Unsupervised learning introduction

2 July 2019, DESY

Andrey Ustyuzhanin

NRU HSE

YSDA

ICL

Overview

- **Anomaly/novelty detection**
  - Density, Nearest Neighbor, Trees
- **Dimensionality reduction**
  - PCA
  - T-SNE
- **Clustering techniques**
  - K-means
  - DBSCAN (density based)

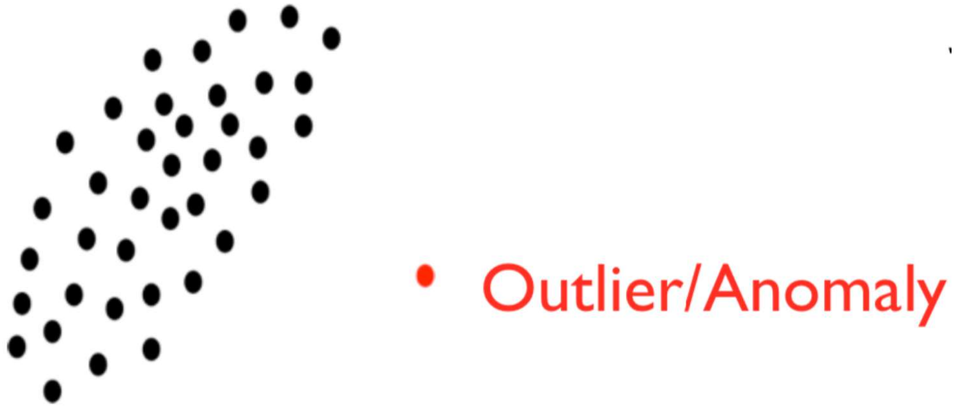
Anomaly detection

“An outlier is an observation in a data set which appears to be inconsistent with the remainder of that set of data.”

Johnson 1992

“An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.”

Hawkins 1980



Types of AD

Supervised AD

> Labels available for both normal data and anomalies
> Similar to rare class mining / imbalanced classification

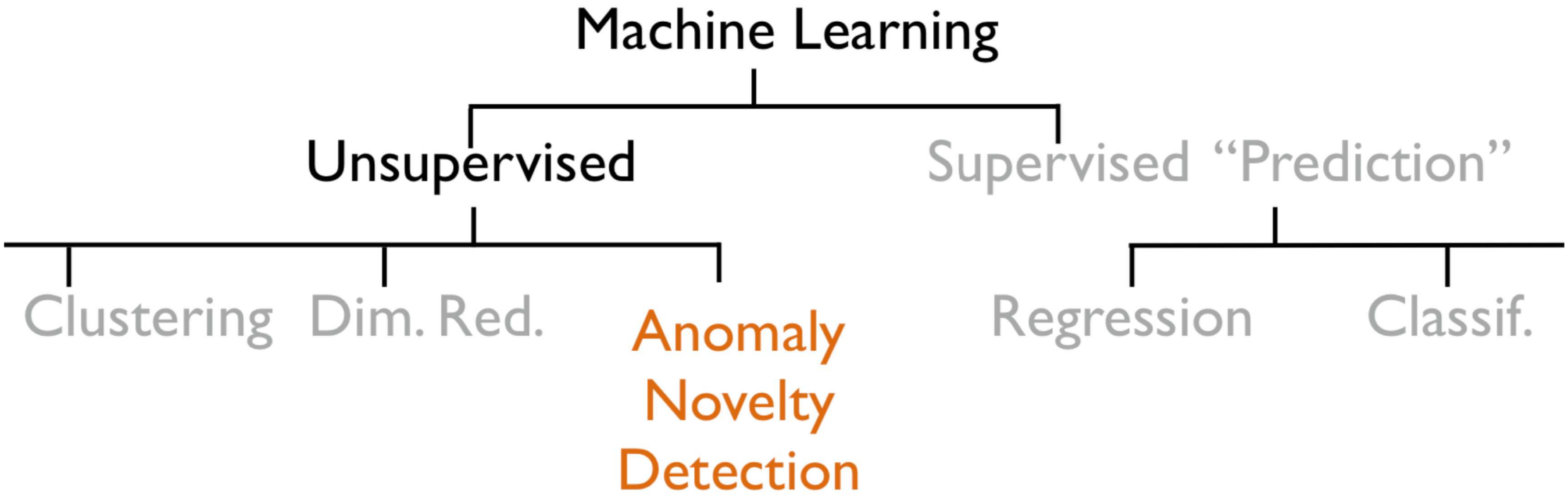
Semi-supervised AD (Novelty Detection)

> Only normal data available to train
> The algorithm learns on normal data only

Unsupervised AD (Outlier Detection)

> no labels, training set = normal + abnormal data
> Assumption: anomalies are very rare

ML Taxonomy


Machine Learning
Unsupervised
Supervised "Prediction"
Clustering
Dim. Red.
Anomaly
Novelty
Detection
Regression
Classif.

Applications

Fraud detection
Network intrusion
Finance
Insurance
Maintenance
Medicine (unusual symptoms)
Measurement errors (from sensors)

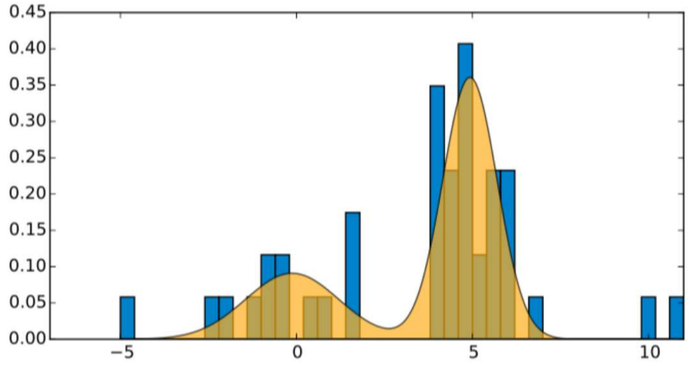
Any application where looking at unusual observations is relevant!

Basic Idea

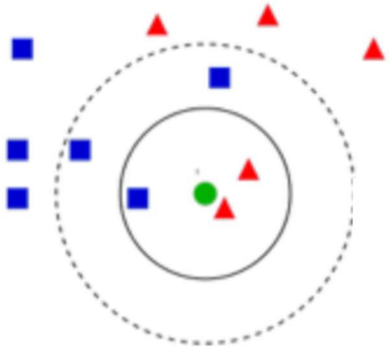
- Look for samples that are in low density regions, isolated
- Look for a region of the space that is small in volume but contains most of the samples

Possible approaches

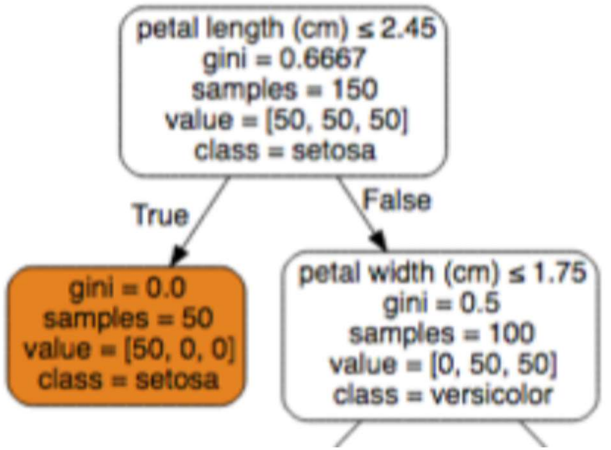
Density based approach (KDE, Gaussian Ellipse, GMM)

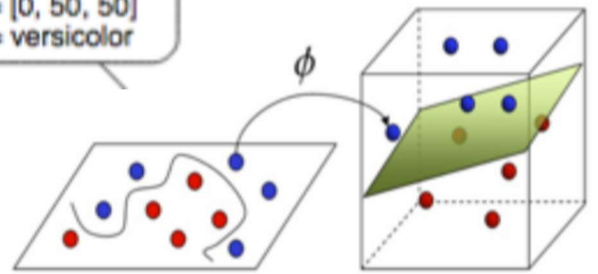


Nearest neighbors



Trees / Partitioning

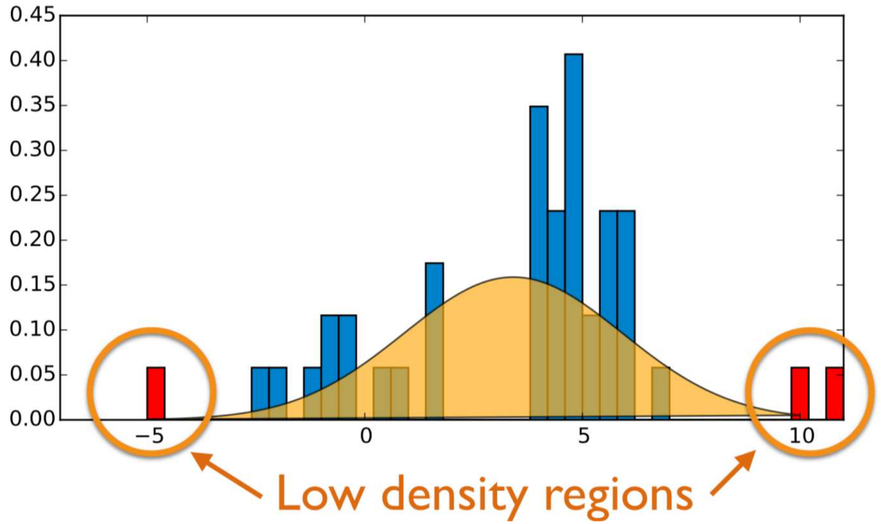




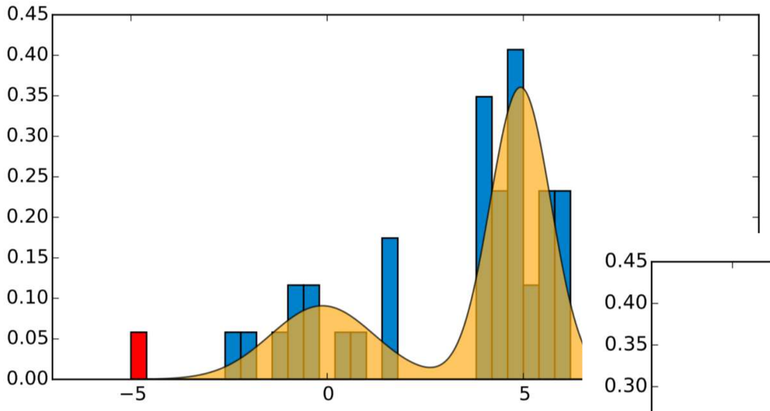
Others (e.g. Kernel methods)

