DATA MINING 2
Dimensionality Reduction

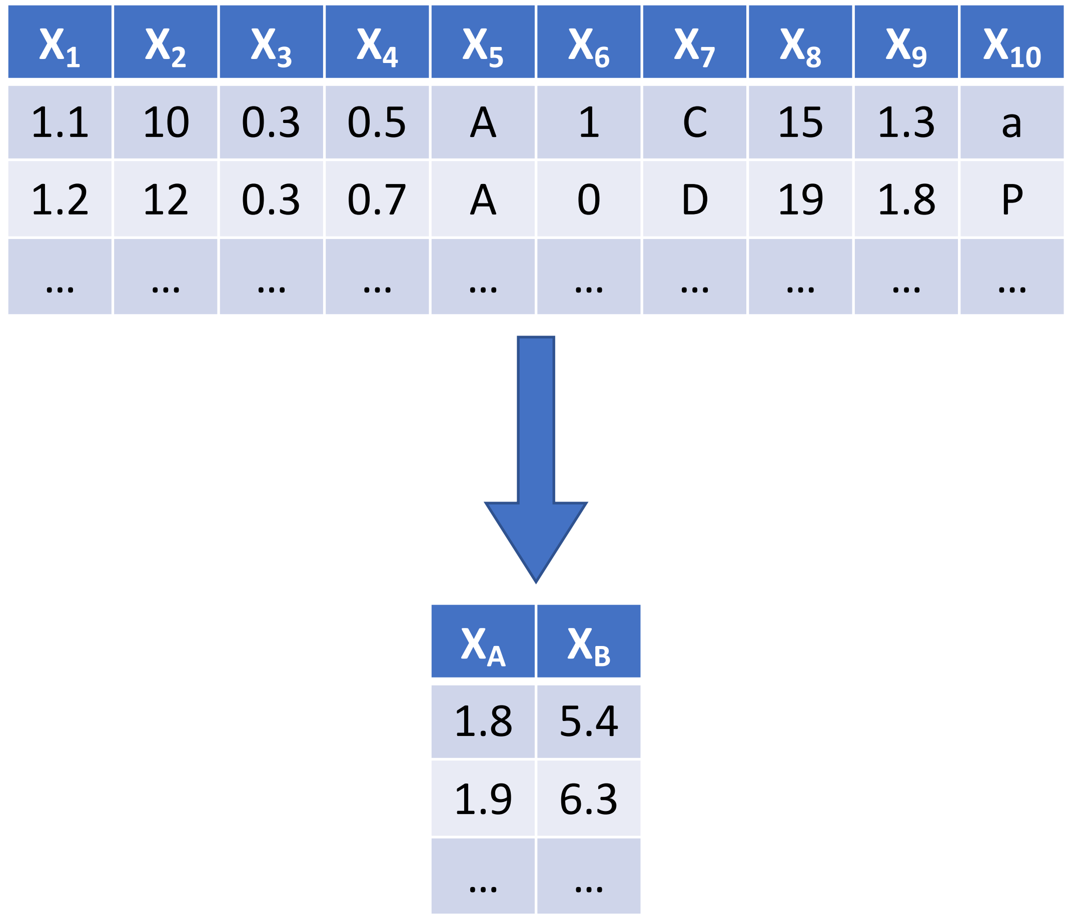
Riccardo Guidotti

a.a. 2021/2022

UNIVERSITÀ DI PISA

Dimensionality Reduction

- Dimensionality reduction is the process of reducing the number of random variables under consideration by obtaining a set of principal variables.
- Approaches can be divided into **feature selection** and **feature projection**.



X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
1.1	10	0.3	0.5	A	1	C	15	1.3	a
1.2	12	0.3	0.7	A	0	D	19	1.8	P
…	…	…	…	…	…	…	…	…	…

X_A	X_B
1.8	5.4
1.9	6.3
…	…

Feature Selection

- Select a subset of the features according to different strategies:
  - the **filter** strategy (e.g. information gain),
  - the **wrapper** strategy (e.g. search guided by accuracy),
  - the **embedded** strategy (selected features add or are removed while building the model based on prediction errors).

- Classification and/or regression or can be done in the reduced space more accurately than in the original space.

Feature Selection

- **Variance Threshold.** It removes all features whose variance does not meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.
- **Univariate Feature Selection.** It selects the best features based on univariate statistical tests. For instance, it removes all but the *k* highest scoring features. An example of statistical test is the ANOVA F-value between label/feature.
- F-value = $\sum_{i=1}^{K} n_i(\bar{Y}_{i\cdot} - \bar{Y})^2/(K-1) / \sum_{i=1}^{K}\sum_{j=1}^{n_i}\left(Y_{ij} - \bar{Y}_{i\cdot}\right)^2/(N-K),$
- where $\bar{Y}_{i\cdot}$ denotes the sample mean in the ith group, n_i is the number of observations in the ith group, $\bar{Y}$ denotes the overall mean of the data, Y_{ij} is the jth observation in the ith out of *K* groups, *K* denotes the number of groups, *N* the overall sample size.
- F-value is large if the numerator is large, which is unlikely to happen if the population means of the groups all have the same value.

Recursive Feature Elimination (RFE)

- Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model, or feature importance of decision tree), RFE selects features by recursively considering smaller and smaller sets of features.
- First, the estimator is trained on the initial set of features and the importance of each feature is obtained.
- Then, the least important features are pruned from current set of features.
- That procedure is recursively repeated on the pruned set until the desired number of features to select is eventually reached.

Feature Projection (a.k.a Feature Extraction)

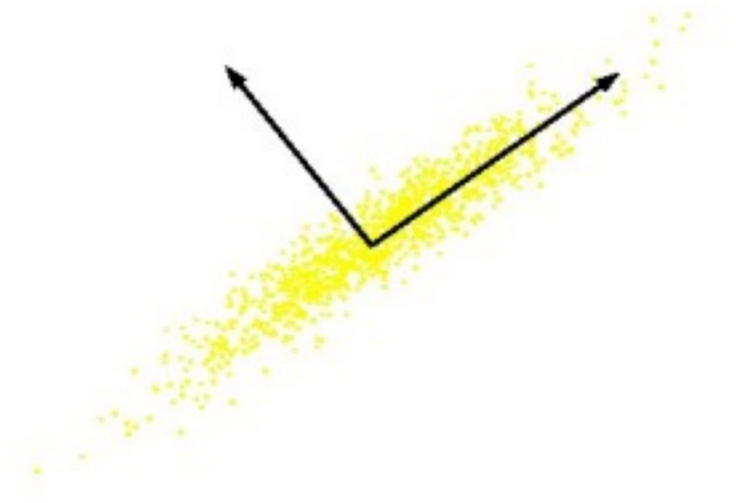
- It transforms the data in the high-dimensional space to a space of fewer dimensions.
- The data transformation may be linear, or nonlinear.
- Approaches:
  - Principal Component Analysis (PCA)
  - Non-negative matrix factorization (NMF)
  - Linear Discriminant Analysis (LDA)
  - Multidimensional Scaling
    - Sammon
    - IsoMap
    - t-SNE
  - Autoencoder

