# Visualizing multidimensional data using PCA and TSNE

**Edgar Acuna**

**University of Puerto Rico at Mayaguez**
**Github: github.com/eacunafer**

**April 2019**

# Visualization techniques  without dimensionality reduction

Basically, the data points are mapped into either the 2D or 3D space considering the dependency among the features.

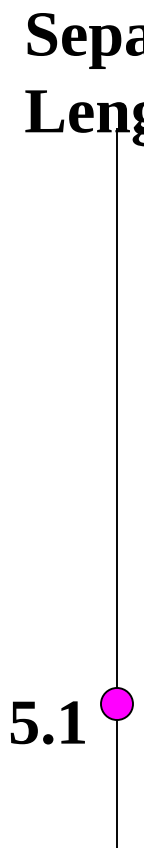
Among these techniques are,

- Parallel coordinates plot
- Radial visualization
- Star coordinates plot

# The parallel coordinate plot (Inselberg, 1985)

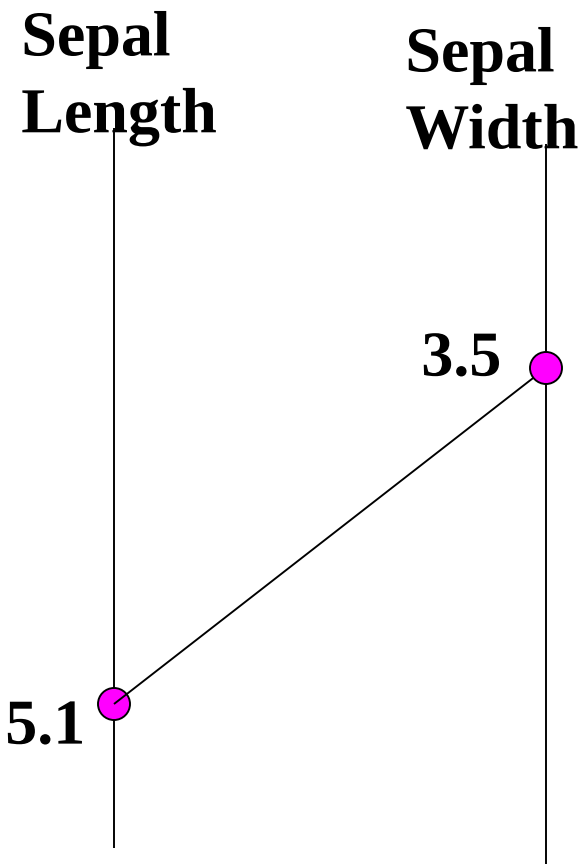
- The parallel coordinate plot represents multidimensional data using lines.
- Whereas in traditional Cartesian coordinates all axes are mutually perpendicular, in parallel coordinate plots, all axes are parallel to one another and equally spaced.
- In this approach, a point in *m*-dimensional space is represented as a series of *m*-1 line segments in 2-dimensional space. Thus, if the original data observation is written as ($x_1$, $x_2$, … $x_m$,), then its parallel coordinate representation is the *m*-1 line segments connecting points (1,$x_1$), (2,$x_2$), . . . (*m*,$x_m$).
- Typically, features will be standardized before a parallel coordinate plot is drawn.