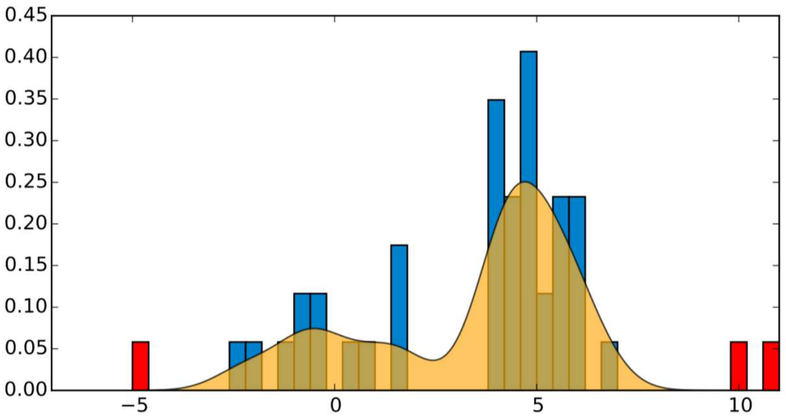
Density-based methods

`GaussianMixture(n_components=1)`



`GaussianMixture(n_components=2)`

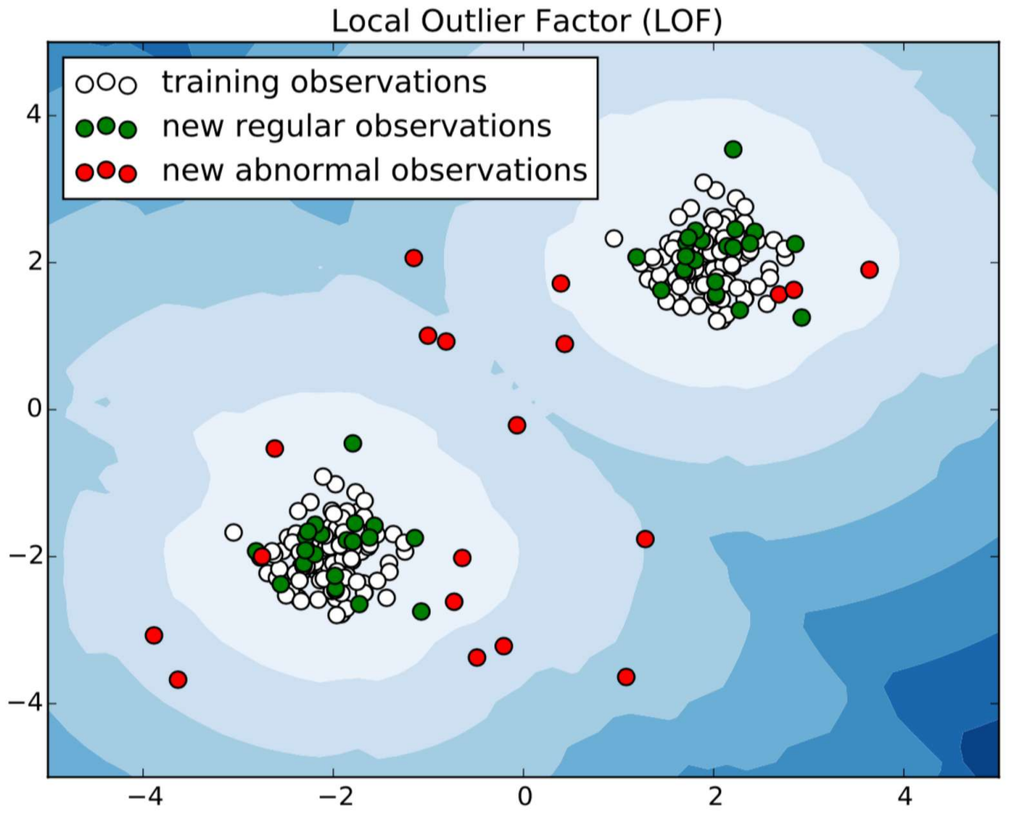




`KernelDensity(kernel='gaussian', bandwidth=0.75)`

Nearest Neighbors (NN) Approach

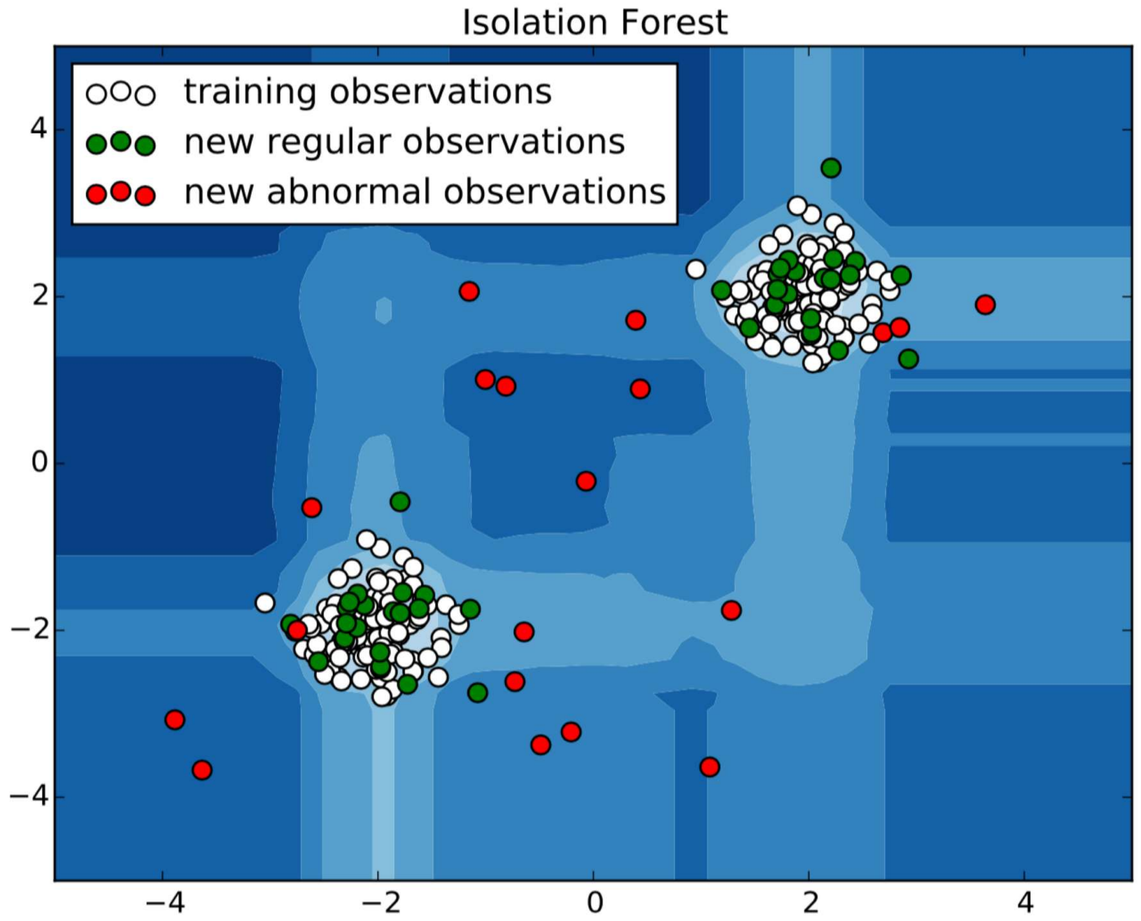
LocalOutlierFactor - The anomaly score of each sample is called Local Outlier Factor. It measures the local deviation of density of a given sample with respect to its neighbors. It is local in that the anomaly score depends on how isolated the object is with respect to the surrounding neighborhood.



https://en.wikipedia.org/wiki/Local_outlier_factor

Partitioning / Tree approach

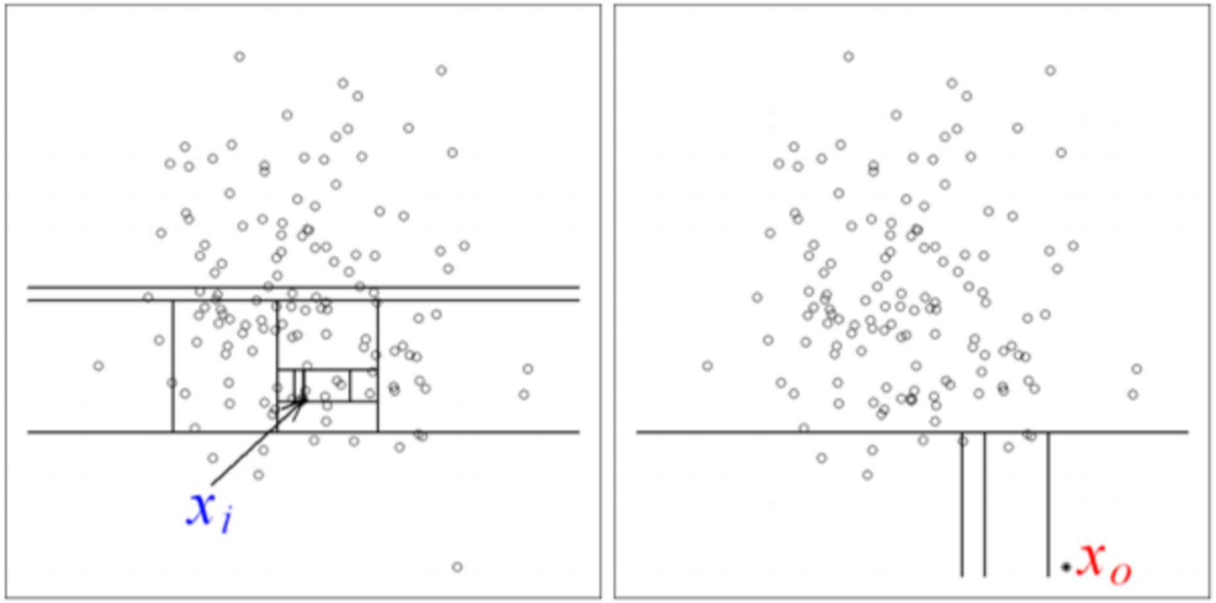
The IsolationForest ‘isolates’ observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature.



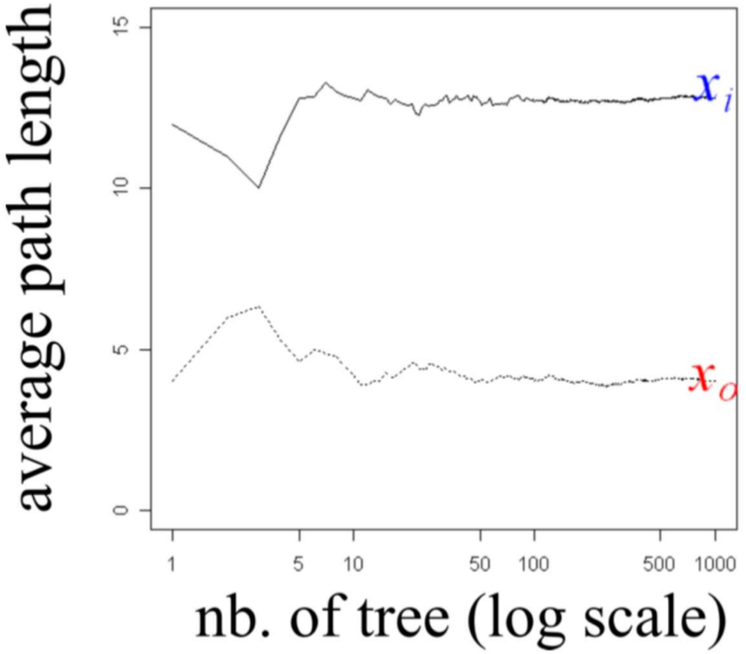
http://scikit-learn.org/dev/modules/generated/sklearn.ensemble.IsolationForest.html

Partitioning / Tree approach

Random partitioning produces noticeably shorter paths for anomalies. Hence, when a forest of random trees collectively produce shorter path lengths for particular samples, they are highly likely to be anomalies.