Feature Projection (a.k.a Feature Extraction)

- It transforms the data in the high-dimensional space to a space of fewer dimensions.
- The data transformation may be linear, or nonlinear.
- Approaches:
  - Principal Component Analysis (PCA)
  - Non-negative matrix factorization (NMF)
  - Linear Discriminant Analysis (LDA)
  - Multidimensional Scaling
    - Sammon
    - IsoMap
    - t-SNE
  - Autoencoder

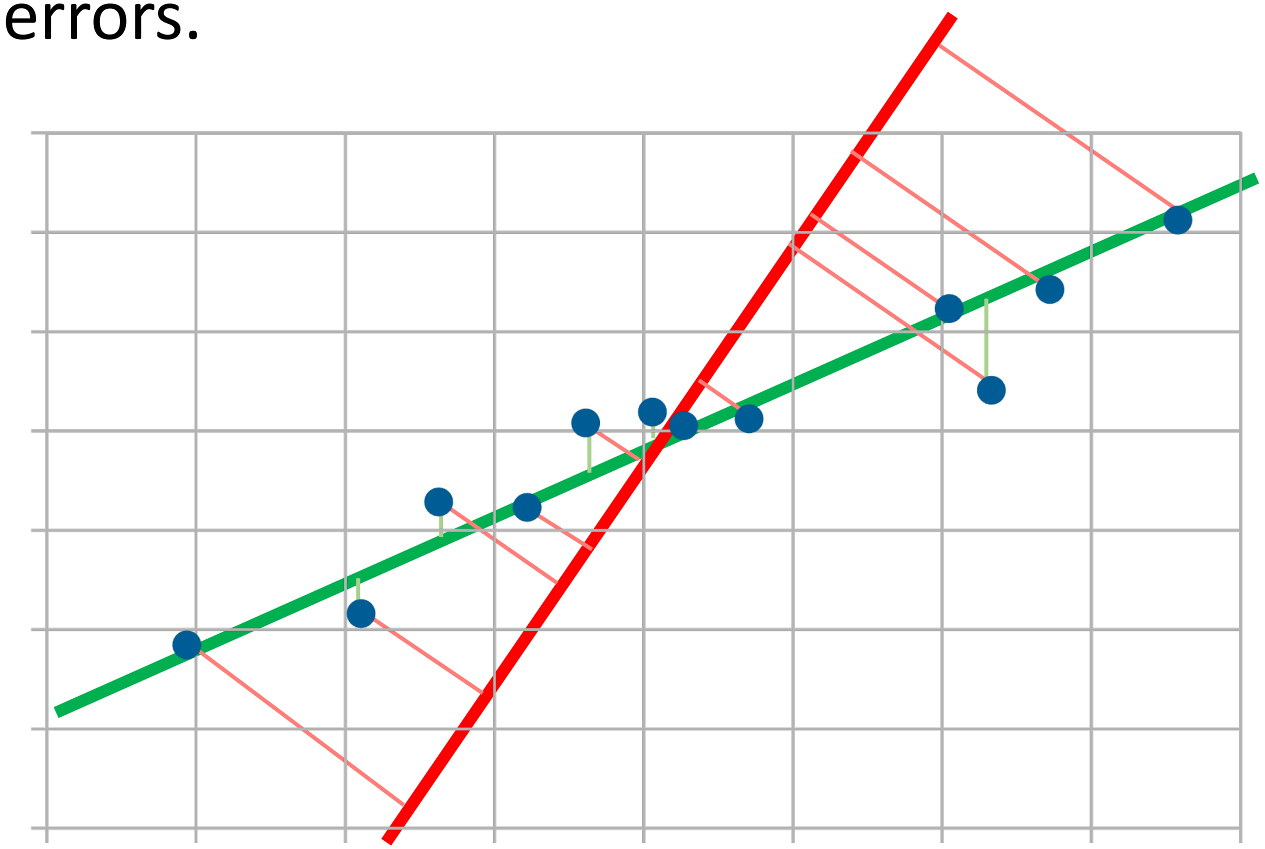
Principal Component Analysis (PCA)

- The goal of PCA is to find a new set of dimensions (attributes or features) that better captures the variability of the data.
- The first dimension is chosen to capture as much of the variability as possible.
- The second dimension is orthogonal to the first and, subject to that constraint, captures as much of the remaining variability as possible, and so on.



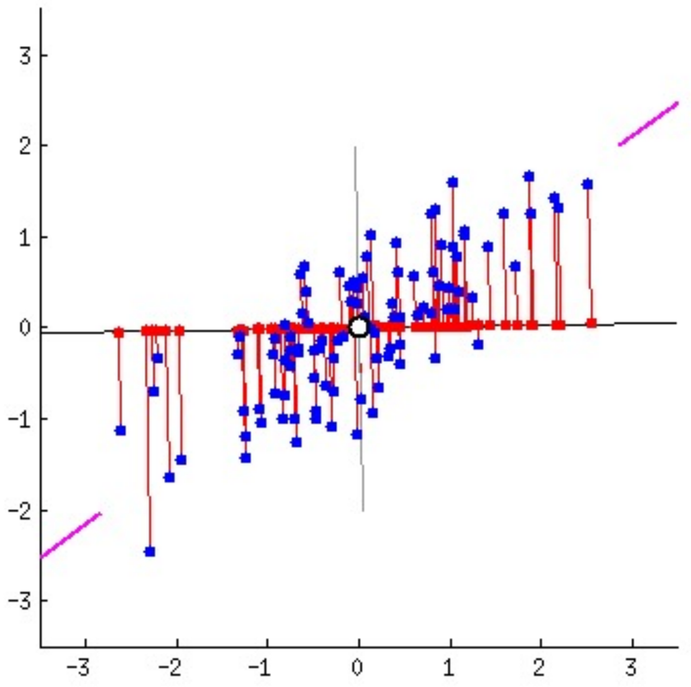
PCA – Conceptual Algorithm

- Find a line such that, when the data is projected onto that line, it has the maximum variance; minimize the sum-of-squares of the projection errors.