# Parallel Coordinates



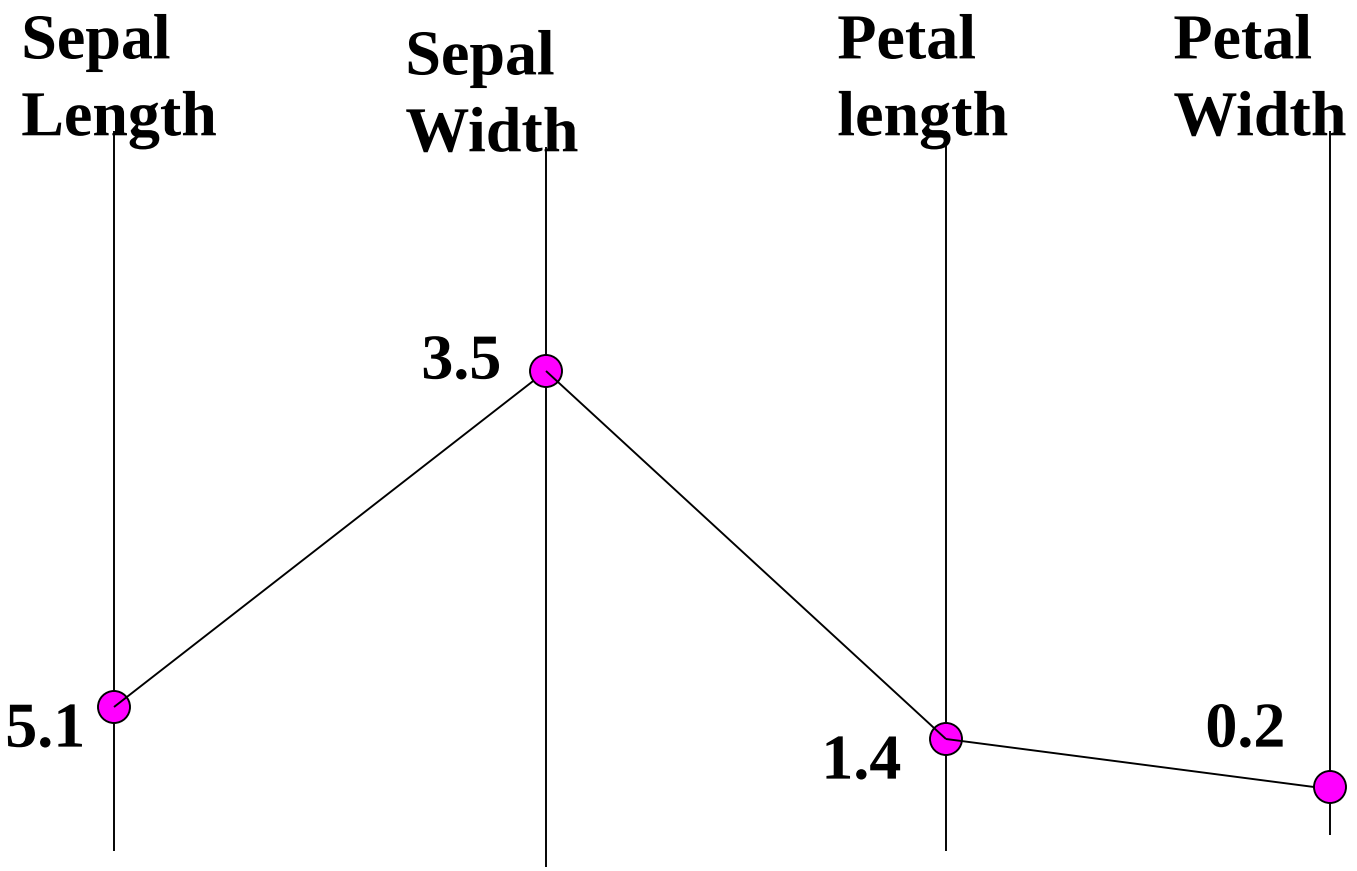

| sepal length | sepal width | petal length | petal width |
|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 |

# Parallel Coordinates: 2 D



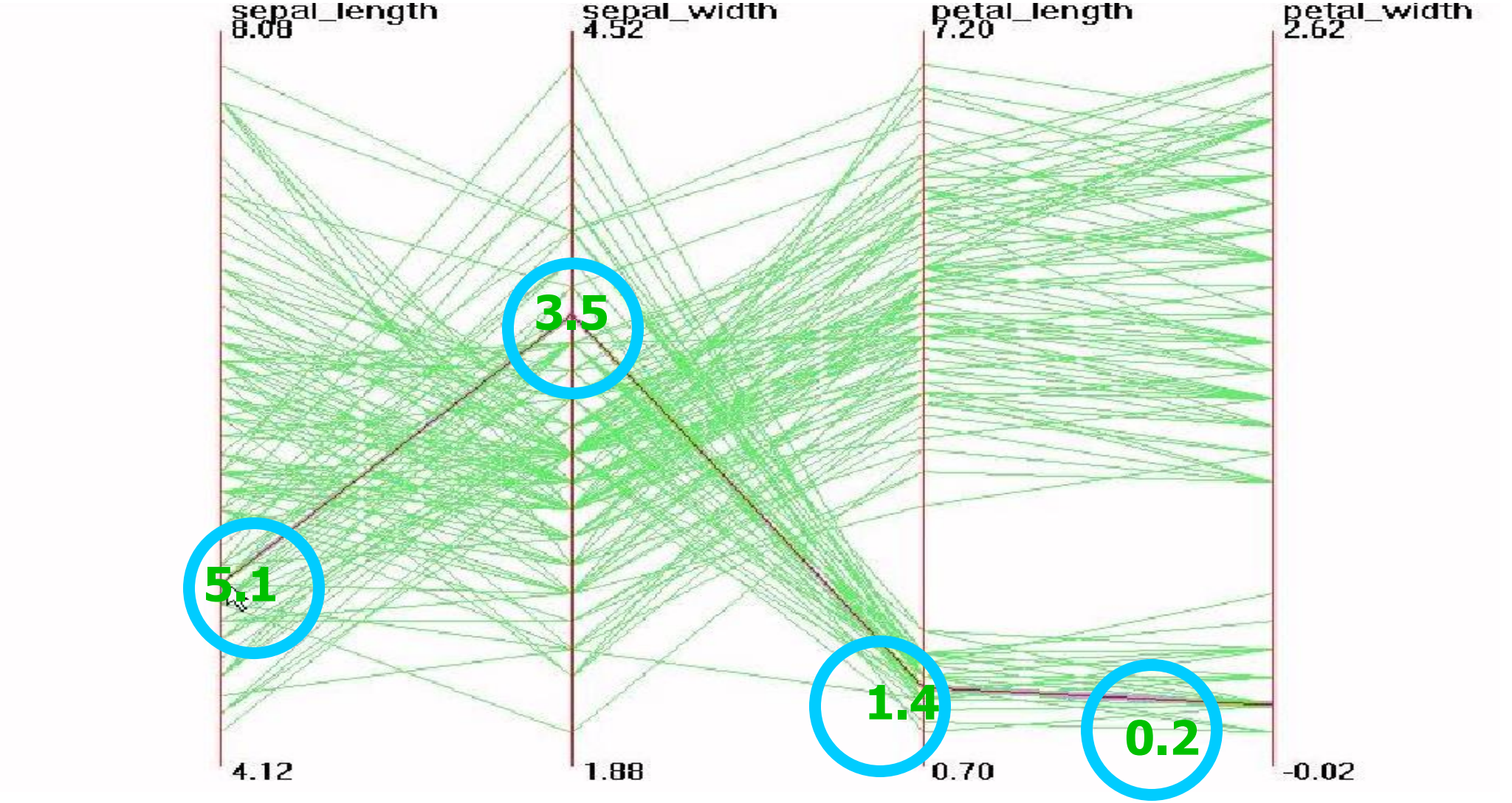

| sepal length | sepal width | petal length | petal width |
|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 |