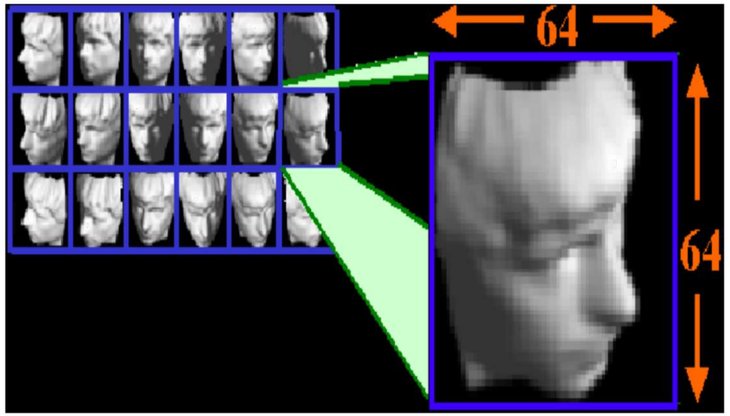


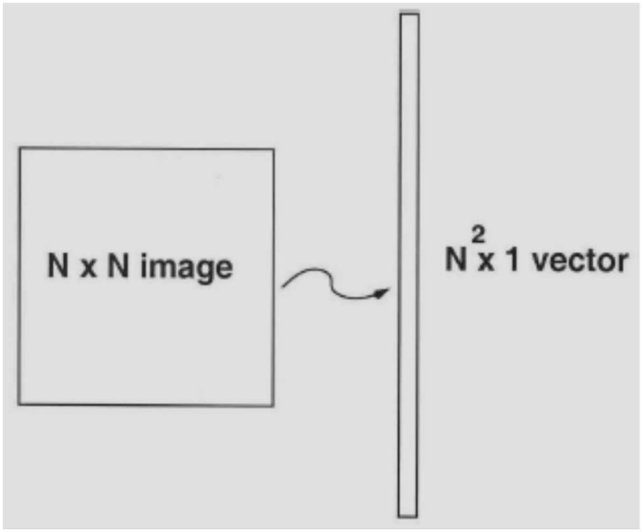
(a) Isolating x_i (b) Isolating x_o

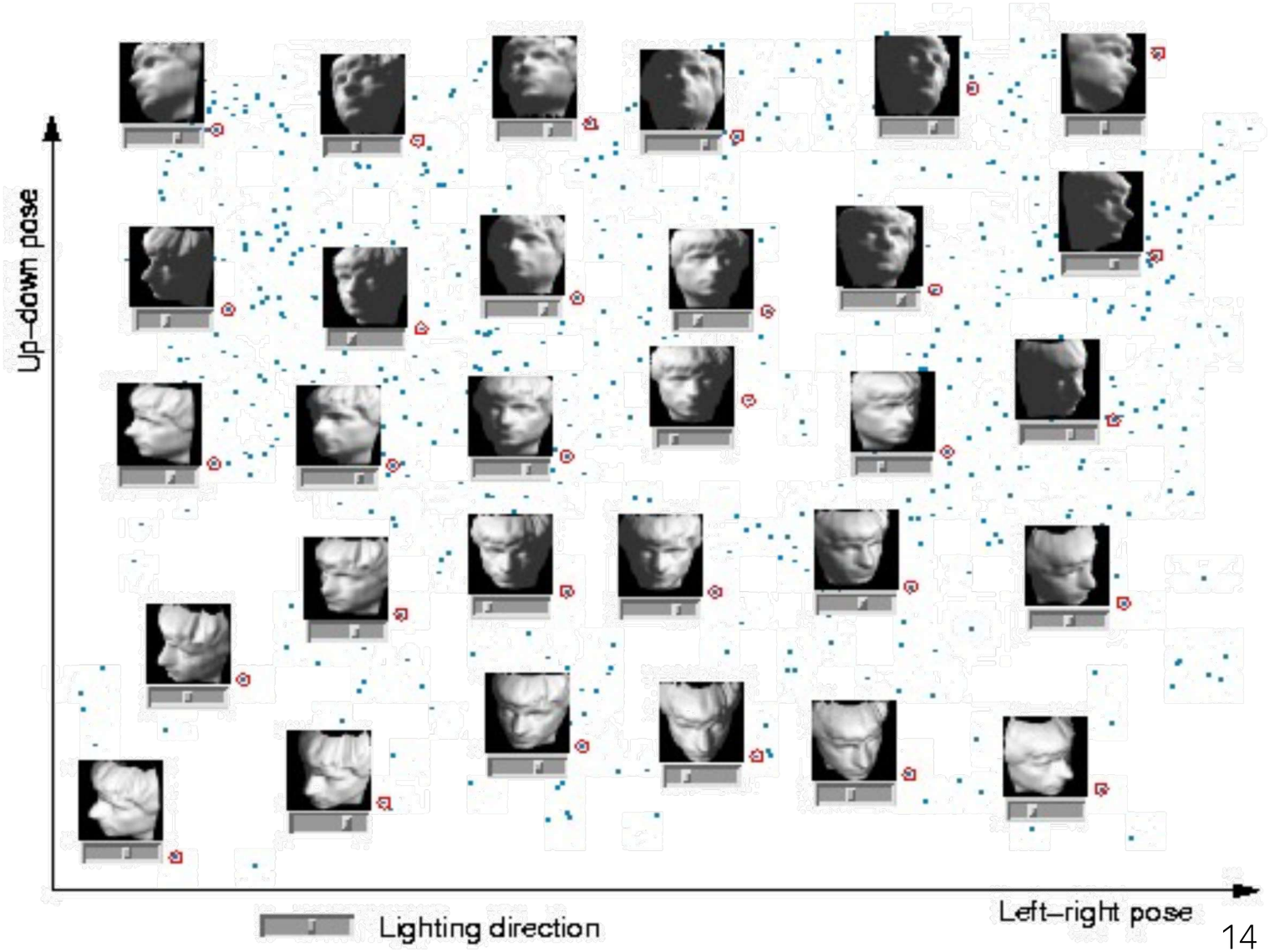


Dimensionality Reduction

Faces

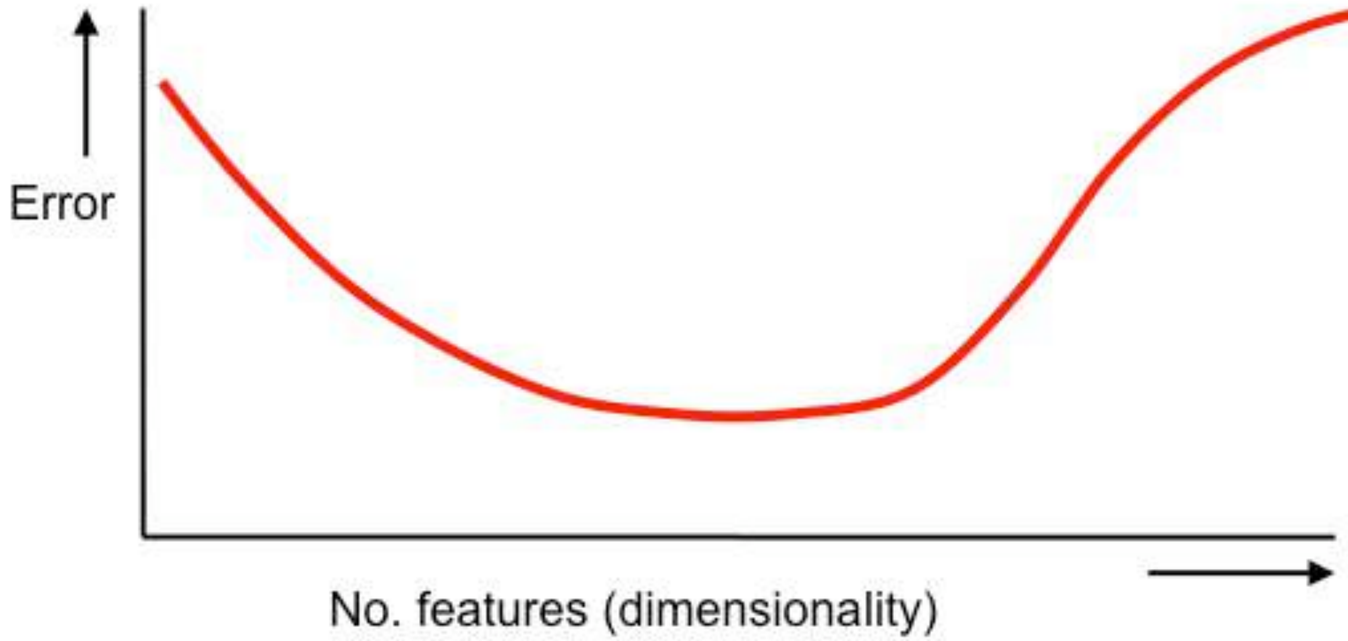

64
64


N x N image
N x 1 vector
2


Up–down pose
Left–right pose
Lighting direction

Motivation: curse of dimensionality, errors

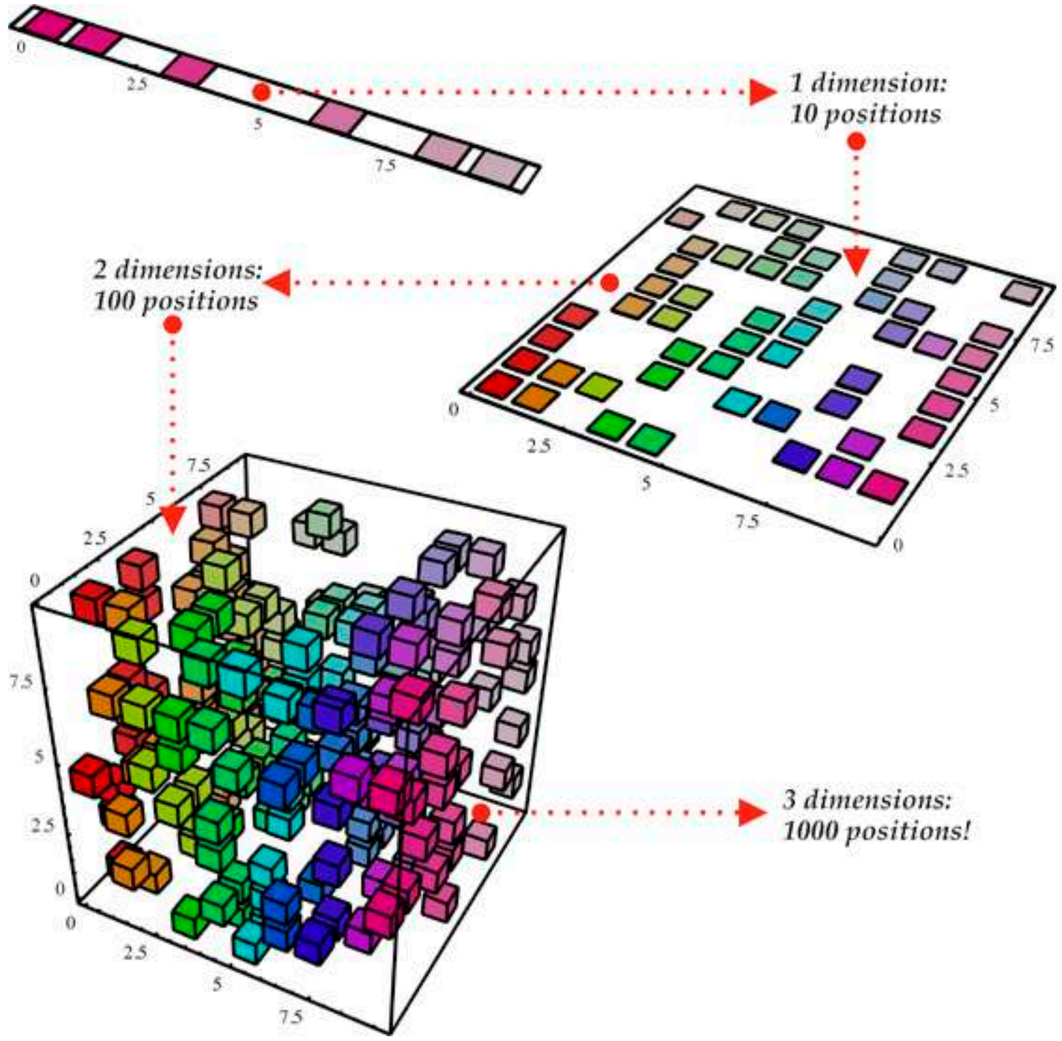
- Too low dimensionality may lead to high variance
- Too high dimensionality may lead to sparsity and high bias
- Classification tasks are harder in higher dimensions



Motivation: curse of dimensionality, sparsity

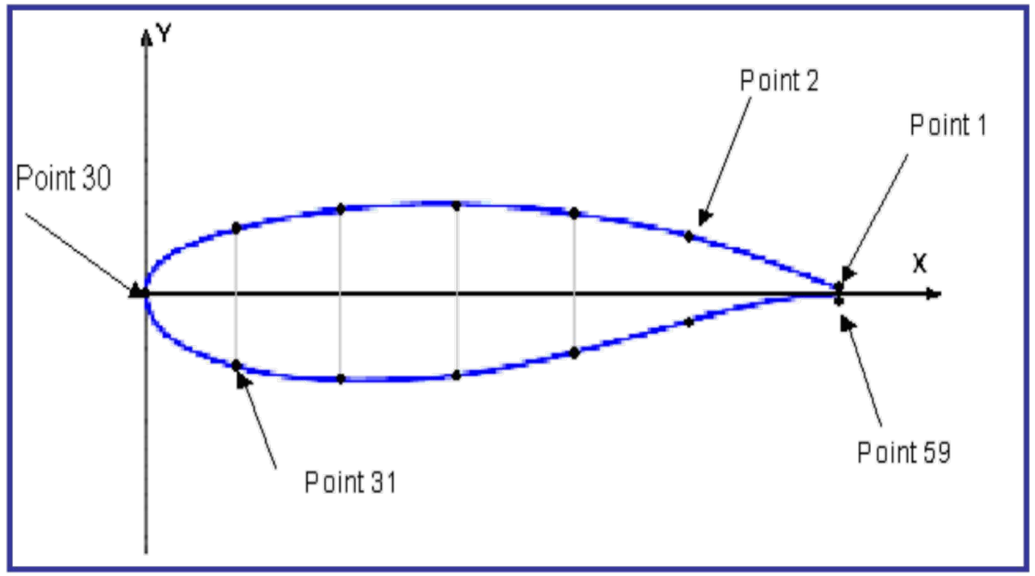
To cover 10% of N-dimensional (100-cube) volume you have to provide:

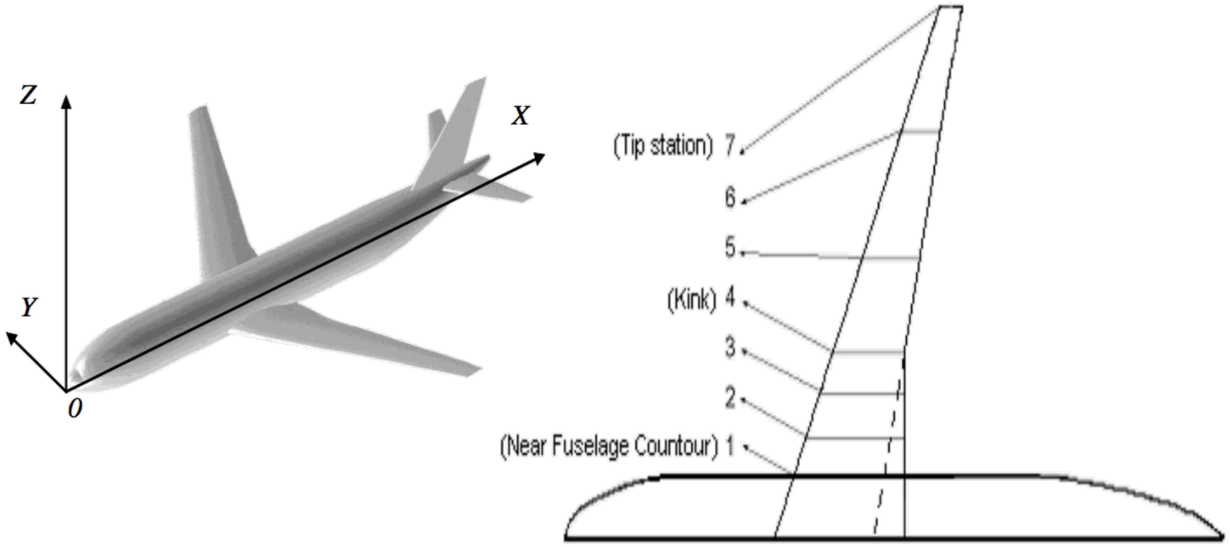
- 10 samples (1D)
- 100 samples (2D)
- 1000 samples (3D)
- ...