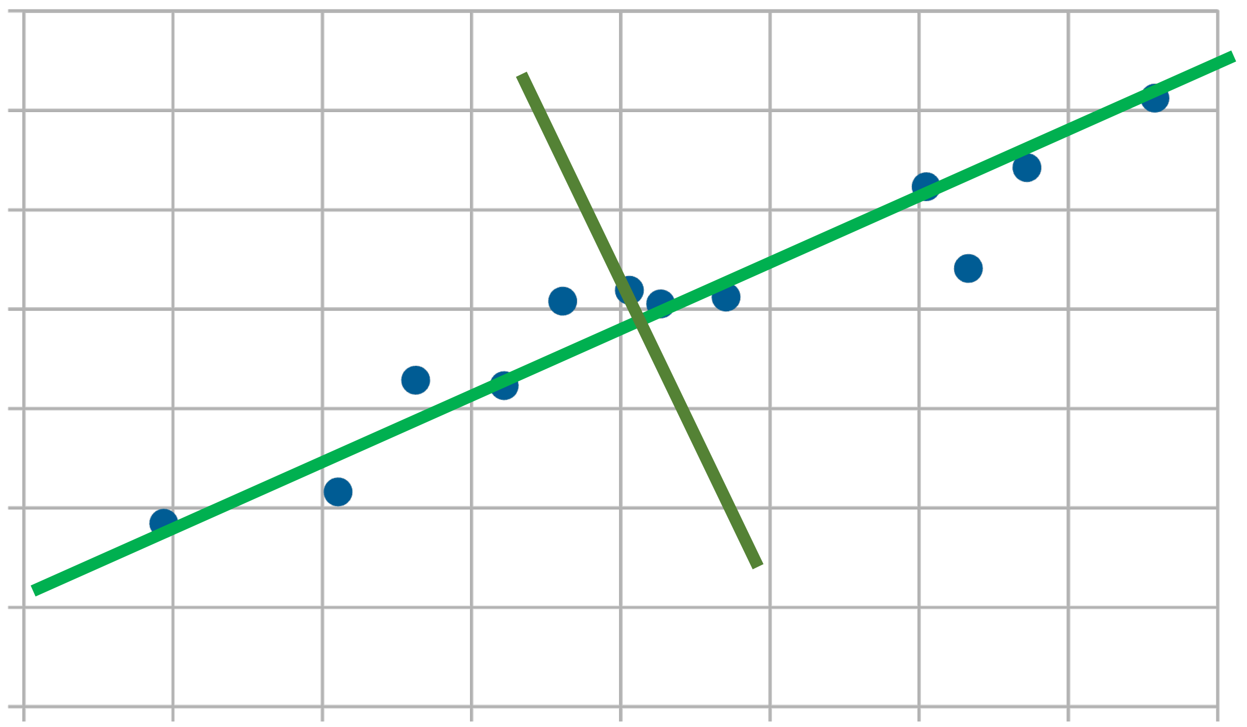
PCA – Conceptual Algorithm

- Find a line such that, when the data is projected onto that line, it has the maximum variance; minimize the sum-of-squares of the projection errors.



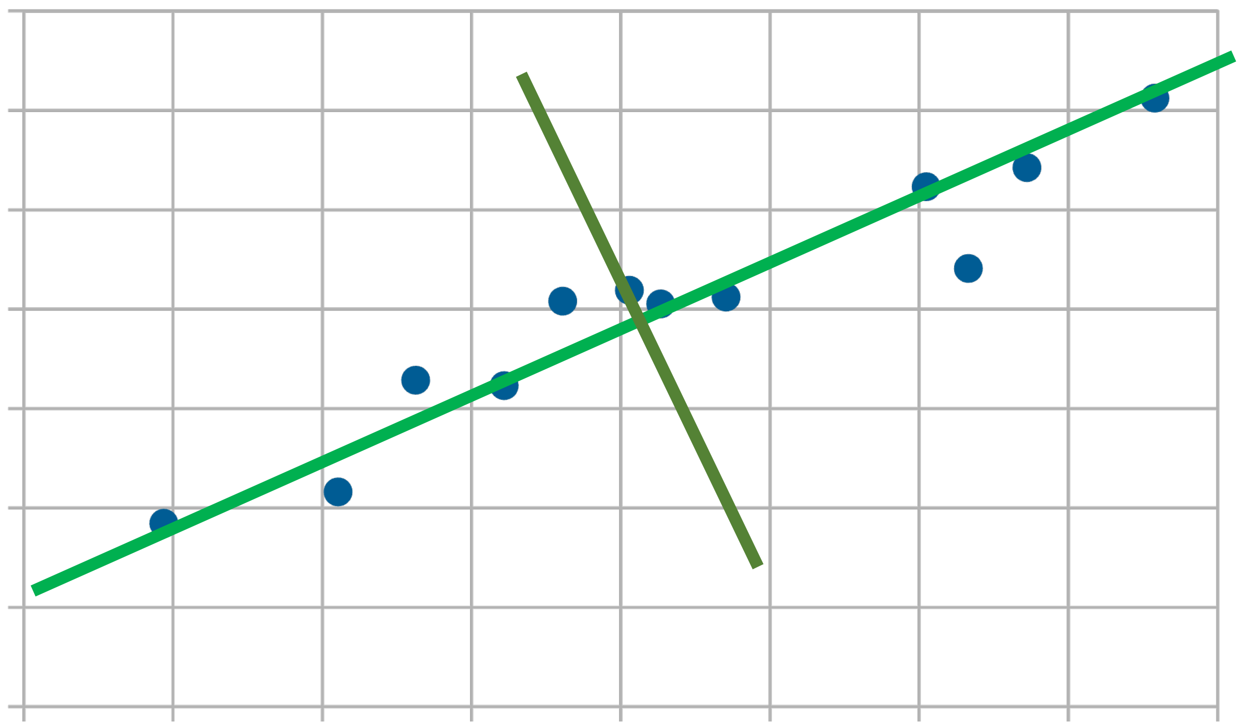
PCA – Conceptual Algorithm

- Find a second line, orthogonal to the first, that has maximum projected variance.



PCA – Conceptual Algorithm

- Repeat until have *k* orthogonal lines.
- The projected position of a point on these lines gives the coordinates in the k-dimensional reduced space.