# Parallel Coordinates: 4 D



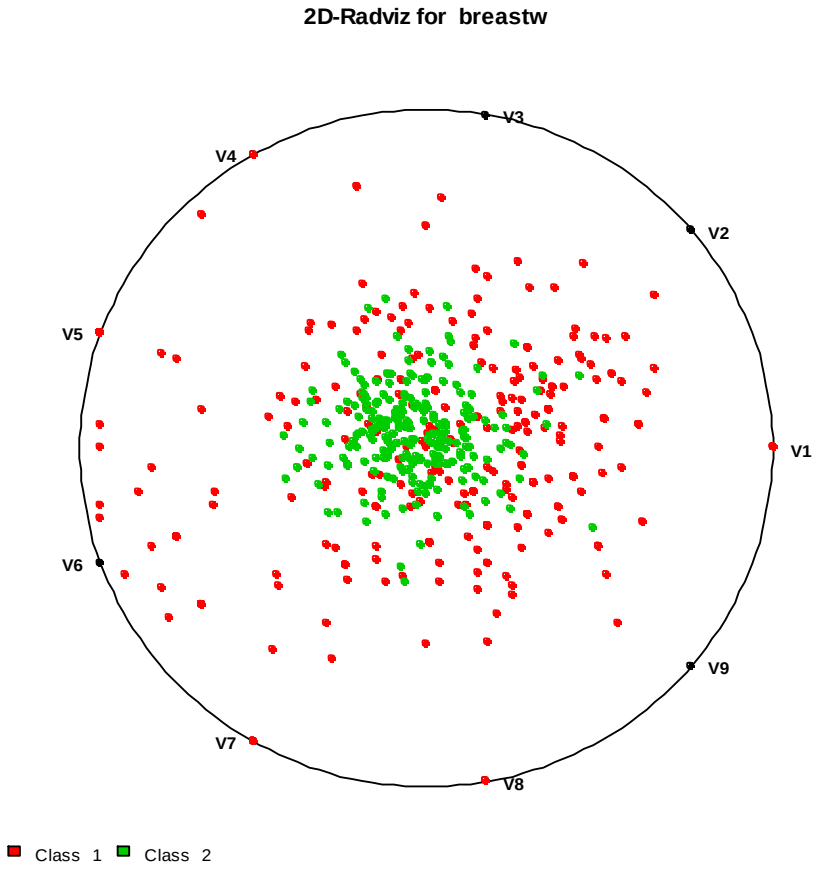

| sepal length | sepal width | petal length | petal width |
|---|---|---|---|
| 5.1 | 3.5 | 1.4 | 0.2 |

# Parallel Visualization of Iris data



# RadViz (Ankerst, et al., 1996)

- a radial visualization
- One spring for each feature .
- One end attached to perimeter point where the feature position is located. The other end attached to a data point.
- Each data point is displayed inside the circle where the sum of the spring forces equals 0.
- Good for outlier detection
- Esta disponible en Pandas, Plotly y en Orange.

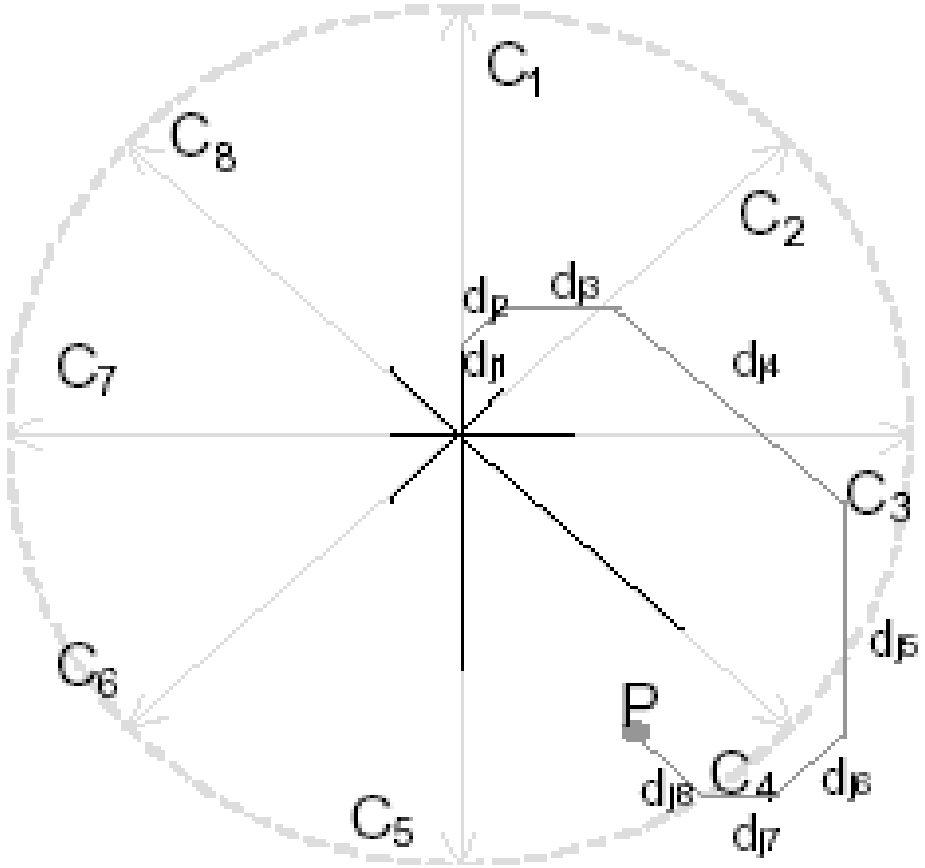


# Star Coordinates (Kandogan, 2001)

- Each dimension shown as an axis
- Data value in each dimension is represented as a vector.
- Data points are scaled to the length of the axis