Airfoil parametrization

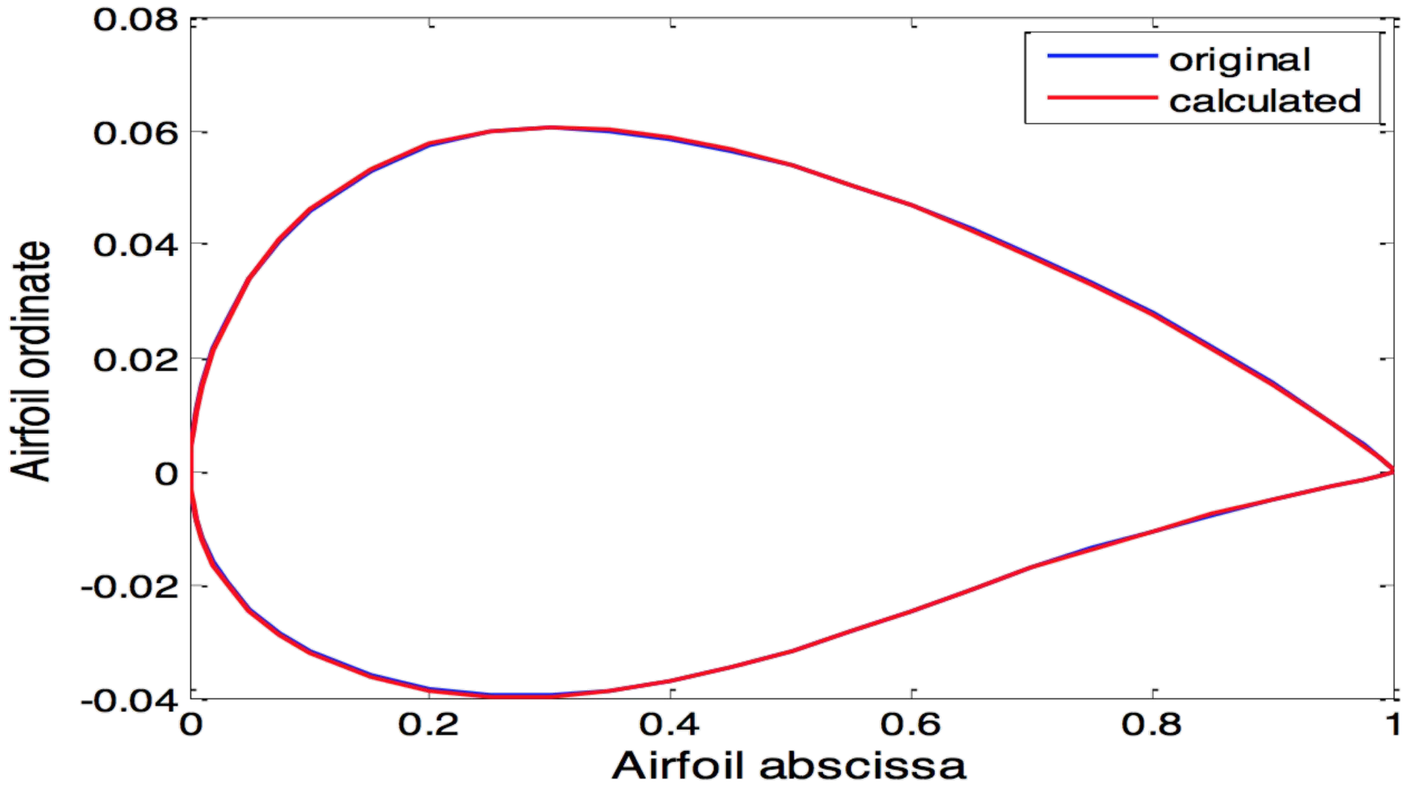
- Aircraft wing can be described by 7 cuts:
  - Each cut takes 59 points to describe 2D shape
- If we could find a better parametrization without losing precision?



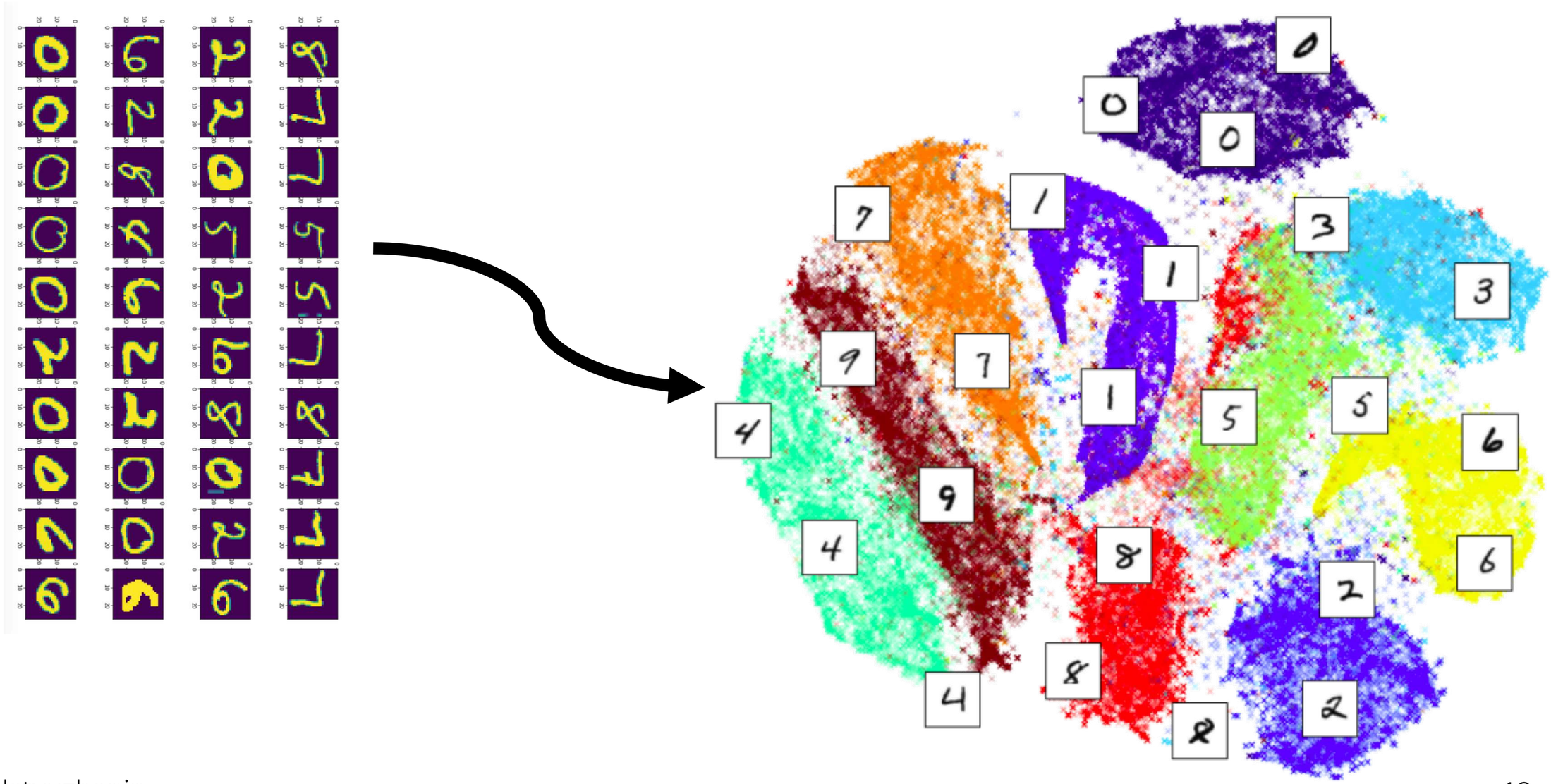


Airfoil parametrization

Blue: original shape (59 points)
Red: 6-parametrization



MNIST dimensionality reduction to 2



Particle Physics examples

Multidimensional objects:

- Hits: Jets, Showers (represented by images)
- Tracks

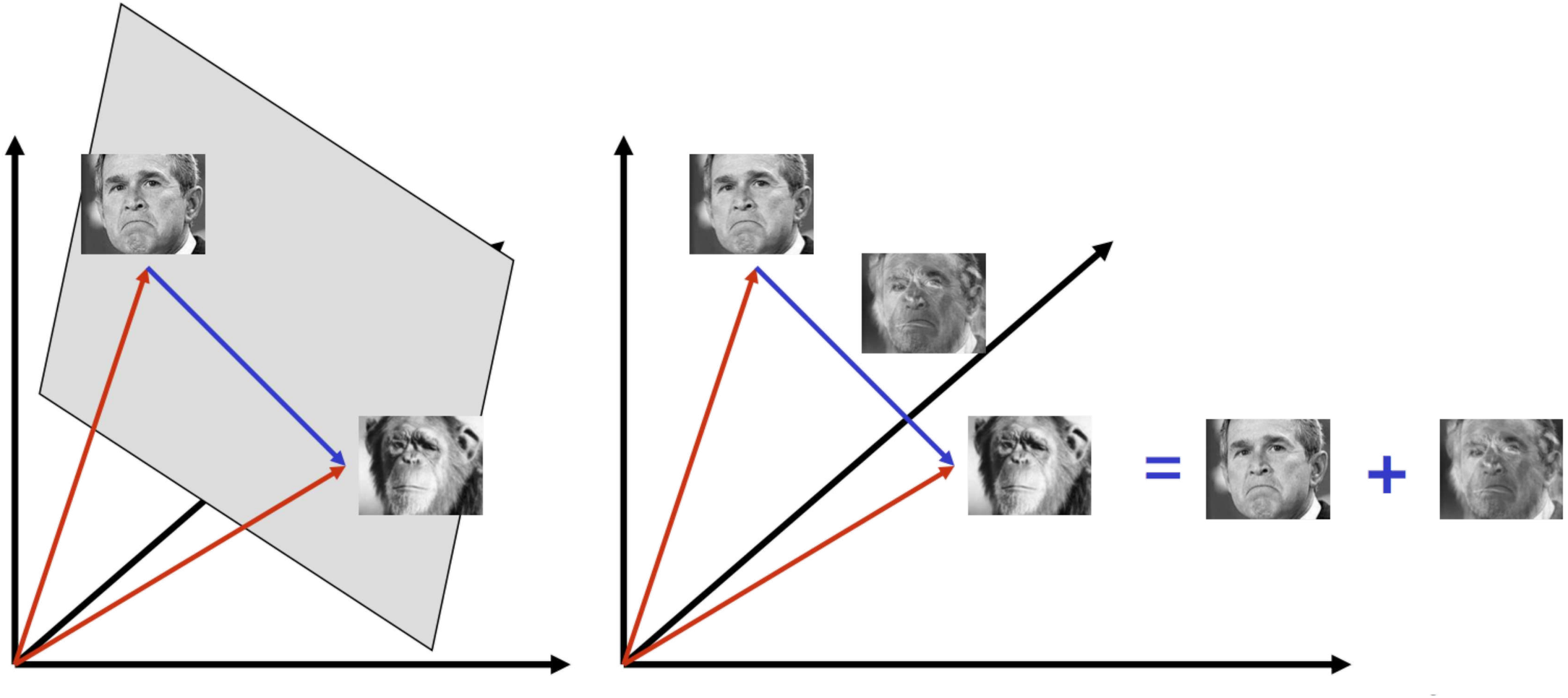
Complex objects

- Events (track collection)
- Trigger/stripping lines
- Collection of subdetector high-level responses for PID, trigger, etc

Applications

- Improve optimization stability by removing multicollinearity (helpful to some models)
- Data Compression (improve operational performance)
  - CPU, Memory, IO, Disk
- Data Visualization
- Feature extraction (meaningful representation)
  - Make classifier more interpretable
  - Vector operations
- (potentially) Increase predictive accuracy of a classifier

Applications. Vector operations



Basic Problem Statement

- given set of objects O_i, described by features $X(O_i)$ from R^p
- Find compressed representation $y(O_i)$ from R^q, $q < p$
- Provided no significant loss of information about O_i

Methods

Linear

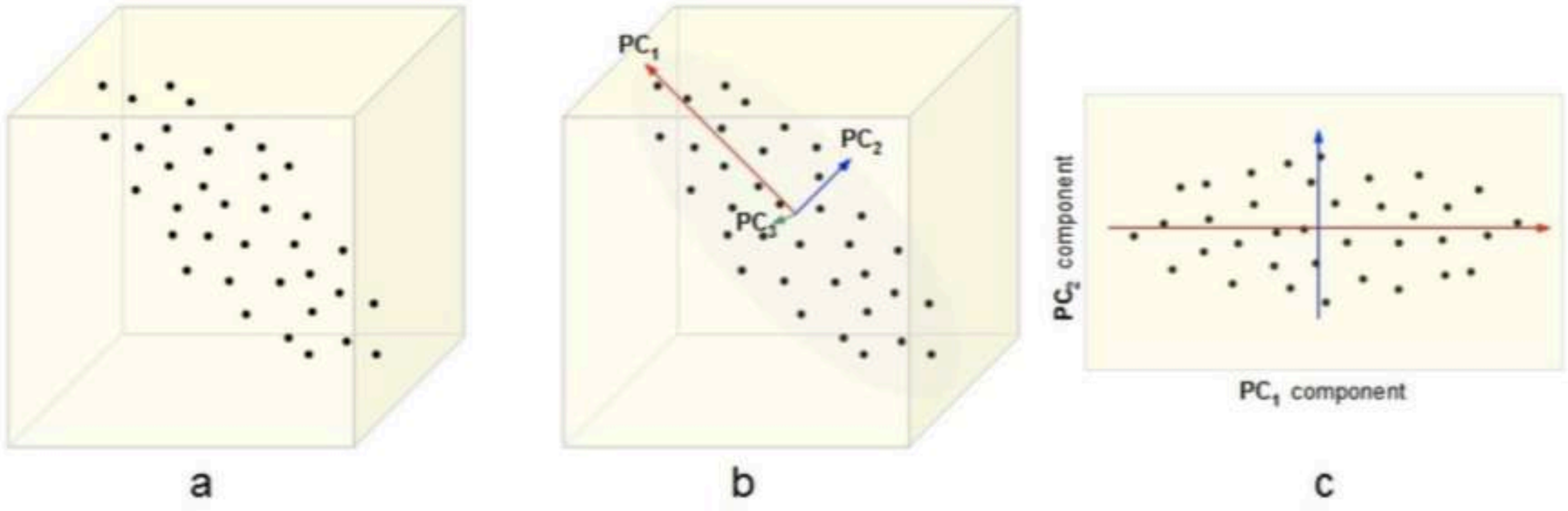
- **Principle Component Analysis (PCA)**
  - Should be run on scaled data
  - Relatively easy to interpret
  - Slow to calculate without SVD hacks (randomized, truncated)
- **Other methods examples**
  - SVD, Non-negative matrix factorization, Pursuit Projection, Multi Dimensional Scaling