Background: Covariance, Eigenvalue and Eigenvectors

- The covariance of two attributes is a measure of how strongly the attributes vary together.

$$\text{covariance}(\mathbf{x}, \mathbf{y}) = s_{xy} = \frac{1}{n-1}\sum_{k=1}^{n}(x_k - \overline{x})(y_k - \overline{y})$$

- Eigenvector of matrix *X*: a vector *v* such that $Xv=\lambda v$
- λ: eigenvalue of eigenvector v
- A square matrix *X* of rank *r*, has *r* orthonormal eigenvectors $v_1, v_2, \ldots, v_r$ with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_r$.
- Eigenvectors define an orthonormal basis for the column space of *X*

Steps in PCA

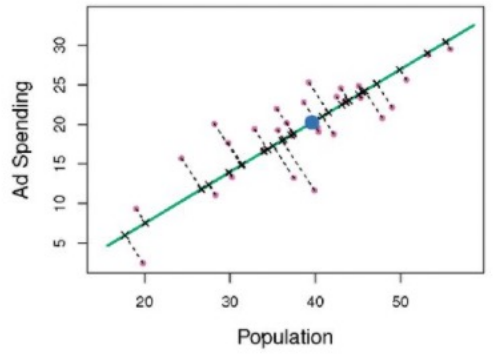
- Calculate the mean value of the data of every dimension

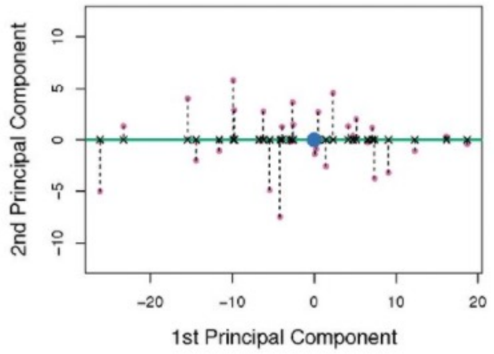
- Calculate the covariance matrix of all pairs of attributes
  - Given matrix of data X, remove the mean of each column from the column vectors to get the centered matrix C
  - The matrix Σ = $C^T C$ is the covariance matrix of the row vectors of X.

- Calculate eigenvalues and eigenvectors of Σ
  - Methods: power iteration method, Singular Value Decomposition
  - Eigenvector with largest eigenvalue λ_1 is the 1st PC
  - Eigenvector with kth largest eigenvalue λ_k is the kth PC
  - $\lambda_k / \Sigma_i \lambda_i$ is the proportion of variance captured by the kth PC

Applying the PCA

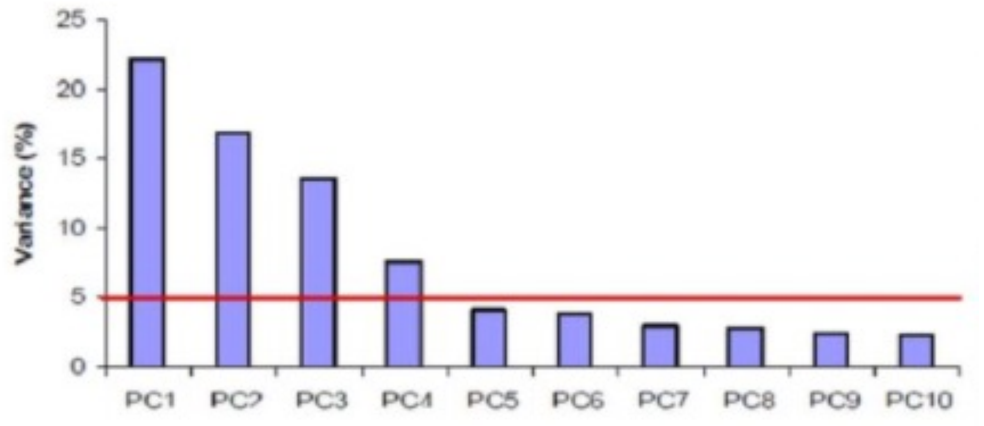
- The full set of PCs comprise a new orthogonal basis for feature space, whose axes are aligned with the maximum variances of original data.
- Projection of original data into first *k* PCs gives a reduced dimensionality representation of the data.
- Transforming reduced dimensionality projection back into original space gives a reduced dimensionality reconstruction of the data.
- Reconstruction will have some error.