  - min mapping to origin

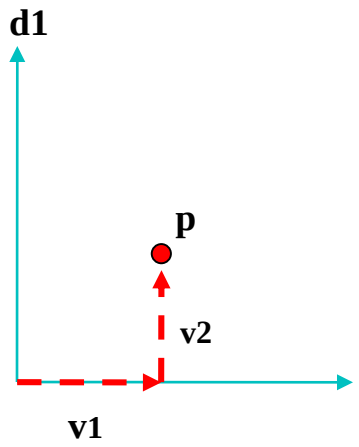
  - max mapping to the end



# Star Coordinates -1

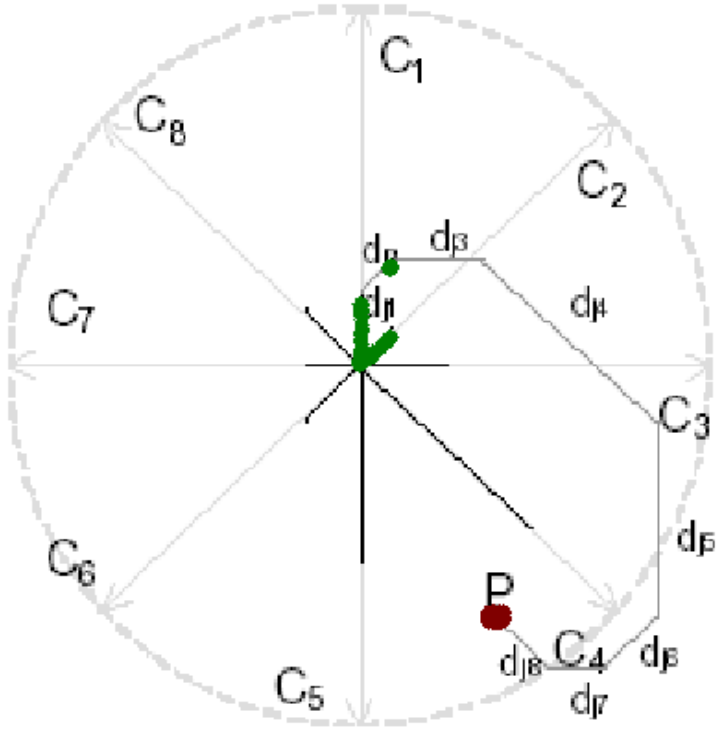
## Cartesian

**P=(v1, v2)**




## Star Coordinates

**P=(v1,v2,v3,v4,v5,v6,v7,v8)**




**Mapping:**

- **Items → dots**
- **Σ attribute vectors → position**

# Visualization using Dimensionality reduction

- Many modern data domains involve huge numbers of features / dimensions
  - Documents: thousands of words
  - Images: thousands to millions of pixels
  - Genomics: thousands of genes, millions of DNA polymorphisms

- Bees activity: one month of activity recording implies 720 hours of recording by bee(720 features) much more features if smaller time periods are considered.

# Why reduce dimensions?

- High dimensionality has many costs
  - Redundant and irrelevant features degrade performance of some ML algorithms
  - Difficulty in interpretation and visualization
  - Computation may become infeasible
  - Curse of dimensionality

# Approaches to dimensionality reduction

- Feature selection
  - Select subset of existing features (without modification)
- Model regularization (Lasso/Ridge regression)
- Map existing features into smaller number of new features
  - Linear combination (projection)
  - Nonlinear combination

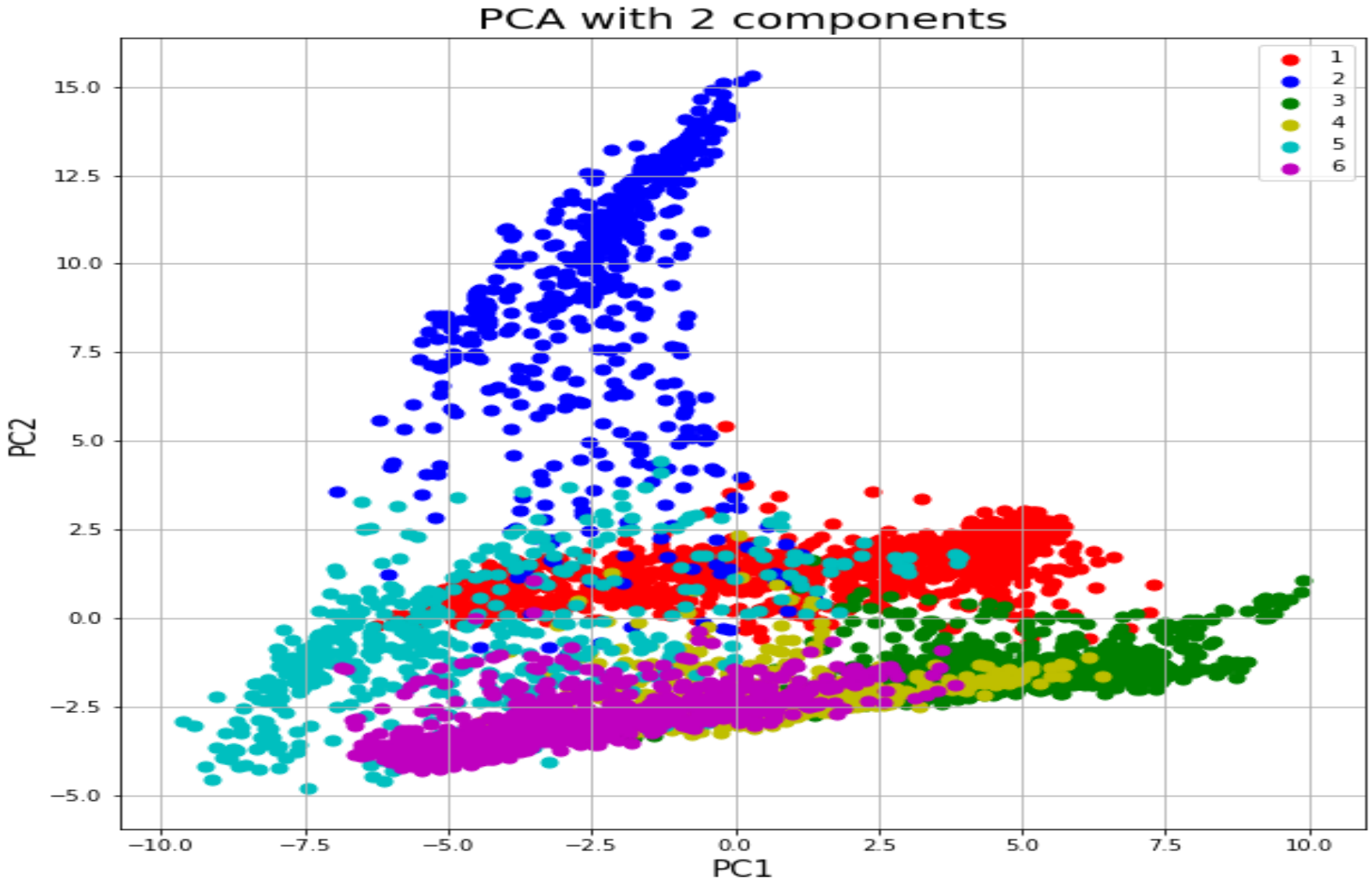
# Principal component analysis (PCA)

