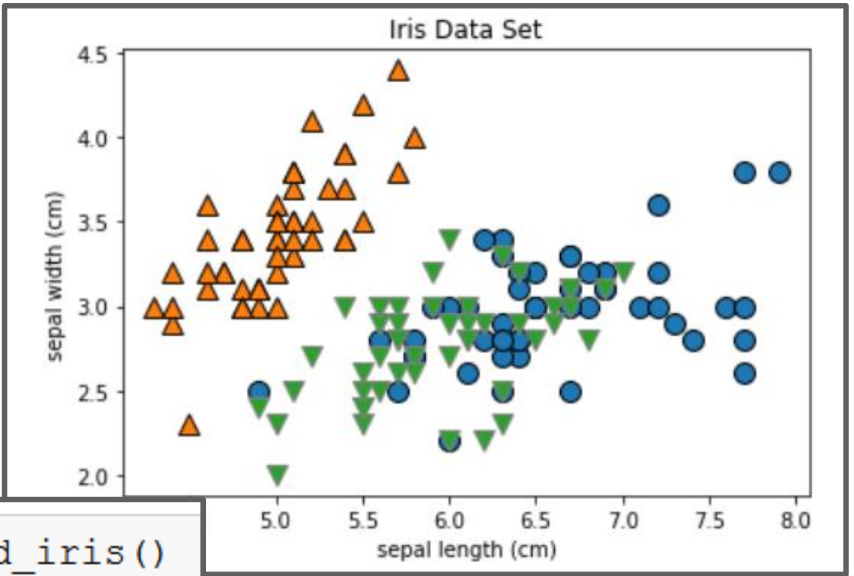
```
pca = decomposition.PCA(n_components=3)
pca.fit(X)
X_rot = pca.transform(X)
```

```
# Plot rotated 4-D iris data in 2D: the first two PCs
mglearn.discrete_scatter(X_rot[:, 0], X_rot[:, 1], y)
plt.title("PCA Rotated: Iris Data Set")
plt.xlabel("PC-1")
plt.ylabel("PC-2")
plt.show()
```



PCA-transformed data in 2-d

original 4-d data in 2-d

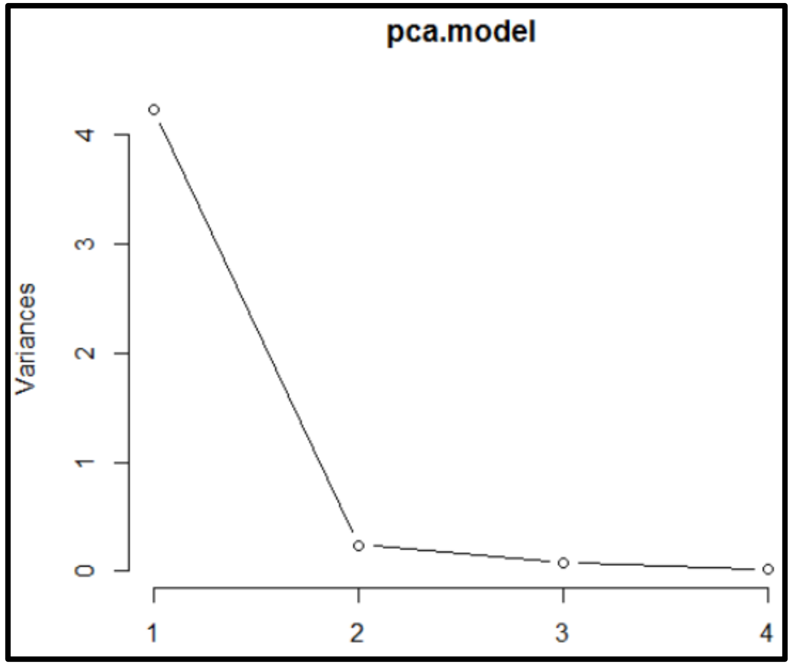



```
iris = datasets.load_iris()
X = iris.data
y = iris.target
print(iris.feature_names)
print(X[0:5])
print (iris.target_names)
print (y[[0, 51, 101]])
```

```
# Plot original 4-D iris data in 2D
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.title("Iris Data Set")
plt.xlabel("sepal length (cm)")
plt.ylabel("sepal width (cm)")
plt.show()
```