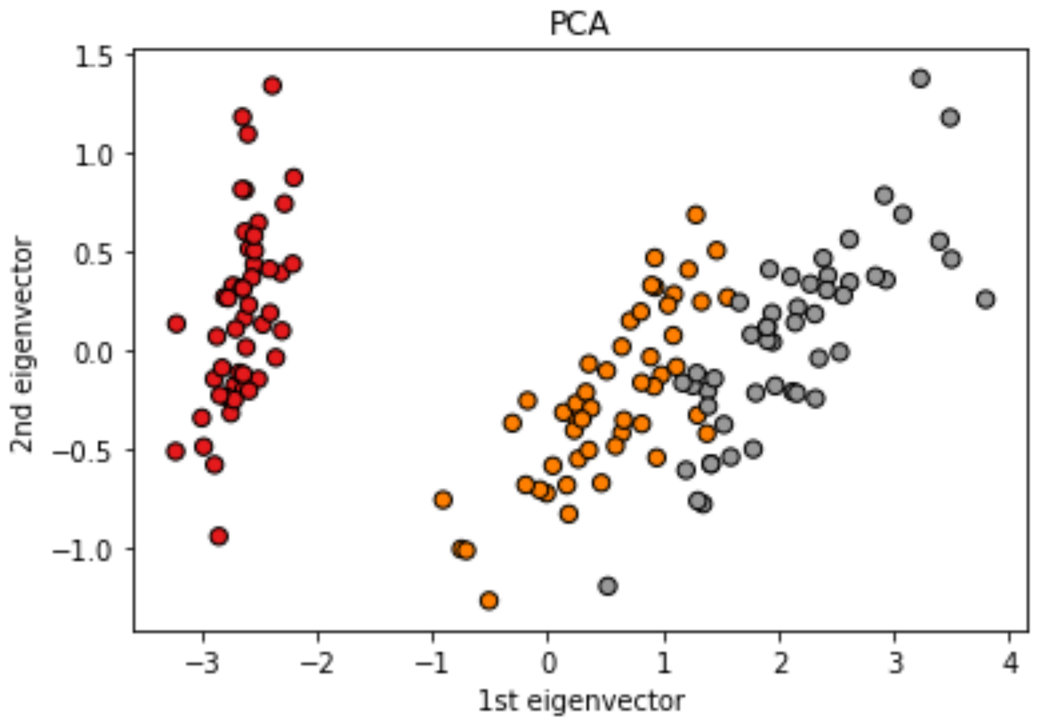
Select the dimension *k*

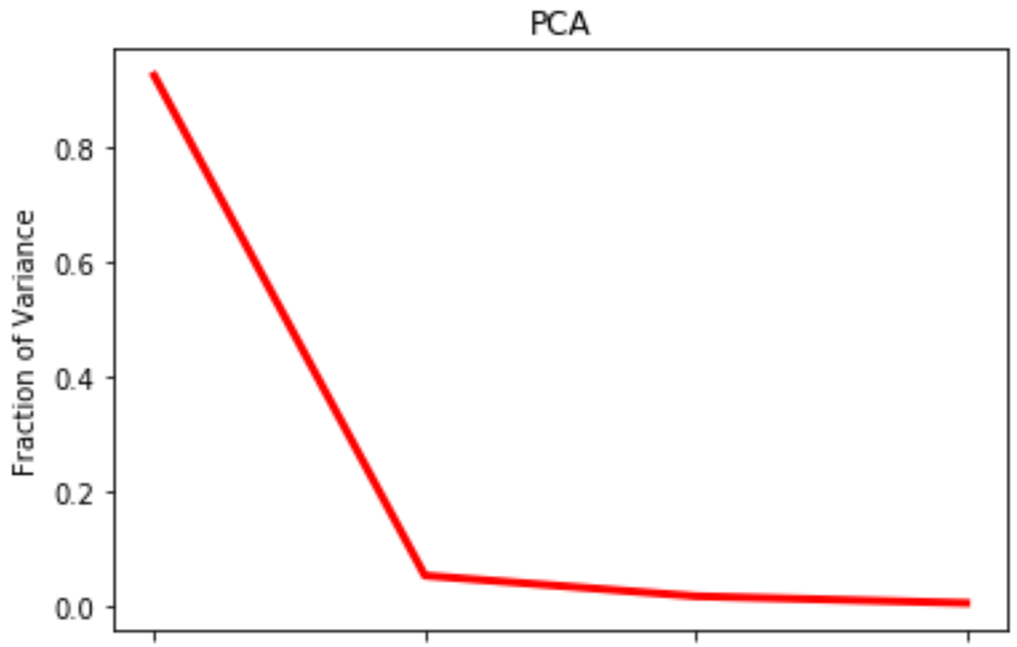
- Rank eigenvalues in decreasing order.
- Select eigenvectors that retain a fixed percentage of variance (e.g., at least a minimum threshold.