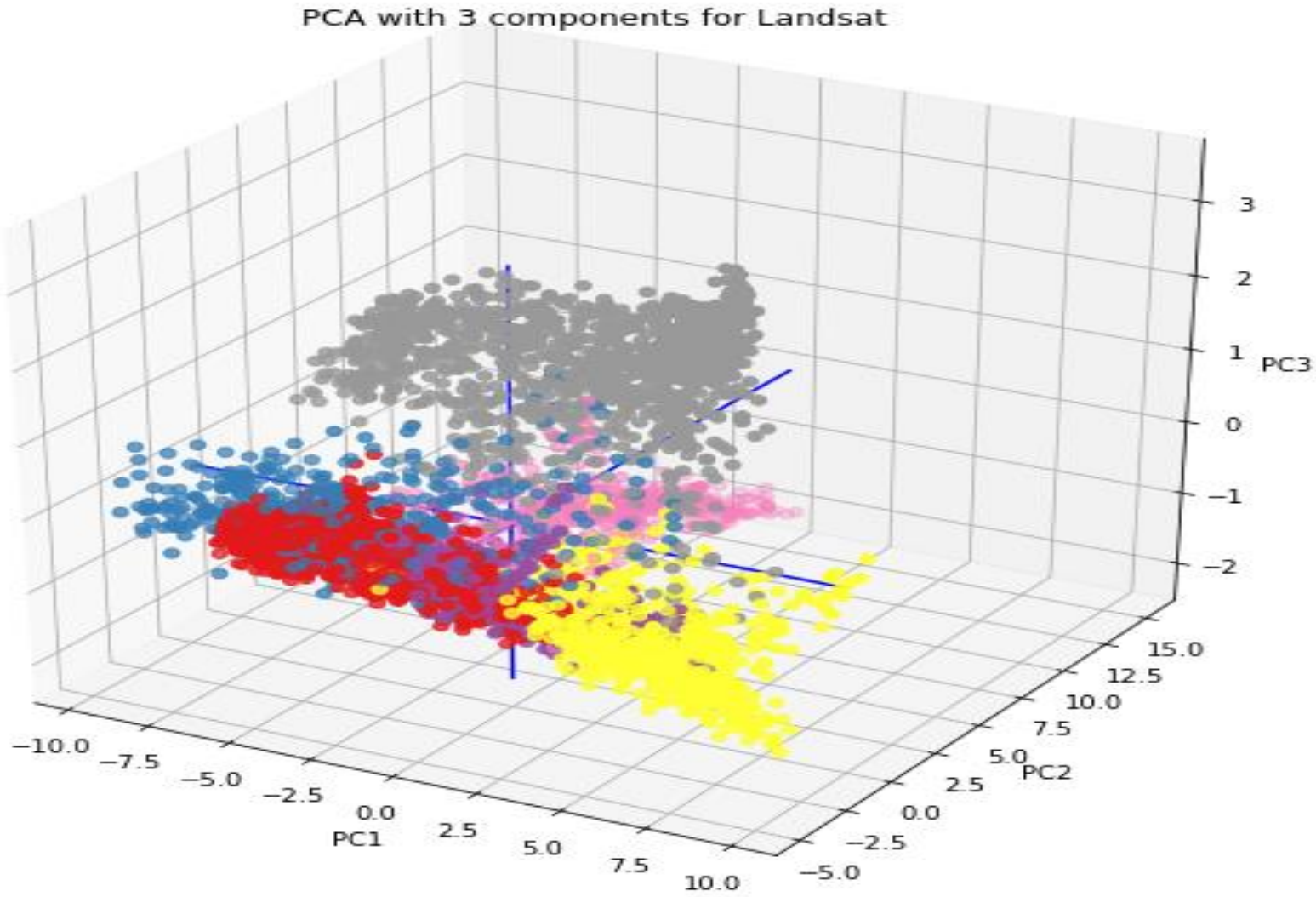
- Widely used method for unsupervised, linear dimensionality reduction
- GOAL: account for variance of data in as few dimensions as possible (using linear projection)

# PCA: Pros and cons

- Pros:
- Helps reduce computational complexity.
- Can help supervised learning.
- PCA can also be seen as noise reduction.
- Cons:
  - Fails when data consists of multiple separate clusters.
  - Directions of greatest variance may not be most informative.
- Practical issue: covariance matrix is n x n.
  - E.g. for image data $\Sigma$ = 32768 x 32768.
  - Finding eigenvectors of such a matrix is slow. Use SVD.

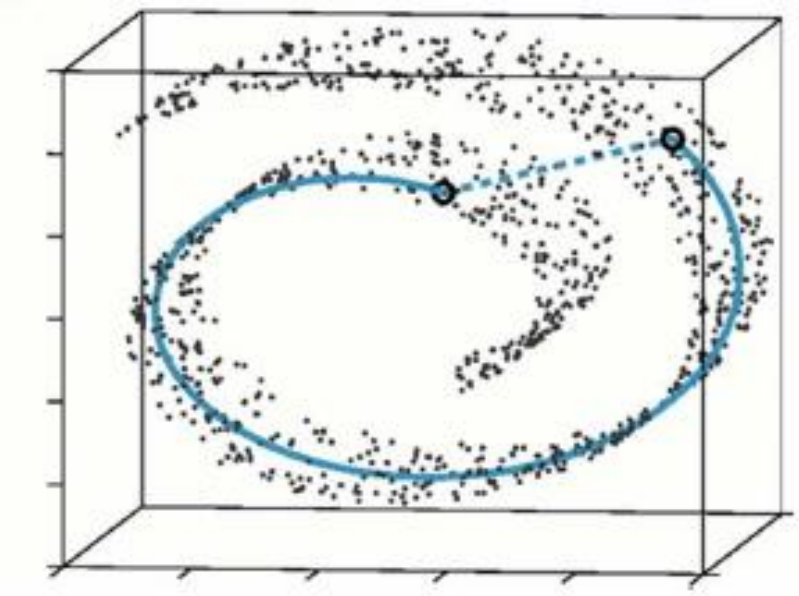
# PCA for Landsat (4435 instances, 36 features and 6 classes)




PCA with 3 components for Landsat
PC1
PC2
PC3
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
−2
−1
0
1
2
3