Non-linear

- **T-SNE**

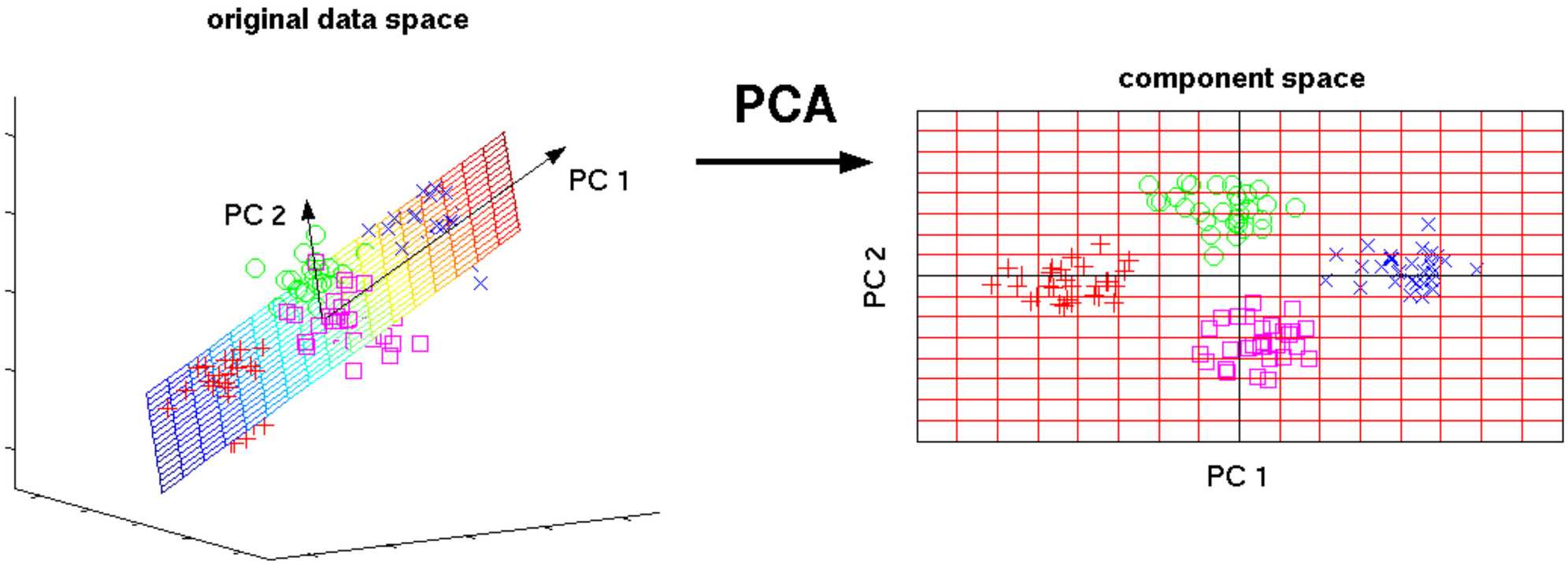
Principle Component Analysis

Intuition 1: Find directions along which our datapoints have the greatest variance



Principle Component Analysis

Intuition 2: Find a subspace L (of lesser dimension) s.t. the sum of squares of differences between points and their projections is minimized

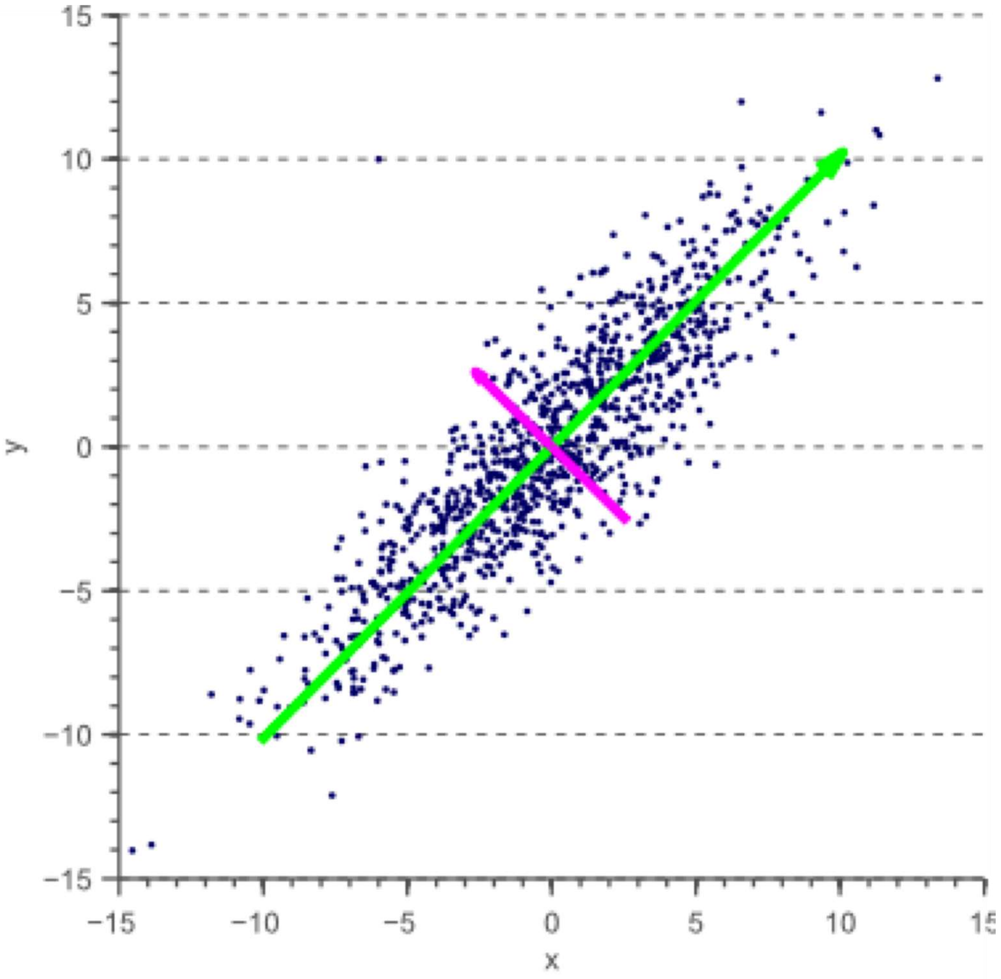


Principle Component Analysis is

A transformation of your initial feature axes ...

- New axes are just a linear combination of initial axes
- New axes are orthogonal (orthonormal) to each other
- Variance of data across those axes is maximized

... which keeps only the most "informative" axes



PCA Algorithm

1. Center (and scale) dataset X
2. Calculate covariance matrix $C=X^TX$
3. Find first k eigenvalues λ_i and eigenvectors a_i

$$A = \begin{bmatrix} | & | & & | \\ a_1 & a_2 & \dots & a_k \\ | & | & & | \end{bmatrix}$$

4. Perform projection to that space ($Z = XA$)

https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf

Explained variance

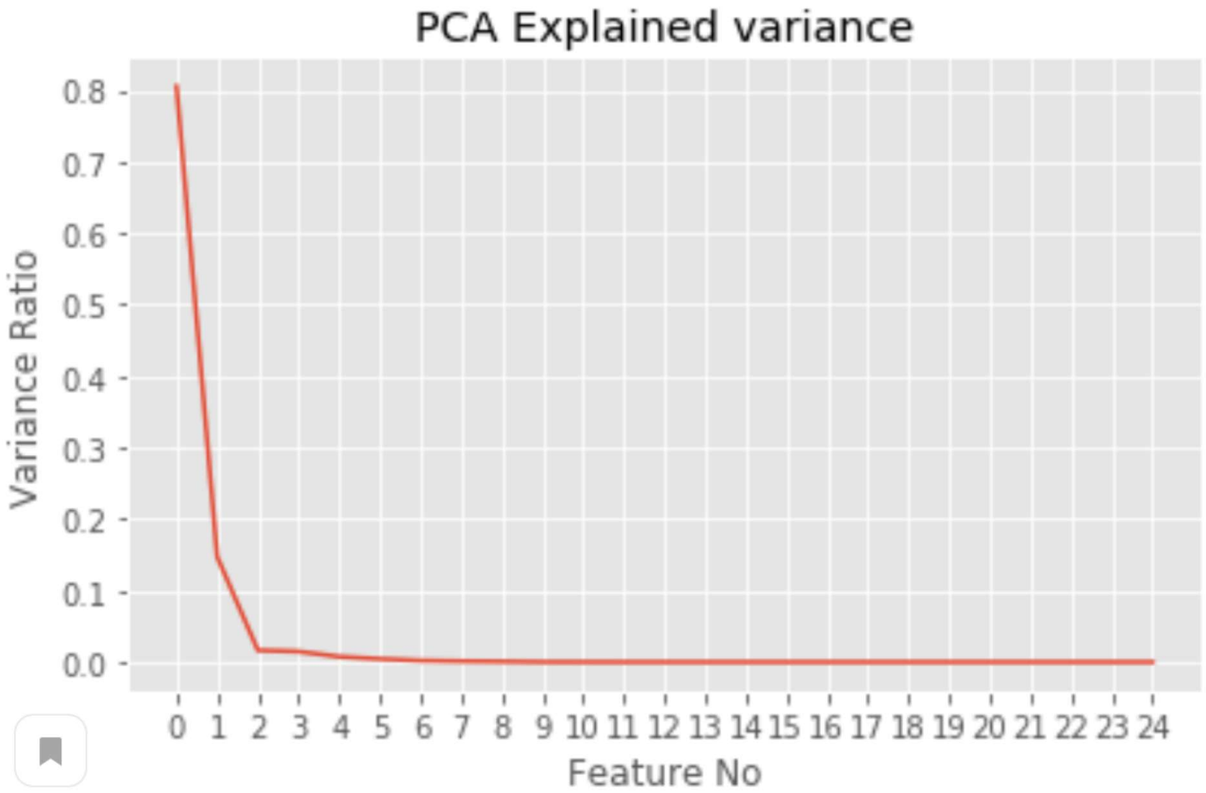
Since

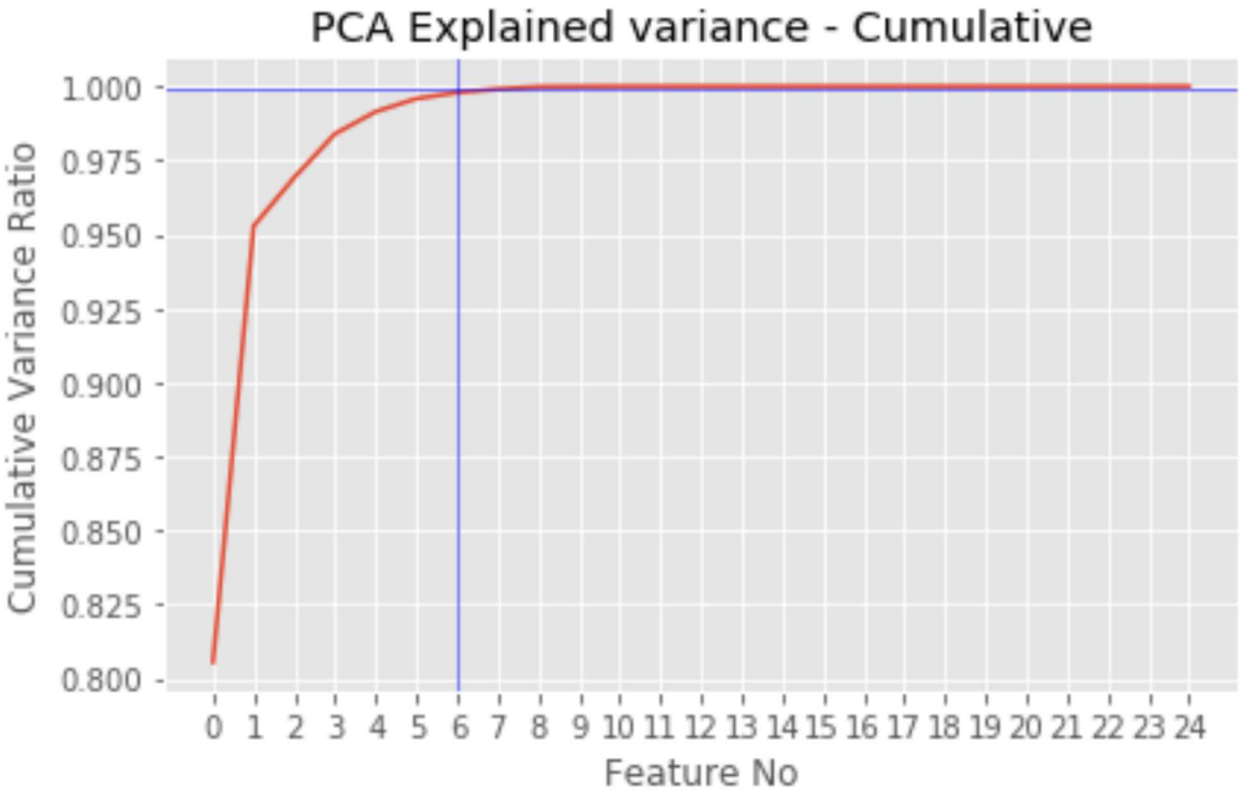
$$a_i{}^T X^T X a_i = \lambda_i,$$

That means λ_i corresponds to the variance of data "explained" by a_i

We can calculate explained variance ratio for a_i $\dfrac{\lambda_i}{\sum_{d=1}^{D} \lambda_d}$