Visualizing PCA Results: Iris Data

Screeplot



Successive variance accounted by each component

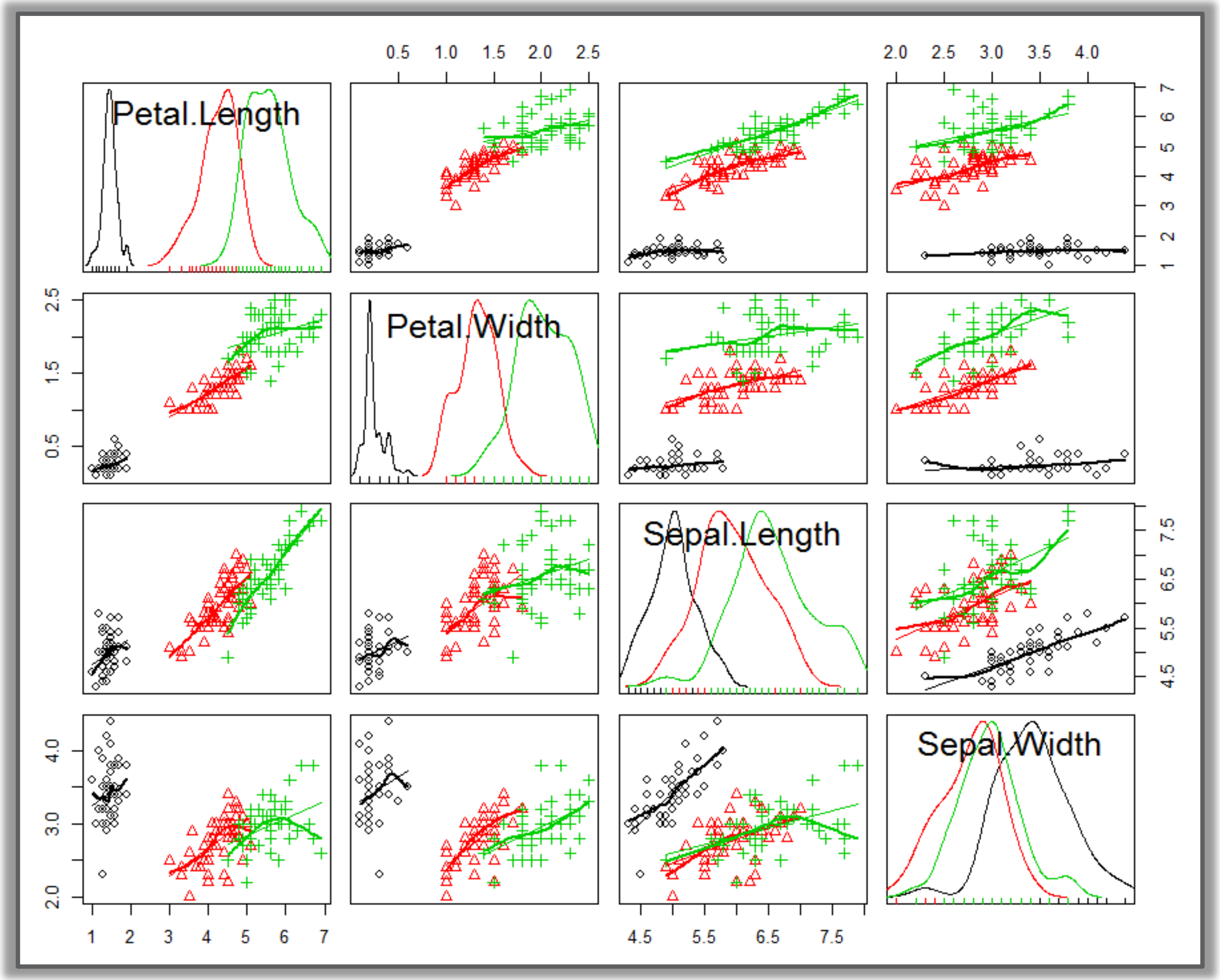
```
Component loadings:
                 Comp.1     Comp.2     Comp.3     Comp.4
Petal.Length  0.5804131 0.02449161  0.1421264  0.8014492
Petal.Width   0.5648565 0.06694199  0.6342727 -0.5235971
Sepal.Length  0.5210659 0.37741762 -0.7195664 -0.2612863
Sepal.Width  -0.2693474 0.92329566  0.2443818  0.1235096

Component variances:
    Comp.1     Comp.2     Comp.3     Comp.4
2.91849782 0.91403047 0.14675688 0.02071484

Importance of components:
                          Comp.1    Comp.2     Comp.3      Comp.4
Standard deviation     1.7083611 0.9560494 0.38308860 0.143926497
Proportion of Variance 0.7296245 0.2285076 0.03668922 0.005178709
Cumulative Proportion  0.7296245 0.9581321 0.99482129 1.000000000
```