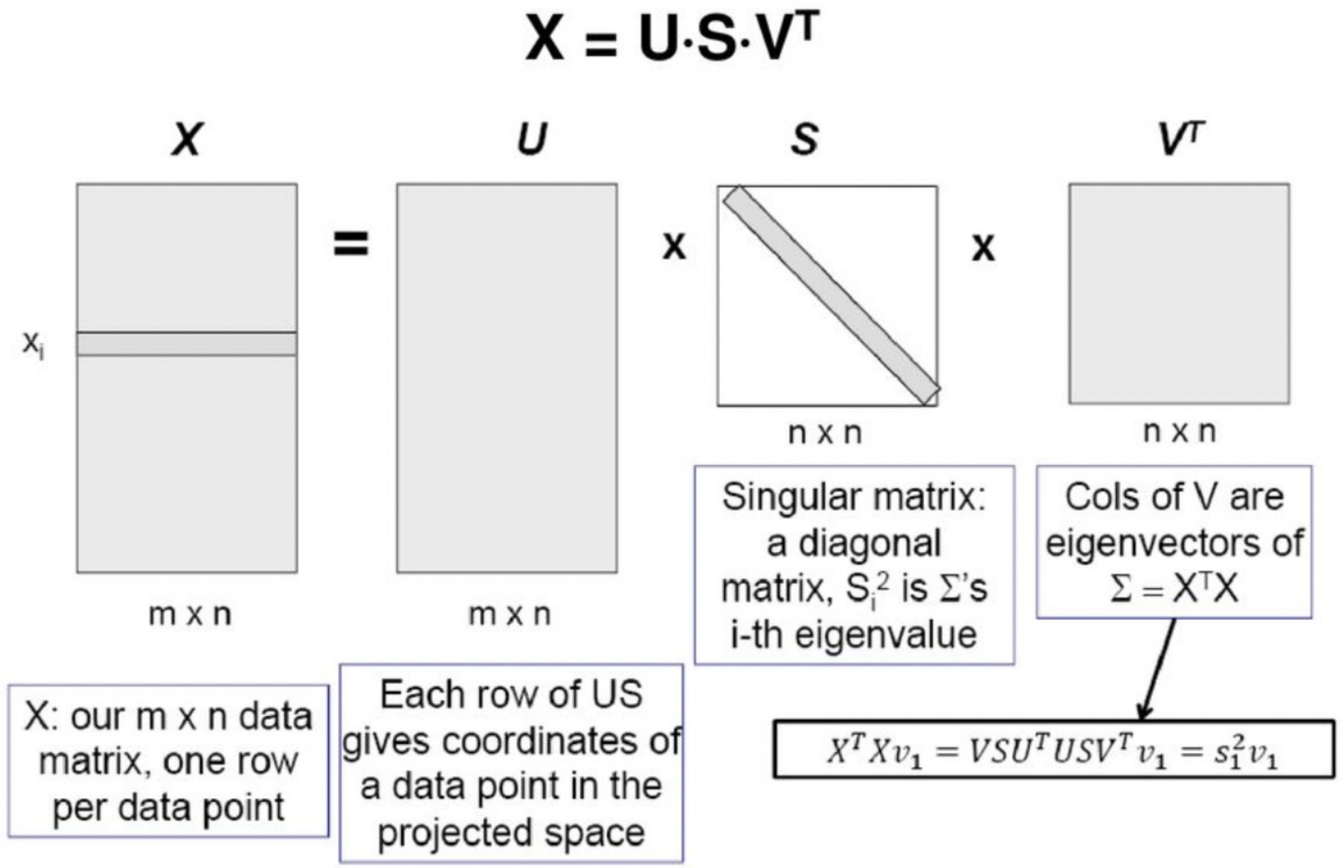
Example

- Iris Dataset

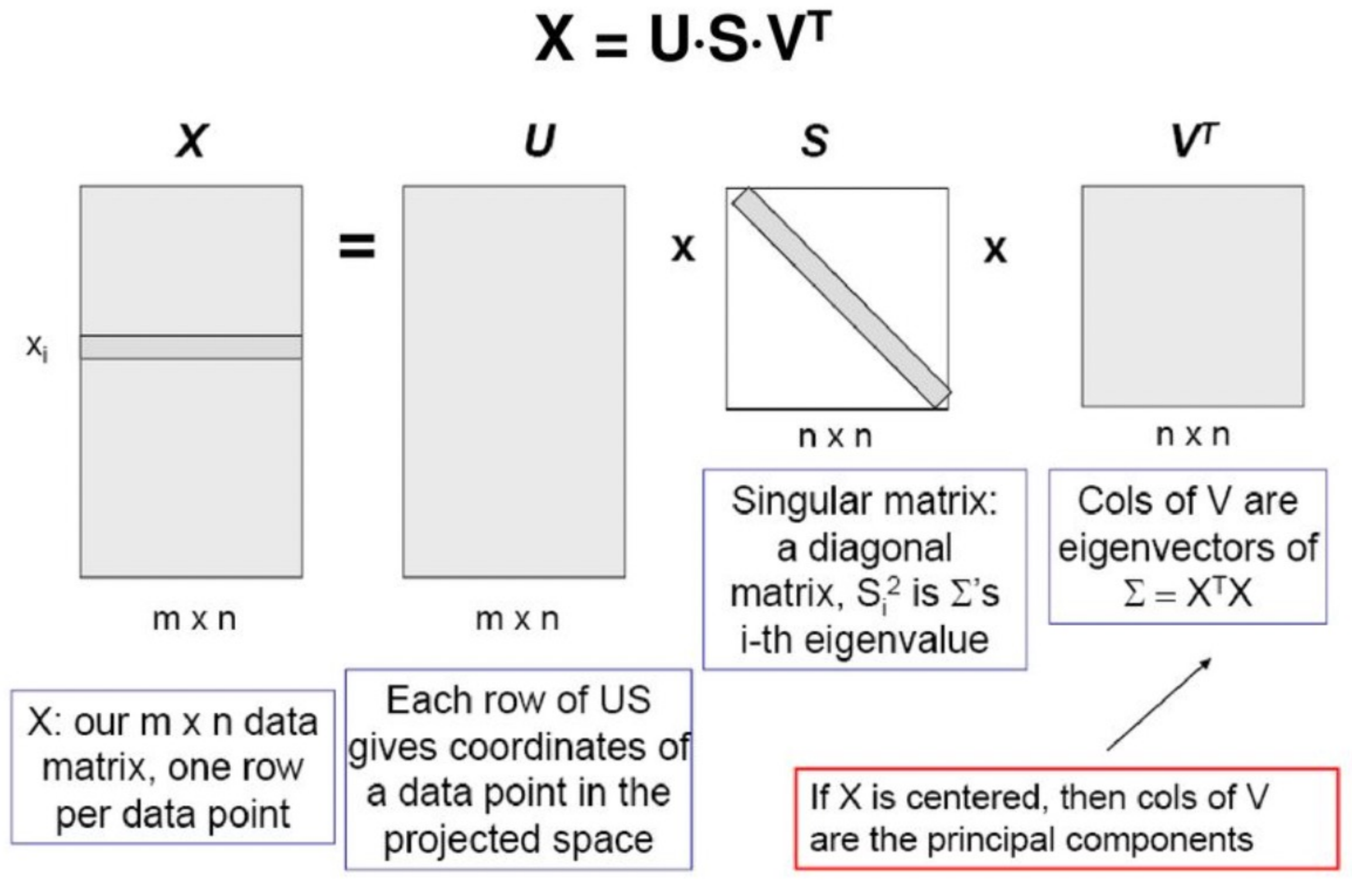




Singular Value Decomposition - SVD


$X = U \cdot S \cdot V^T$
X
U
S
V^T
=
x
x
x_i
m x n
m x n
n x n
n x n
Singular matrix: a diagonal matrix, S_i^2 is Σ's i-th eigenvalue
Cols of V are eigenvectors of $\Sigma = X^T X$
X: our m x n data matrix, one row per data point
Each row of US gives coordinates of a data point in the projected space
$X^T X v_1 = VSU^T USV^T v_1 = s_1^2 v_1$

Singular Value Decomposition - SVD



$X = U \cdot S \cdot V^T$
X
U
S
V^T
=
x
x
x_i
m x n
m x n
n x n
n x n
X: our m x n data matrix, one row per data point
Each row of US gives coordinates of a data point in the projected space
Singular matrix: a diagonal matrix, S_i^2 is Σ's i-th eigenvalue
Cols of V are eigenvectors of $\Sigma = X^T X$
If X is centered, then cols of V are the principal components

PCA via SVD

- Create mean-centered data matrix X
- Solve SVD: $X = USV^T$
- Columns of V are the eigenvectors of Σ sorted from largest to smallest eigenvalues.

- Limits of PCA:
- Limited to linear projections

Partial Least Squares (PLS)

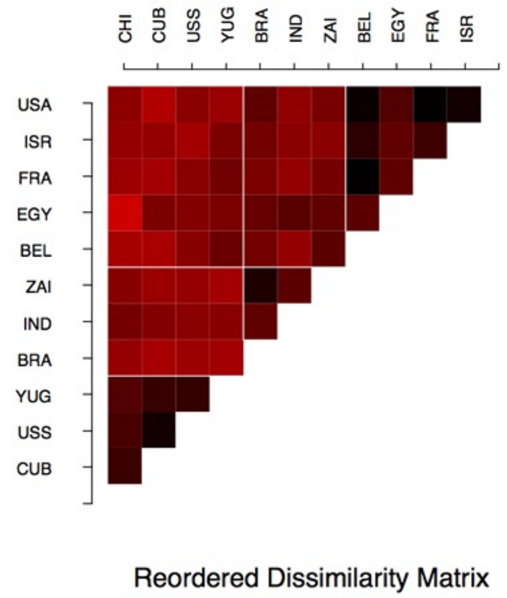
- Supervised alternative to PCA
- Attempts to find the set of orthogonal directions that explain both outcome and features.
- First direction:
  - Calculate simple linear regression (see next lectures) between each feature and outcome
  - Use coefficients to define first direction giving greatest weight to predictors which are highly correlated with outcome (large coefficients)
- Repeat procedure on residuals of predictors