Explained variance

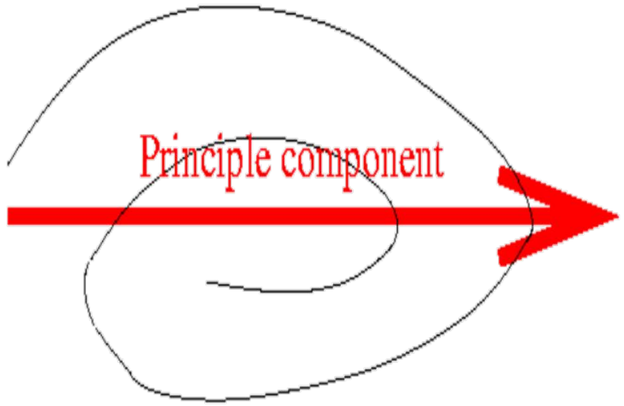




https://www.cs.princeton.edu/picasso/mats/PCA-Tutorial-Intuition_jp.pdf

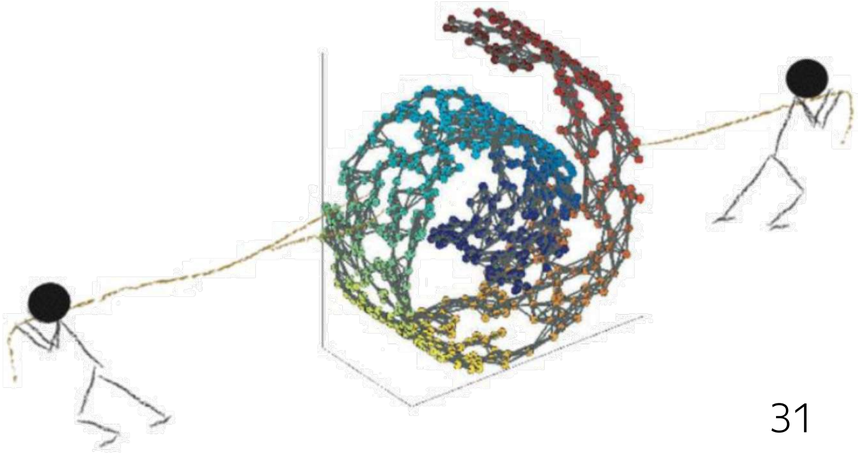
Non-linear

Find feature space with lesser dimensions s.t. **distances** in initial space are conserved in the new one. A bit more formally:

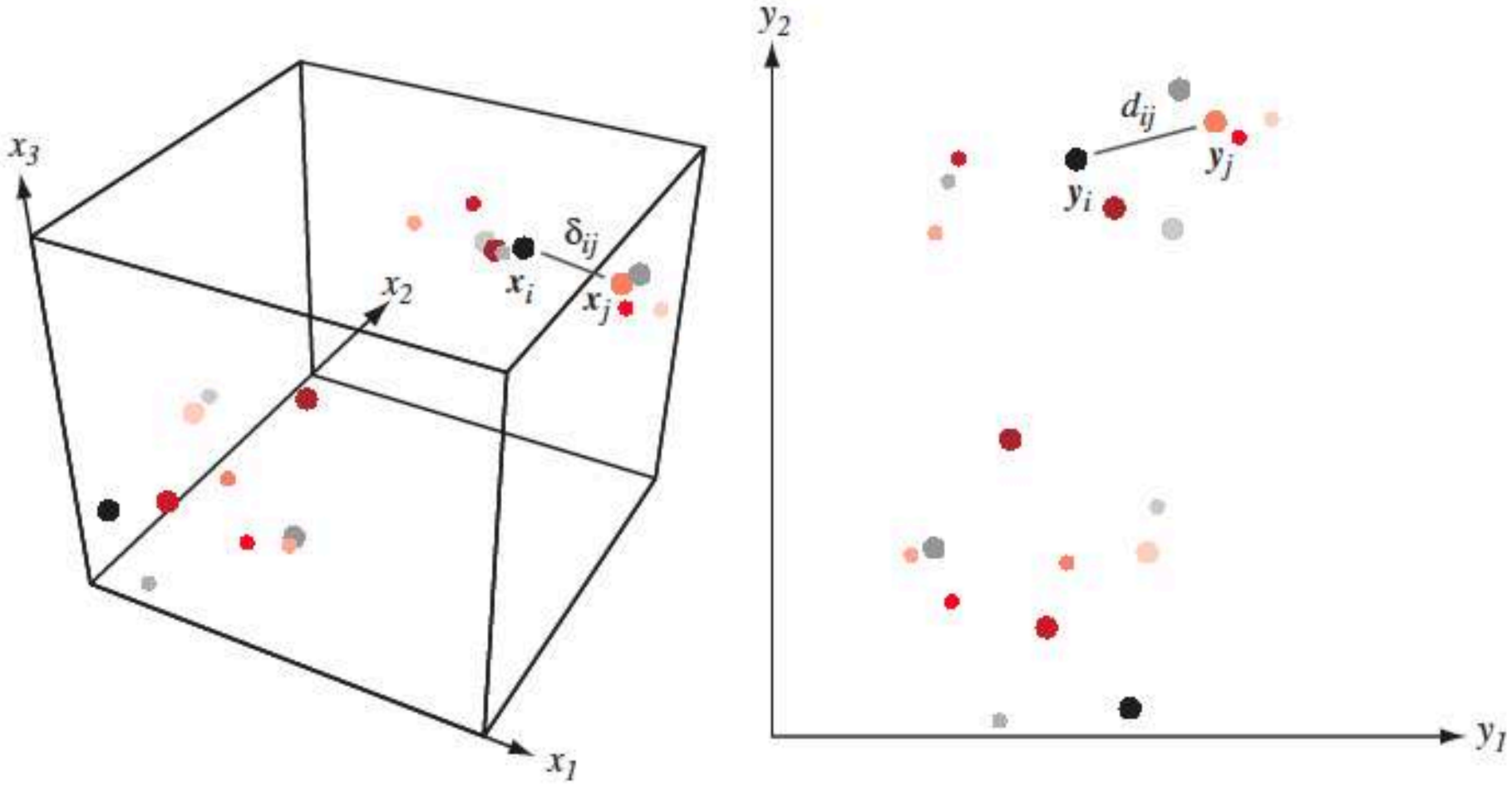


Given $\mathbf{X}=[\mathrm{x}_1,\ldots,\mathrm{x}_\mathrm{n}]\in\mathbb{R}^{\mathrm{N}\times\mathrm{D}}$ and/or δ_{ij} - proximity measure between (x_i,x_j) in initial feature space.

Find $\mathbf{Y}=[\mathrm{x}_1,\ldots,\mathrm{x}_\mathrm{n}]\in\mathbb{R}^{\mathrm{N}\times\mathrm{d}}$ such that $\delta_{ij}\approx d(y_i,y_j)=\|y_i-y_j\|^2$

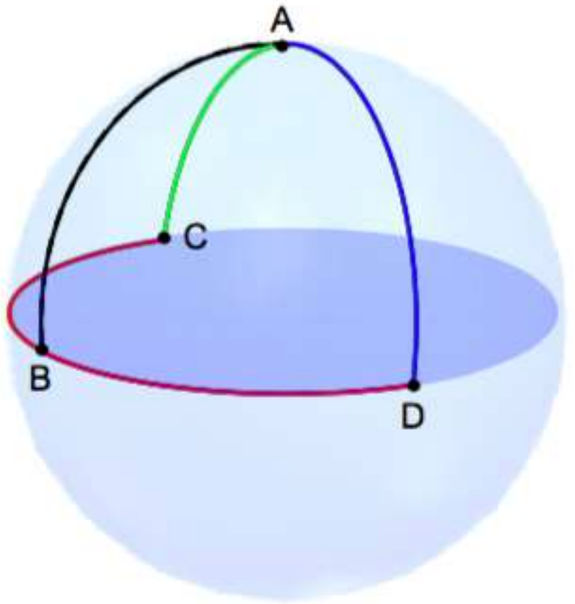


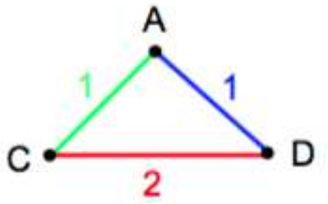
Multidimensional scaling - intuition

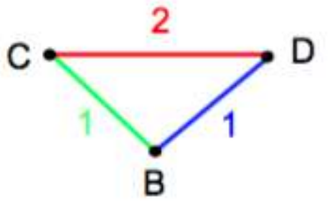


Multidimensional scaling

It is clear, that most of the times distances won't be conserved completely







What if we want to conserve not the distances themselves, but the structure of initial dataset. [illegible] neighborhood of each point?

Stochastic Neighbor Embedding

Converts the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities

t-distributed stochastic neighbor embedding

For set of points in a high-dimensional space find a faithful representation of those points in a lower-dimensional (2D) space.

Faithful means it preserve 3 types of similarities:

> **Similarity between points in initial feature space**
> **Similarity between points in derived feature space**
> **Similarity between feature spaces!**

The algorithm is non-linear and adapts to the underlying data, performing different transformations on different regions. Those differences can be a major source of confusion.

Similarities

- Similarity between points in initial feature space $\mathbb{R}^D$

$$p(i,j) = \frac{p(i|j) + p(j|i)}{2N}, \quad p(j|i) = \frac{\exp(-\|\mathbf{x}_j - \mathbf{x}_i\|^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-\|\mathbf{x}_k - \mathbf{x}_i\|^2/2\sigma_i^2)}$$

  σ_i is set by user (implicitly)

- Similarity between points in derived feature space $\mathbb{R}^d, d << D$

$$q(i,j) = \frac{g(|\mathbf{y}_i - \mathbf{y}_j|)}{\sum_{k \neq l} g(|\mathbf{y}_i - \mathbf{y}_j|)}$$

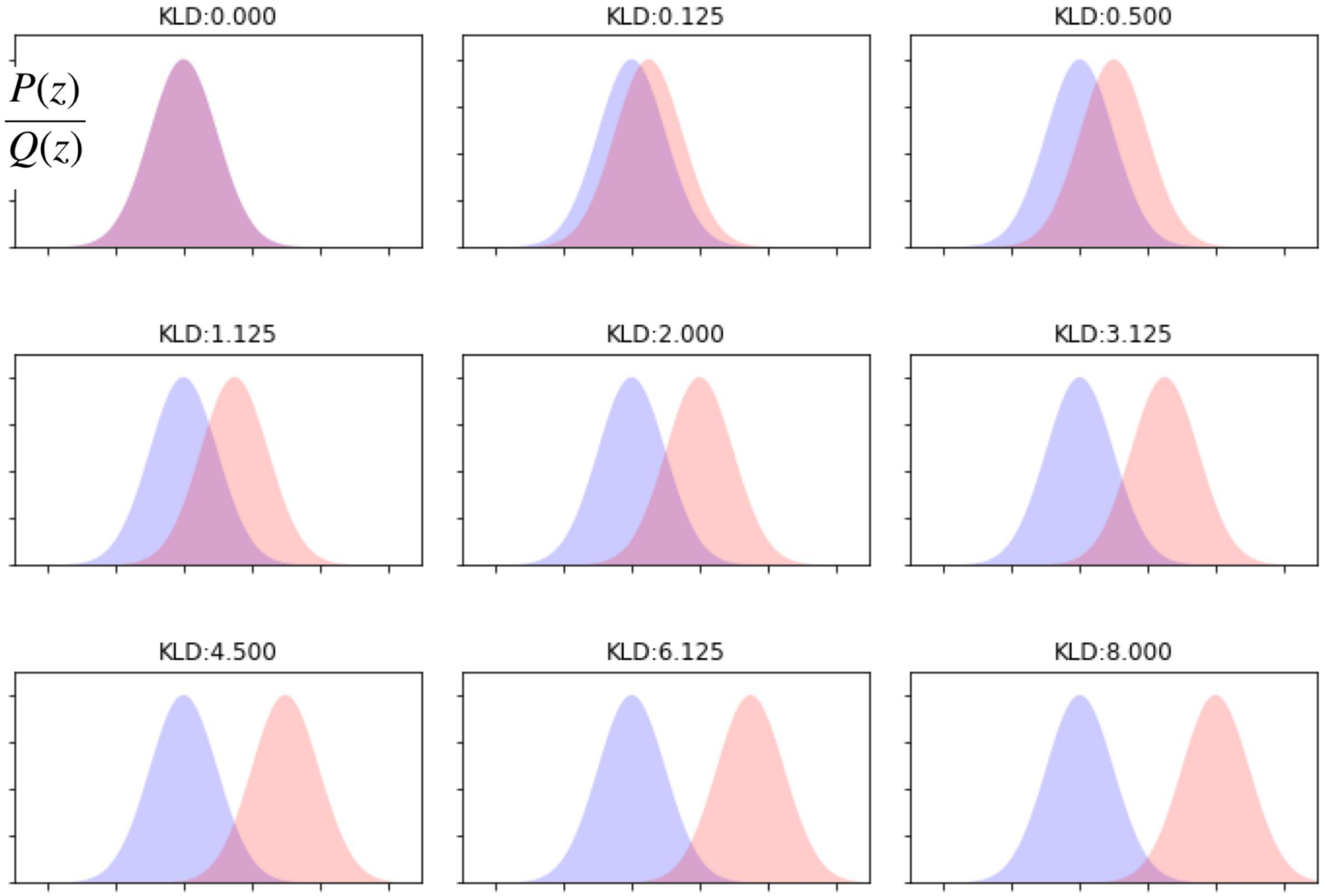
  where $g(z) = \frac{1}{1+z^2}$ - is student t-distribution with dof=1