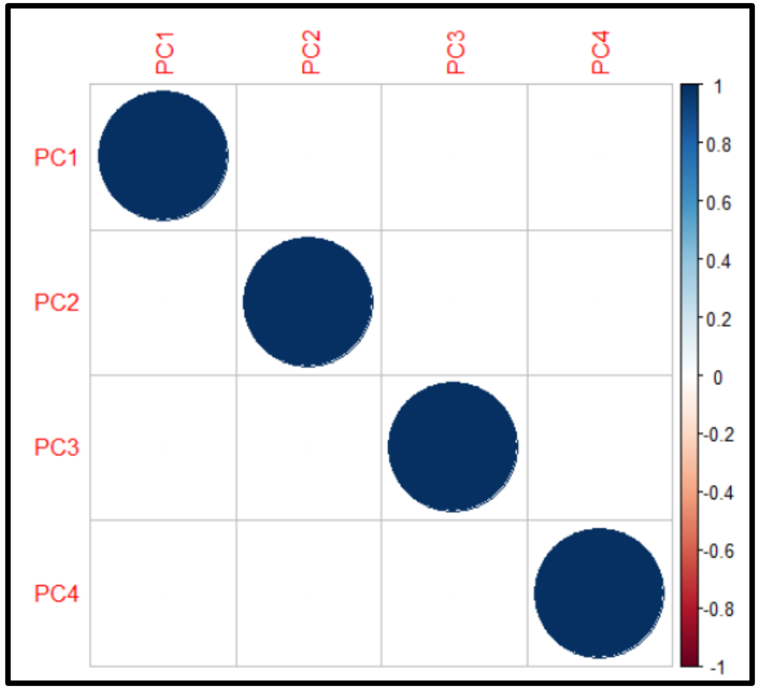
Original Relationships: Features are Correlated!

Scatterplot Matrix



Extracted Principal Components (PCs) are Uncorrelated

	PC1	PC2	PC3	PC4
PC1	1e+00	-1e-16	-1e-15	2e-15
PC2	-1e-16	1e+00	2e-16	-5e-16
PC3	-1e-15	2e-16	1e+00	-1e-15
PC4	2e-15	-5e-16	-1e-15	1e+00



The new extracted features (PCs) are UNCORRELATED!

PC is a weighted linear sum of original features

Extracted Feature → PCA-based Feature Extraction:

$$\boxed{PC = w_1 * f_1 + w_2 * f_2 + \cdots + w_d * f_d}$$

The magnitude of each **weight** indicates how important the corresponding feature is → it could be used as a **feature selection** technique!

	PC1	PC2	PC3	PC4
Sepal.Length	0.36	-0.66	0.58	0.3
Sepal.Width	-0.08	-0.73	-0.60	-0.3
Petal.Length	0.86	0.17	-0.08	-0.5
Petal.Width	0.36	0.08	-0.55	0.8

PC1 = 0.86 Petal.Length + 0.36 Petal.Width
+ 0.36 Sepal.Length – 0.08 Sepal.Width

PC2 = -0.73 Sepal.Width - 0.66 Sepal.Length
+ 0.17 Petal.Length + 0.08 Petal.Width

Preserved Variability for Top-k PCs: d → k (k << d)