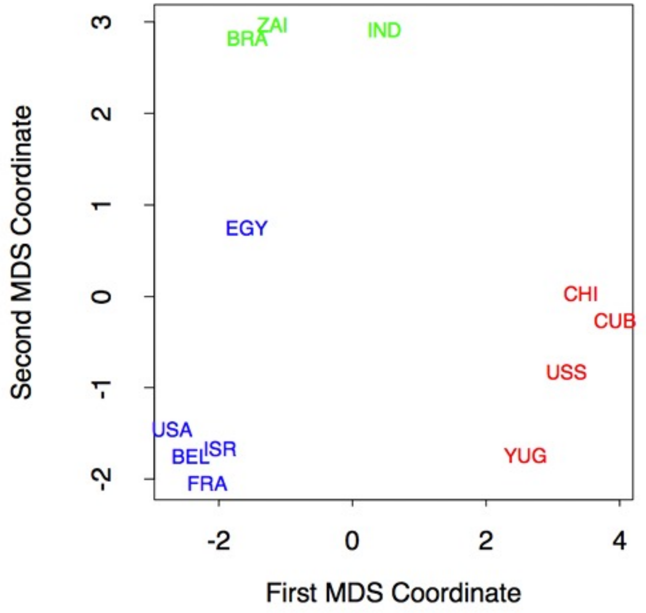
Random Subspace Projection

- High-dimensional data is projected into low-dimensional space using a random matrix whose columns have unit length.
- No attempt to optimize criterion.
- Preserve structure of data (e.g. distances)
- Computationally cheap.

Multi-Dimensional Scaling (MDS)

- Given a pairwise dissimilarity matrix (no need to be a metric), the goal of MDS is to learn a mapping of data into a lower dimensionality such that the relative distances are preserved.
- If two points are close in the feature space, it should be close in the latent factor space.





MDS methods

- MDS is a family of different algorithms designed to map data into a very low configuration, e.g., k=2 or k=3.
- MDS methods include
  - Classical MDS
  - Metric MDS
  - Non-metric MDS

- MDS cannot be inverted

Distance, dissimilarity and similarity

- Distance, dissimilarity and similarity (or proximity) are defined for any pair of objects in any space. In mathematics, a distance function (that gives a distance between two objects) is also called *metric*, satisfying:
  - $d(x, y) \geq 0,$
  - $d(x, y) =$ *iff* $x = y,$
  - $d(x, y) = d(y, x)$
  - $d(x, z) \leq d(x, y) + d(y, z)$
- If the last condition does not hold, than *d* is a distance function but it is not a metric.

MDS – Conceptual Algorithm

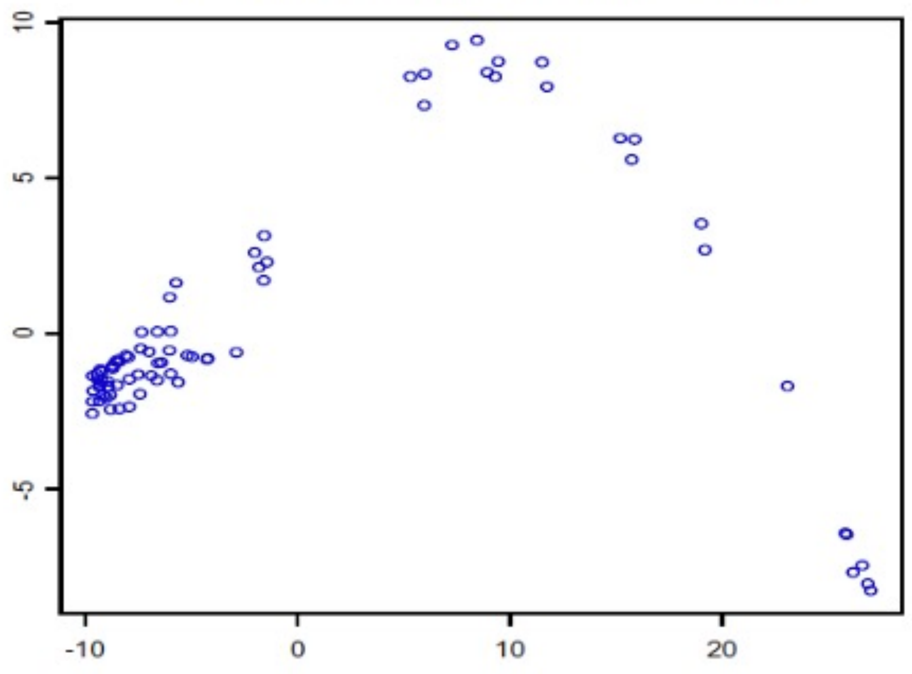
- Given a pairwise dissimilarity matrix *D* and the dimensionality *k,* find a mapping such that $d_{ij} = ||x_i - x_j||$ for all points in *D*.
- Usually, a *gradient descent* approach is adopted to solve an optimization problem that aims at minimizing the function
- $J(x) = \sum_i^n \sum_j^n d_1(d_{ij}, d_2(x_i, x_j))$
- Depending on the distances adopted to calculate *D* and the distance function used for d_1 and d_2 the approach returns a different result.
- The Classic MDS adopts the Euclidean distance for every calculus.
- Metric-MDM adopts metrics as distances
- Non metric-MDM deals with ranks of distances instead of their values