- Similarity between feature spaces (Kullback–Leibler divergence)

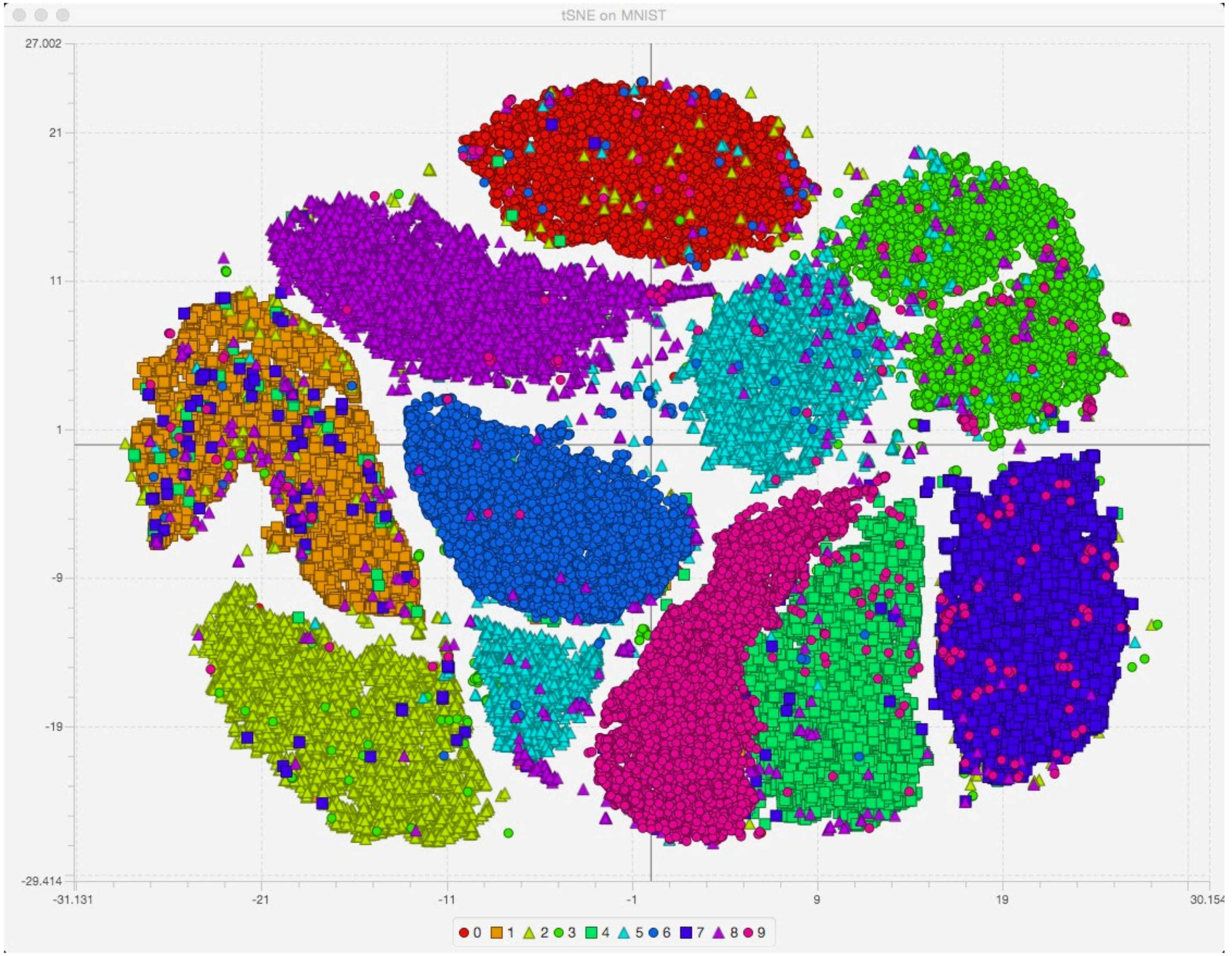
$$J_{t-SNE}(y) = KL(P\|Q) = \sum_i \sum_j p(i,j) \log \frac{p(i,j)}{q(i,j)} \to \min_{\mathbf{y}}$$

KL-Divergence

$$KL(P\|Q) = \sum_z P(z) \log \frac{P(z)}{Q(z)}$$



http://jmlr.csail.mit.edu/papers/volume9/vandermaaten08a/vandermaaten08a.pdf


tSNE on MNIST
27.002
21
11
1
-9
-19
-29.414
-31.131
-21
-11
-1
9
19
30.154
0 1 2 3 4 5 6 7 8 9

T-SNE Demo

http://distill.pub/2016/misread-tsne/

http://lvdmaaten.github.io/tsne/

Semi-conclusion

Linear - Principle Component Analysis (PCA)

- Should be run on scaled data
- Relatively easy to interpret
- Solution is achieved on top k eigenvectors $a_1,...a_k$ of covariance matrix.
- Slow to calculate without SVD hacks (randomized, truncated)

Non-linear – t-SNE

- Can produce nice visualization
- Unstable
- Size of clusters can mean anything
- Distance can mean anything
- Random data can provide structure

Clustering

Goal

Clustering is partitioning of objects into groups so that:

> **inside groups objects are very similar**
> **objects from different groups are dissimilar**

Unsupervised learning

> **No definition of similar**

different algorithms use different formalizations of similarity

Applications

data summarization

- feature vector is replaced by cluster number

feature extraction

- cluster number, cluster average target, distance to native cluster center / other clusters

customer segmentation

- e.g. for recommender service

community detection in networks

- nodes - people, similarity - number of connections

outlier detection

- outliers do not belong any cluster

Comparison

We can compare clustering algorithms in terms of:

computational complexity

do they build at or hierarchical clustering?

can the shape of clustering be arbitrary?

- if not is it symmetrical, can clusters be of different size? can clusters vary in density of contained objects?

robustness to outliers

Nearest neighbors approach (K-means)

The k-means algorithm captures the insight that each point in a cluster should be near to the center of that cluster.

first we choose k, the number of clusters. Then, the centers of those k clusters, called centroids, are initialized in some fashion.

Then proceeds in two alternating parts:

Reassign Points step: we assign every point to the cluster of the nearest centroid

Update Centroids step, we recalculate each centroid's location as the mean (center) of all the points assigned to its cluster.

Repeat until convergence

For those who love coding & math

Initialize μ_j, $j = 1, 2, ...K$.

WHILE not converged:

FOR $i = 1, 2, ...N$:

find cluster number of x_i:

$z_i = \arg\min_{j \in \{1,2,...K\}} ||x_i - \mu_j||_2^2$

FOR $j = 1, 2, ...K$:

$\mu_j = \frac{1}{\sum_{n=1}^{N} \mathbb{I}[z_n = j]} \sum_{n=1}^{N} \mathbb{I}[z_n = j] x_i$

$$\sum_{n=1}^{N} \|x_n - \mu_{z_n}\|_2^2 \to \min_{z_1, ...z_N, \mu_1, ...\mu_K} \quad (1)$$

K-means properties

K-means Demo

Initialization:

- typically centers are initialized to randomly chosen training objects
  "farthest" heuristic
  use result of another clustering method as starting point
  k-means++, http://en.wikipedia.org/wiki/K-means%2B%2B

Convergence conditions:

- maximum number of iterations reached
  cluster assignments stop to change (exact)
  centers stop changing significantly (approximate)

K-means optimality

Optimality:

> criteria is non-convex
> solution depends on starting conditions
> may restart several times from different initializations and select solution giving minimal value of (1).

Complexity: O(NDKI)

> K is the number of clusters
> I is the number of iterations.

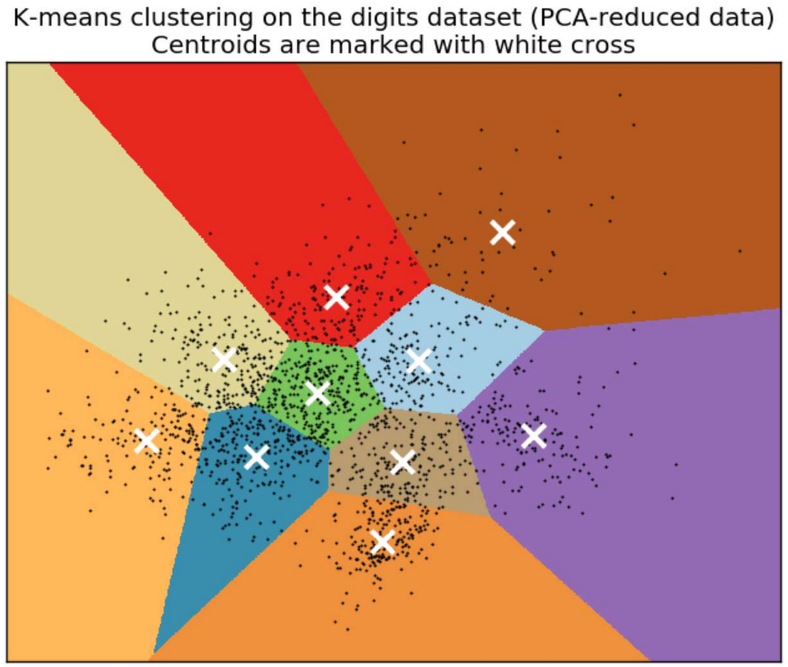
› usually few iterations are enough for convergence.