Percentage of variability preserved if the first **k** PCs are used for **projection**:

$$\boxed{\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d} \lambda_j}}$$

SORTED Eigenvalues: pca.model$sdev

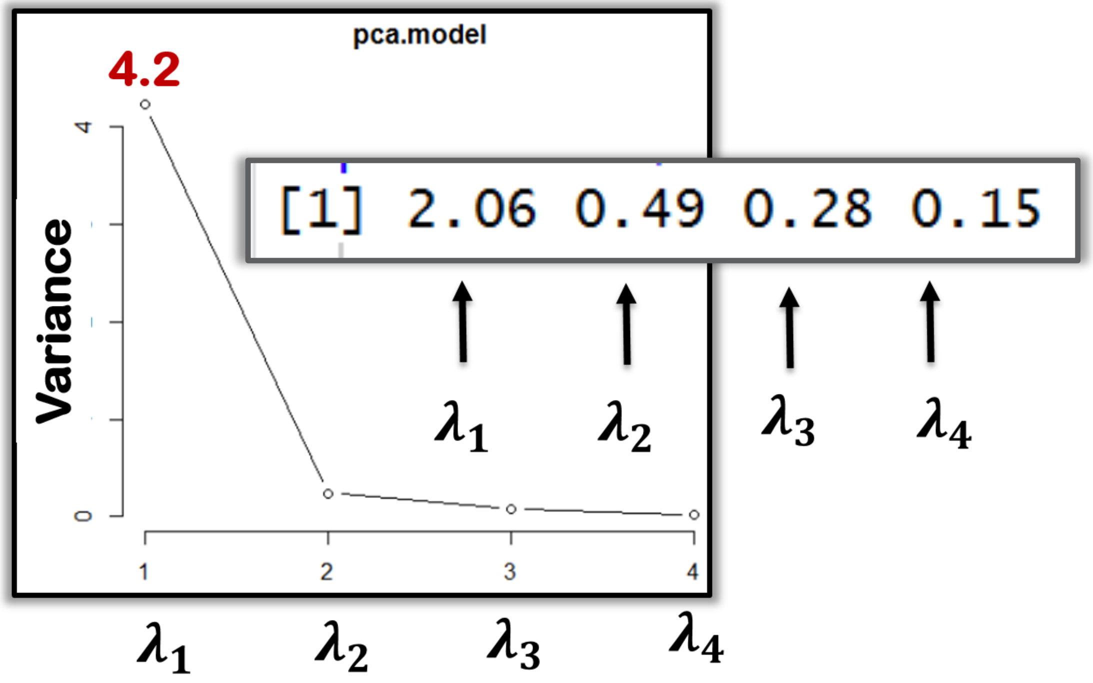
$$\underbrace{\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k}\geq \ldots \geq \lambda_d \geq 0$$

Standard deviation = sqrt (Variance)

```
Standard deviations (1, .., p=4):
[1] 2.06 0.49 0.28 0.15
```

Variance preserved by PC1:

$$2.06 \times 2.06 = 4.2$$



Eigenvalues: Importance of Eigenvectors

SORTED: $\boxed{\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_k \geq \ldots \geq \lambda_d \geq 0}$

Importance of Principal Components (PC), or Eigenvectors, or Rotations

PC1 preserves more variance than PC2
PC2 preserves more variance than PC3
....