# Nonlinear dimensionality reduction

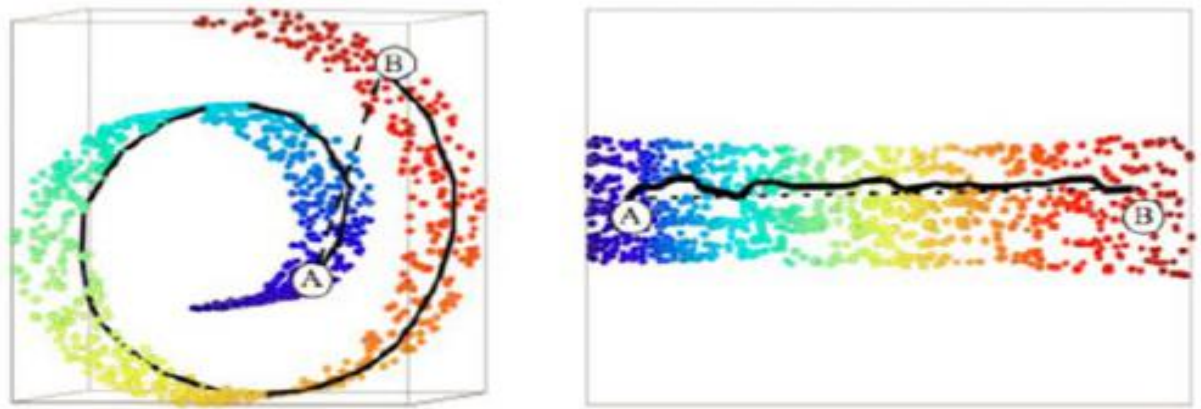
- Data often lies on or near a nonlinear low-dimensional surface
- Such low-dimensional surfaces are called *manifolds*.



Swiss roll data

# ISOMAP: Isometric Feature Mapping
## (Tenenbaum et al. 2000)

- A nonlinear method for dimensionality reduction
- Preserves the global, nonlinear geometry of the data by preserving the geodesic distances
- Geodesic: originally geodesic means the shortest route between two points on the surface of the manifold

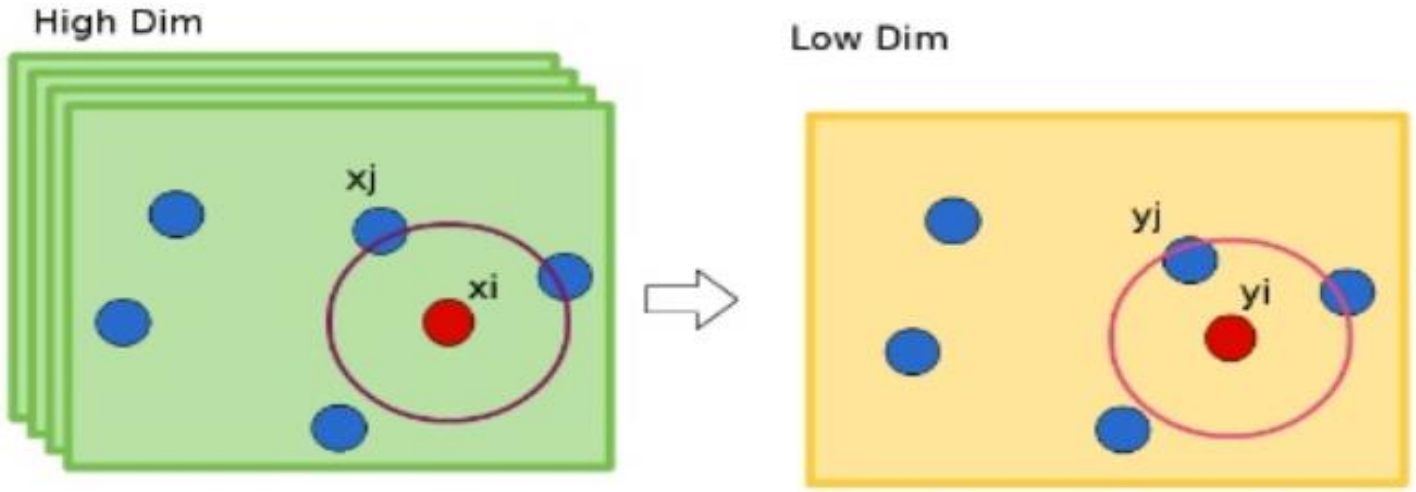


# ISOMAP

- Two steps
  1. Approximate the geodesic distance between every pair of points in the data
     - The manifold is locally linear
     - Euclidean distance works well for points that are close enough
     - For the points that are far apart, their geodesic distance can be approximated by summing up local Euclidean distances
  2. Find a Euclidean mapping of the data that preserves the geodesic distance

# t-Stochastic neighbor embedding (t-SNE) (van der Maaten and Hinton, 2008)

Measure pairwise similarities between high-dimensional and low-dimensonal objects




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

Figure taken from K. Zhao's presentation (10/2014)

# t-Stochastic neighbor embedding (t-SNE)

Converting the high-dimensional Euclidean distances into conditional probabilities that represent similarities

- Similarity of datapoints in High Dimension

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

- Similarity of datapoints in Low Dimension

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

- Cost function

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

Minimize the cost function using gradient descent

# t-Stochastic neighbor embedding (t-SNE)

Use heavier tail distribution than Gaussian in low-dim space, we choose

$$q_{ij} \propto (1 + ||y_i - y_j||^2)^{-1}$$

Then the gradient could be

$$\frac{\partial C}{\partial y_i} = 4 \sum_{j \neq i} (p_{ij} - q_{ij})(1 + ||y_i - y_j||^2)^{-1}(y_i - y_j)$$

# t-Stochastic neighbor embedding (t-SNE)

The t distribution is used to approximate probabilities $q_{i/}$
An important feature of t-SNE is a tuneable parameter, “perplexity,” which says (loosely) how to balance attention between local and global aspects of your data. The parameter is, in a sense, a guess about the number of close neighbors each point has. The perplexity value has a complex effect on the resulting pictures.