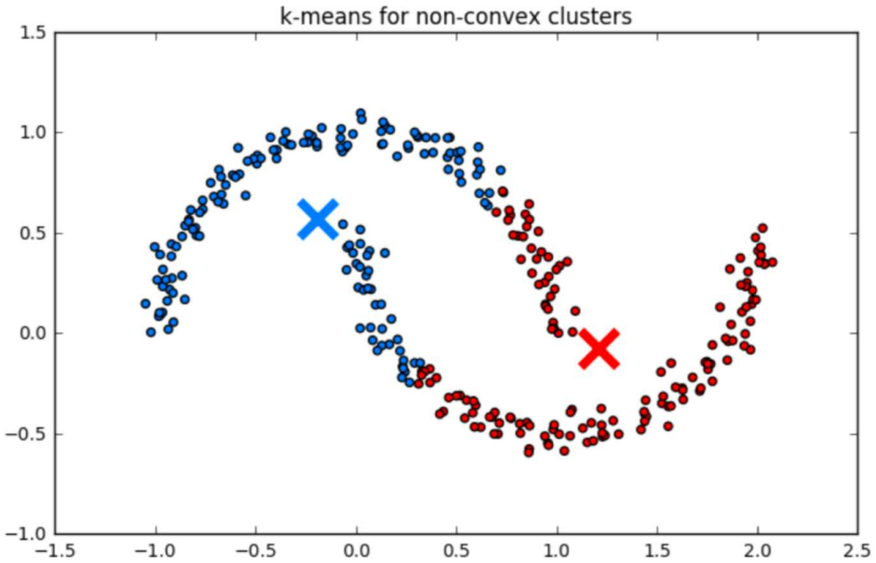
K-means gotchas

K-means assumes that clusters are convex

It always finds clusters even if none actually exist

- need to control cluster quality metrics

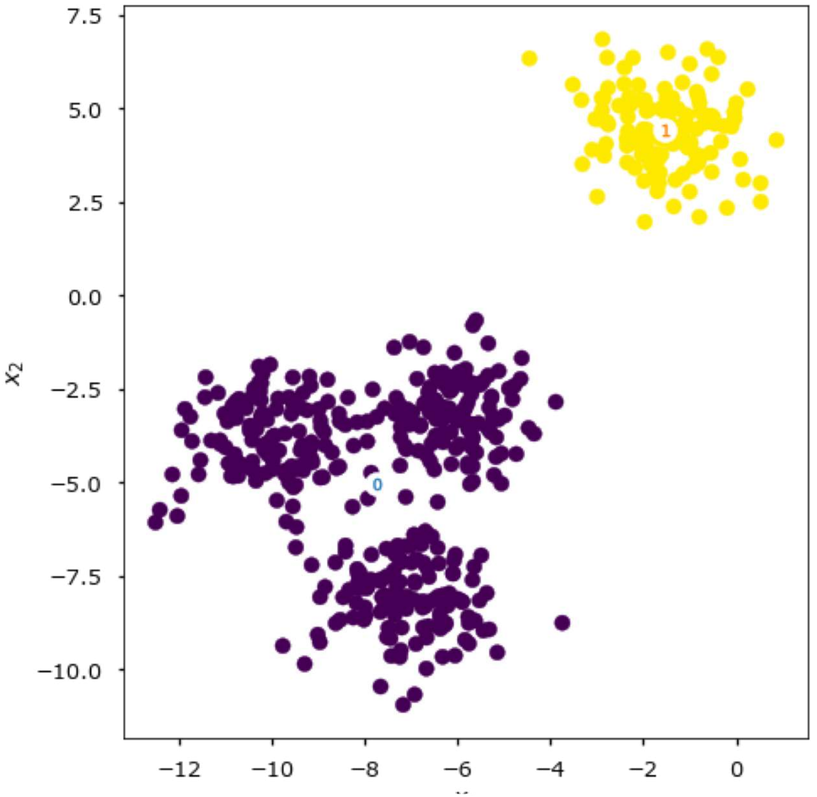


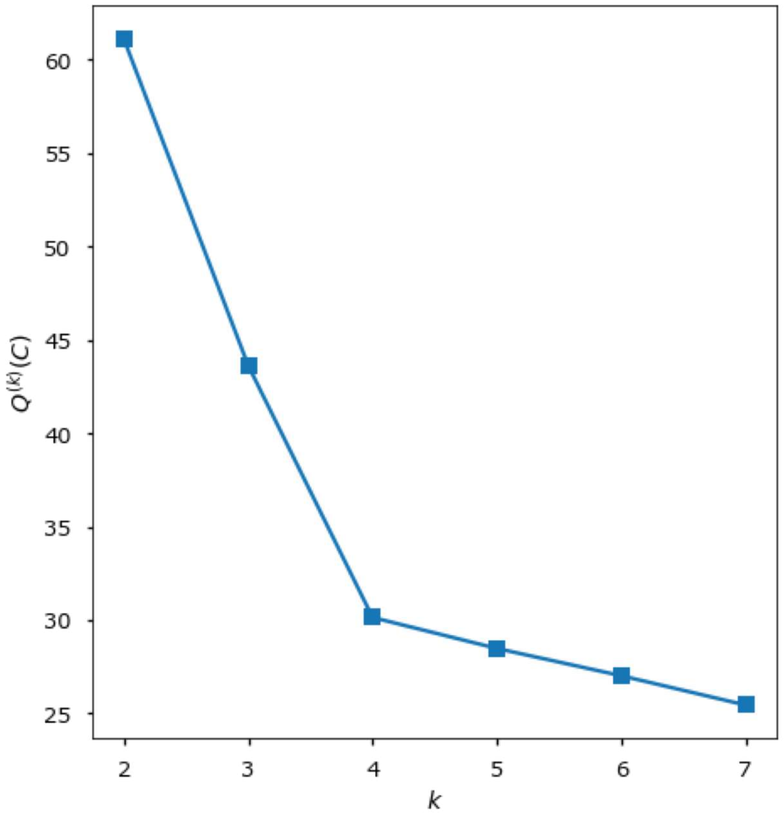


Elbow method

Elbow method gives us an idea on what a good *k* number of clusters would be based on the sum of squared distance (SSE) between data points and their assigned clusters' centroids.

$$\sum_{n=1}^{N} \|x_n - \mu_{z_n}\|_2^2 \to \min_{z_1,\dots z_N,\mu_1,\dots\mu_K} \quad (1)$$



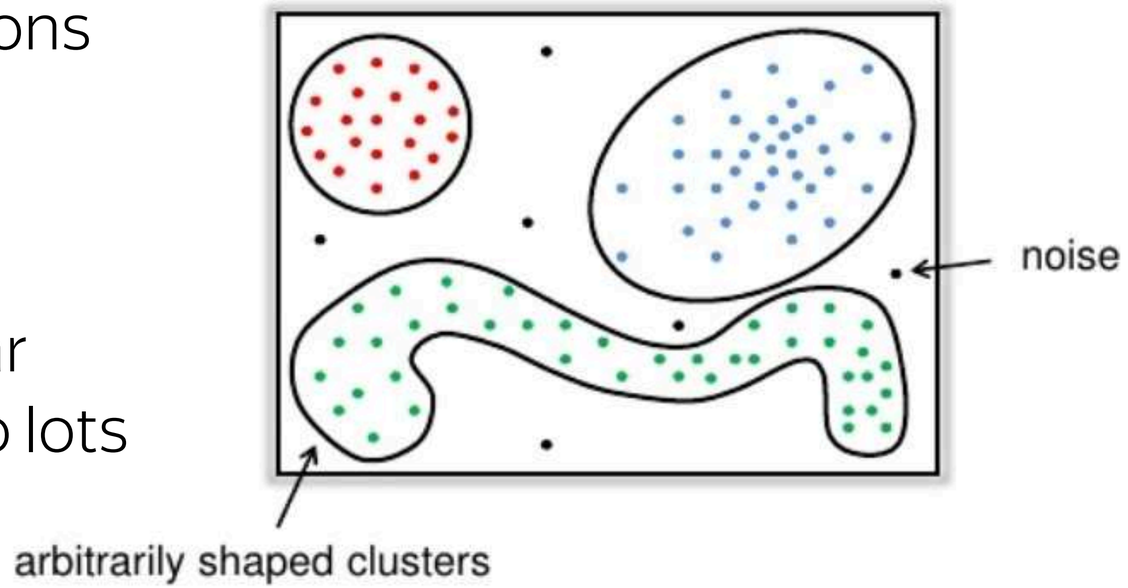


$$D(k) = \frac{|Q^{(k)}(C) - Q^{(k+1)}(C)|}{|Q^{(k-1)}(C) - Q^{(k)}(C)|}$$

DBSCAN

Density-Based Spatial Clustering of Applications with Noise

captures the insight that clusters are dense groups of points. The idea is that if a particular point belongs to a cluster, it should be near to lots of other points in that cluster.



Idea

Identify 2 parameters

- ϵ - radius of neighborhood of each point. $\mathrm{N}_{\epsilon}(\mathrm{p})$ - neighborhood of point p
- min_pts - minimal number of points inside neighborhood

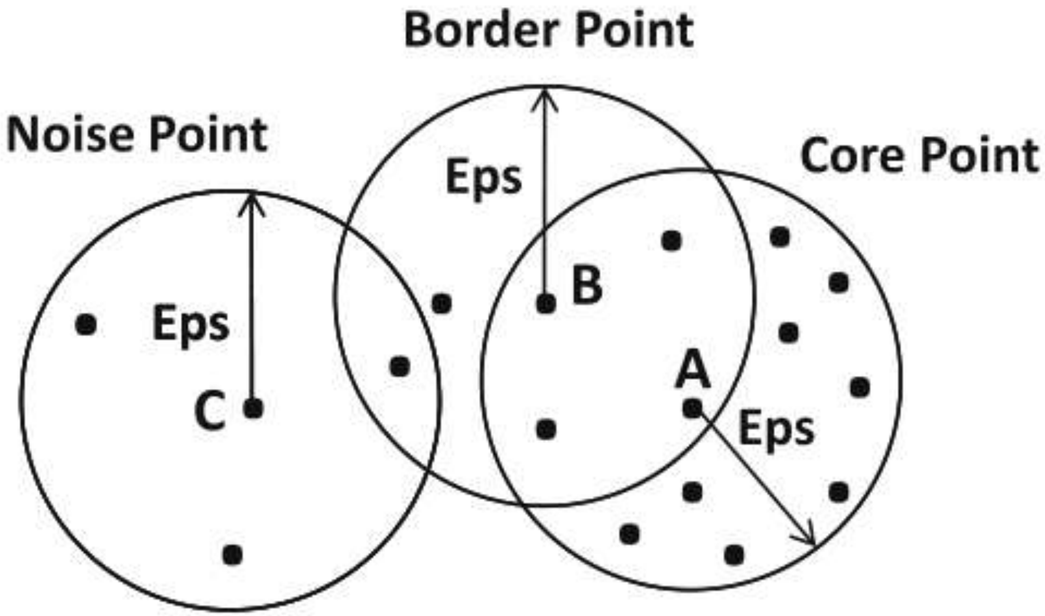
To start a cluster from point there should be at least min_pts points in $\mathrm{N}_{\epsilon}(\mathrm{p})$
If this conditions is satisfied - spread cluster and check all points neighbors by the same criterion

Types of points

Core point: point having $\geq$`min_pts` points in its ε-neighbourhood

Border point: not core point, having at least 1 core point in its ε-neighbourhood

Noise point: neither a core point nor a border point



Demo

Semi-conclusion

K-means

- Works best in datasets that have with clusters that are roughly equally-sized and shaped roughly regularly
- Can only be applied when the data points lie in a Euclidean space, failing for more complex types of data
- Requires choice of k and initial centroids
- Works well, relatively fast, well-interpretable

DBSCAN

- Do not indicate number of clusters
- Arbitrary shapes of clusters
- Identifies outliers
- Problems with varying density

Cluster validity and Quality Measures

Evaluate using "ground-truth" clustering (Quality Measure)

- **invariant to cluster naming**

Unsupervised criterion (Cluster Validity)

- **objects from same cluster should be similar**
- **objects from different clusters should be different**

Based on ground-truth

Rand index $$\mathrm{Rand}(\hat{\pi}, \pi^*) = \frac{a+d}{a+b+c+d},$$

a - number of pairs that are grouped both in $\hat{\pi}$ and $\pi*$
d - number of pairs that are separated both in $\hat{\pi}$ and $\pi*$,
b (c) - number of pairs that are separated both in $\hat{\pi}$ ($\pi*$), but grouped in $\pi*$ ($\hat{\pi}$)

Precision / recall

$$\text{Rand}(\hat{\pi}, \pi^*) = \frac{tp + tn}{tp + fp + fn + tn},$$

- $\text{Precision}(\hat{\pi}, \pi^*) = \frac{tp}{tp+fn}$
- $\text{Recall}(\hat{\pi}, \pi^*) = \frac{tp}{tp+fp}$
- $\text{F-measure}(\hat{\pi}, \pi^*) = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$

Cluster validity

Intuition

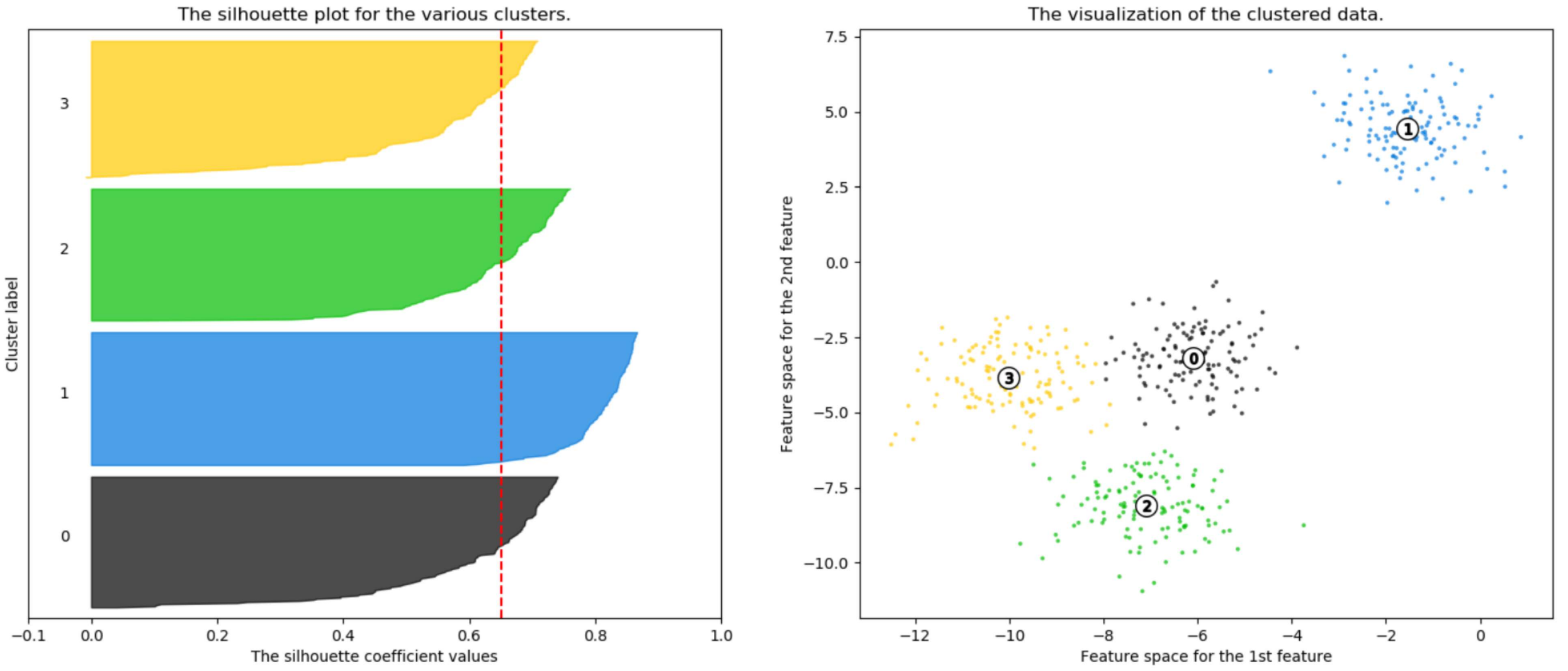
> objects from same cluster should be similar
> objects from different clusters should be different

For each object x_i define:

> s_i - mean distance to objects in the same cluster
> d_i - mean distance to objects in the nearest cluster

$$Silhouette_i = \frac{d_i - s_i}{\max\{d_i, s_i\}}$$

$$Silhouette = \frac{1}{N} \sum_{i=1}^{N} \frac{d_i - s_i}{\max\{d_i, s_i\}}$$



http://bit.ly/2LyyV6o

Discussion

Advantages

- The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering.
- Scores around zero indicate overlapping clusters.
- The score is higher when clusters are dense and well separated.

Disadvantages

- complexity $O(N^2D)$
- favors convex clusters

Conclusion

Unsupervised learning – methods for exploration of the geometrical properties of the dataset

Novelty/anomaly detection
Representation/Compression
Similarity between instances

References

PCA

http://sebastianraschka.com/Articles/2015_pca_in_3_steps.html
http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html
https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python/
http://www.math.union.edu/~jaureguj/PCA.pdf

T-SNE

https://oreillymedia.github.io/thebe/examples/t-sne-build.html
https://distill.pub/2016/misread-tsne/
https://www.analyticsvidhya.com/blog/2017/01/t-sne-implementation-r-python/
http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html