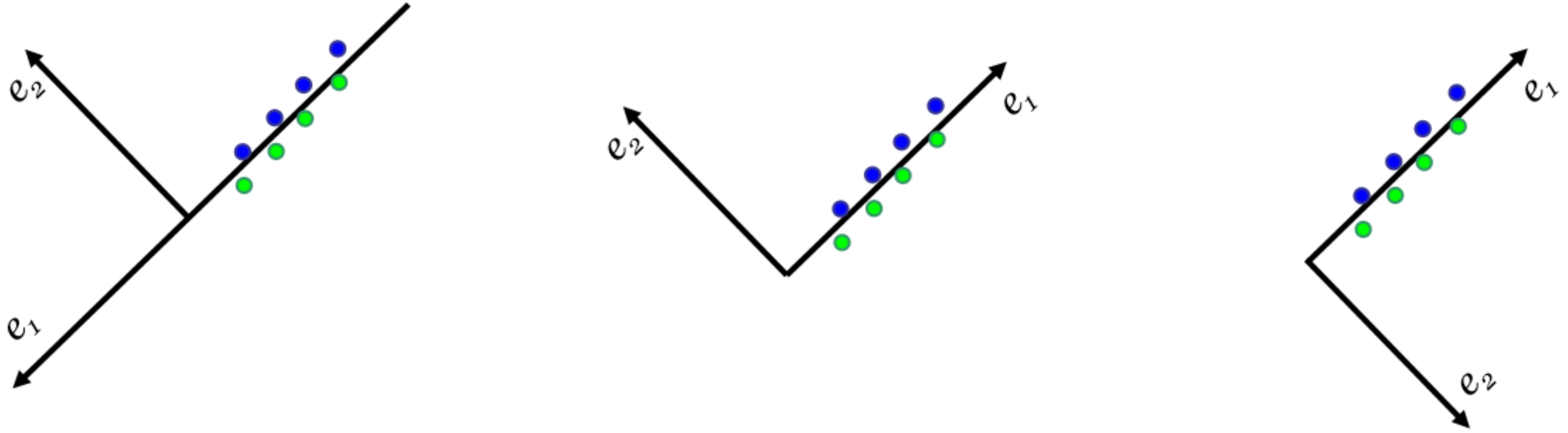
Proportion of Variance Preserved if only *k* PCs are used:

$$\boxed{\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{j=1}^{d} \lambda_j}}$$

Principal Components (eigenvectors) are NOT unique

PCA: Linear Orthogonal Transformation:

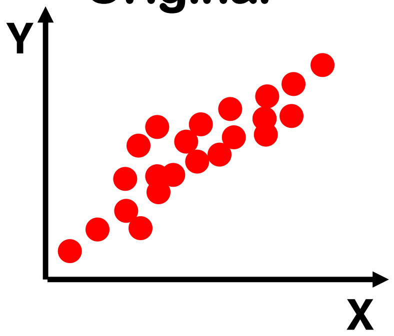
- PCs are simply a rotation of the original coordinate system
- Each PC is a line
- Each PC is perpendicular to the other PCs
  - PCs are Uncorrelated
  - The angle between any pair of PCs is $90°$



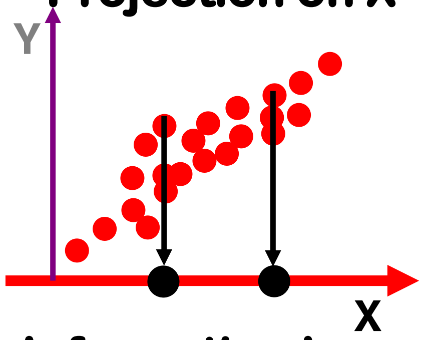
PCA: Linear Orthogonal Dimension Reduction

Principal Component Analysis (PCA) finds intrinsic dimensionality and allows for low-dimensional representation of the data.

Original



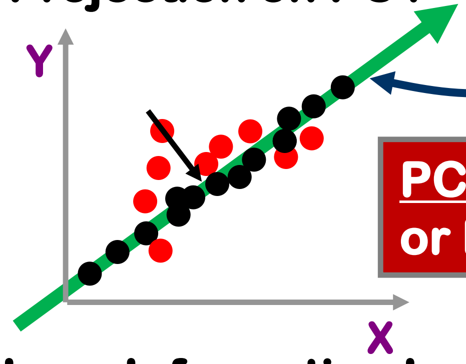
Projection on X



Information Loss

Principal Component Analysis is the Spectral Decomposition of the Covariance Matrix.

Projection on PC1



PC: Principal Component or Eigenvector

Less Information Loss

PCA: Covariance vs. Correlation Matrix

- **PCA results depend on the scales at which the variables are measured:**
  - PCA should only be used with the raw data if all variables have the same units of measure
- **PCA results depend on the variances of the variables: the ones with the highest sample variances will tend to be emphasized in the first few principal components:**
  - Use PCA with covariance matrix only if you wish to give variables with higher variances more weight in the analysis
- **If the variables either have different units of measurement (i.e., pounds, feet, gallons, etc), or if we wish each variable to receive equal weight in the analysis, then the variables should be standardized (Z-scores) before a principal components analysis is carried out.**