In the original paper is stated that *“The performance of t-SNE is fairly robust to changes in the perplexity, and typical values are between 5 and 50.”*

The t-SNE algorithm doesn’t always produce similar output on successive runs, for example, and there are additional hyperparameters related to the optimization process.

j

# Effect of the perplexity parameter

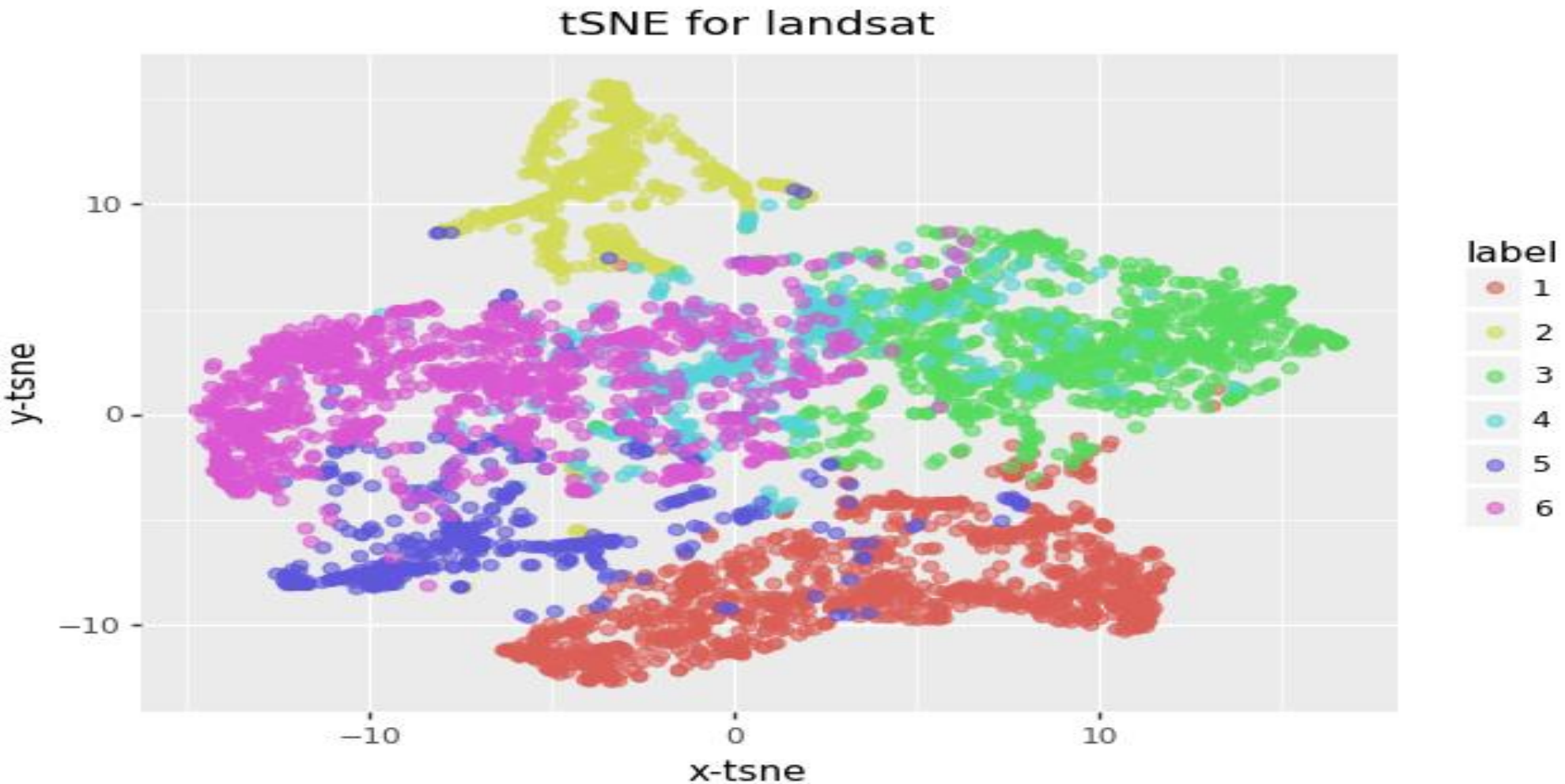
*Original*

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000


tSNE for landsat
y-tsne
10
0
−10
−10
0
10
x-tsne
label
1
2
3
4
5
6

# Example: The MNIST dataset

It was used by LeCun, Cortes and Burges (1998) for handwritten recognition.
The training dataset consists of 60k images of the digits 0 to 9 and the test dataset consists of 10k images.
Each row of the datasets has 785 entries containing 784 pixels (28x28) and the first entry is the digit label.
The training dataset is 104MB and the test is 17MB.

# Visualization of classes in MNIST data using PCA

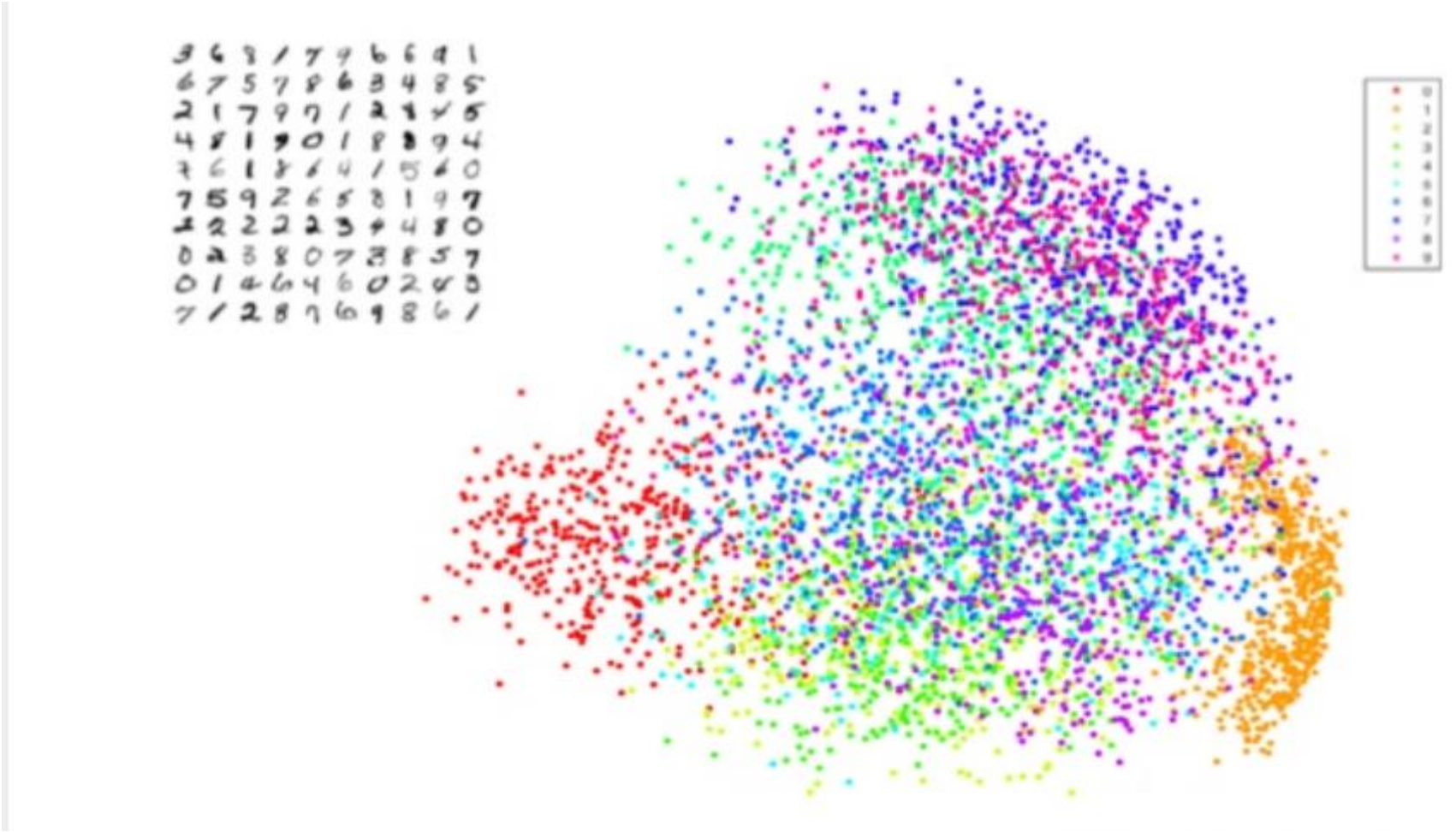
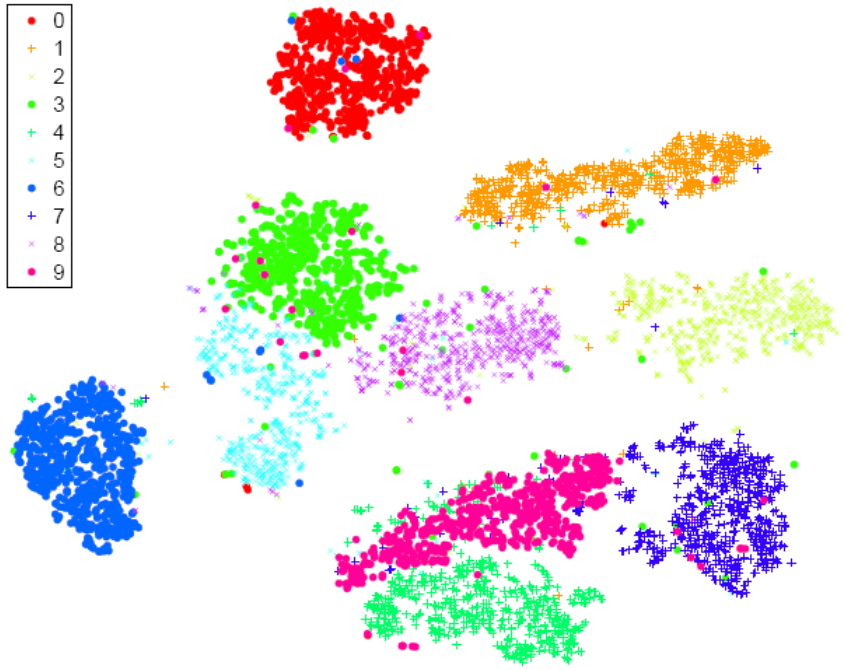
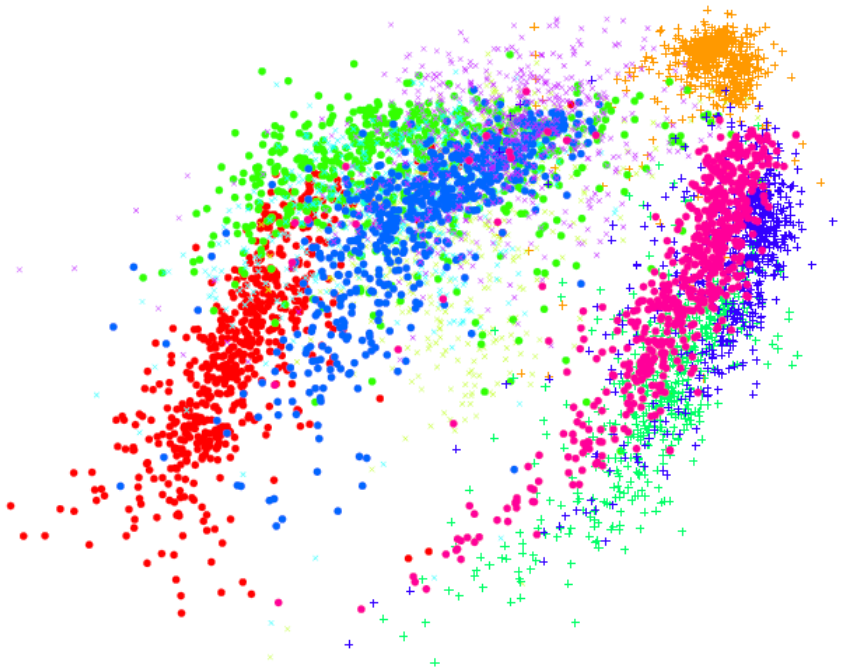


**Figure taken from K. Zhao's presentation (10/2014)**

# Visualization of classes in MNIST data




t-SNE



ISOMAP

# Disadvantages of t-SNE

t-SNE has a quadratic time and space complexity in the number of data points. This makes it particularly slow and resource draining while applying it to data sets comprising of more than 10,000 observations.

In order to reduce the time complexity the creators of t-SNE recommend to apply first PCA to reduce the number of dimension, say to 50, and after that apply t-SNE.
In the paper, "**Mapping the stereotyped behaviour of freely moving fruit flies" by** Berman GJ, Choi DM, Bialek W, Shaevitz JW, (2014), the authors apply PCA to reduce a 40,000 pixels images to a 50 dimension feature space and after that they apply t-SNE.

# Running PCA and tSNE in Tensorboard