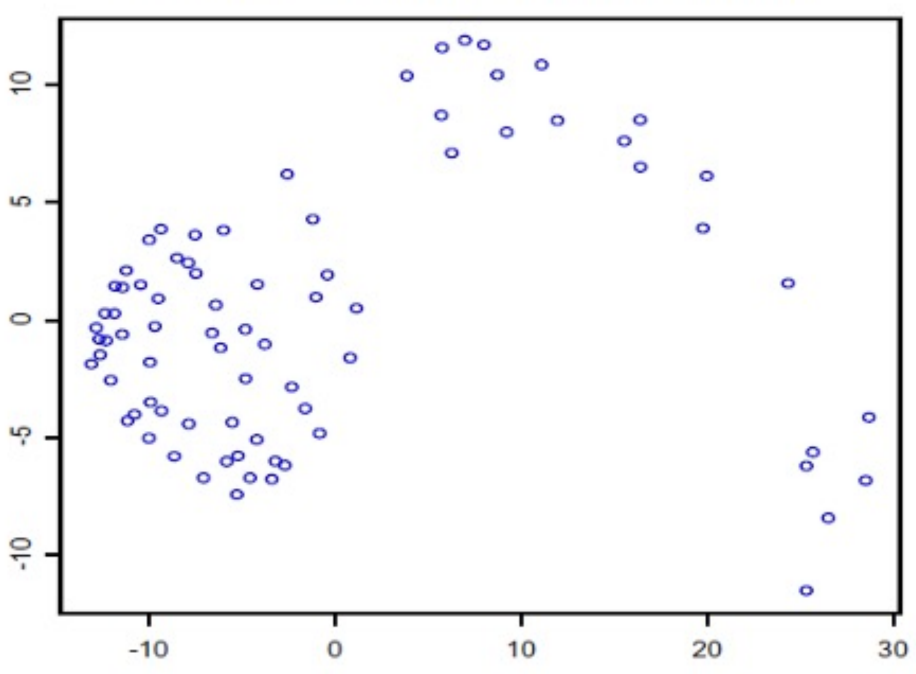
Sammon Mapping

- Sammon mapping is a generalization of the usual metric MDS.
- It introduces a weighting system that normalizes the squared-errors in pairwise distances by using the distance in the original space.
- $J(x) = \sum_i^n \sum_j^n d_1(d_{ij}, d_2(x_i, x_j))/d_{ij}$
- As a result, Sammon mapping preserves the small d_{ij}, giving them a greater degree of importance in the fitting procedure than for larger values of d_{ij}

Classic-MDS vs Sammon Mapping

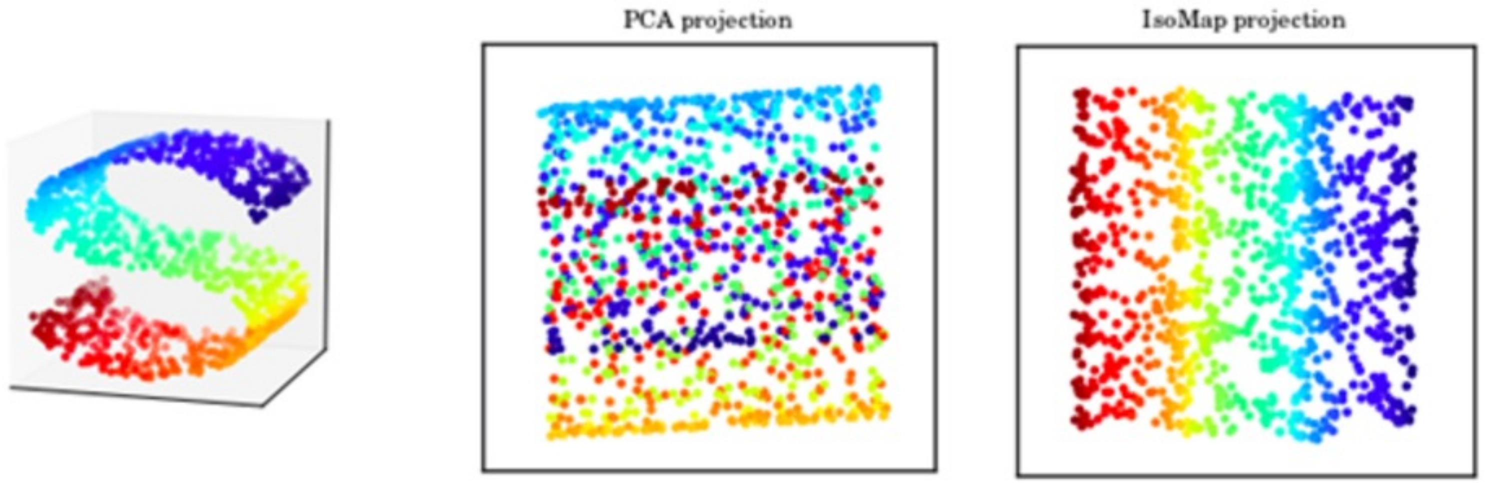
- Sammon mapping better preserves inter-distances for smaller dissimilarities, while proportionally squeezes the inter-distances for larger dissimilarities.





Isometric Feature Mapping (IsoMap)

- Preserves the intrinsic geometry of the data
- Uses the geodesic manifold distances between all pairs.
- It is a MDS method.
- IsoMap Handles non-linear manifold



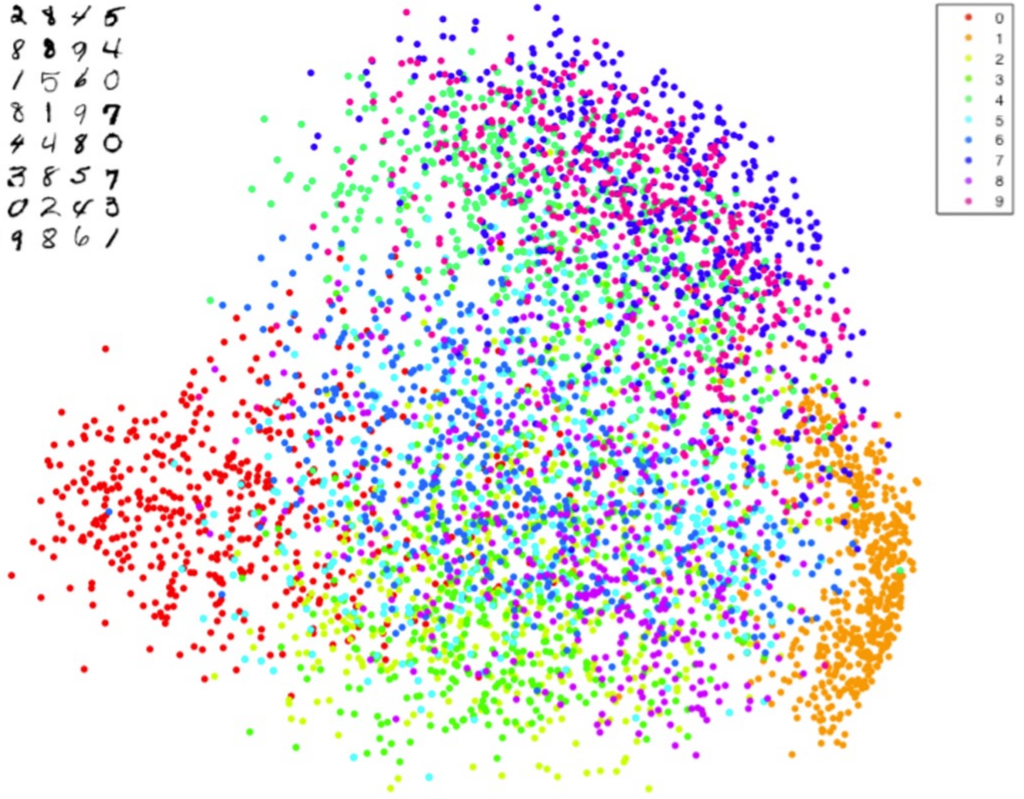
IsoMap Algorithm

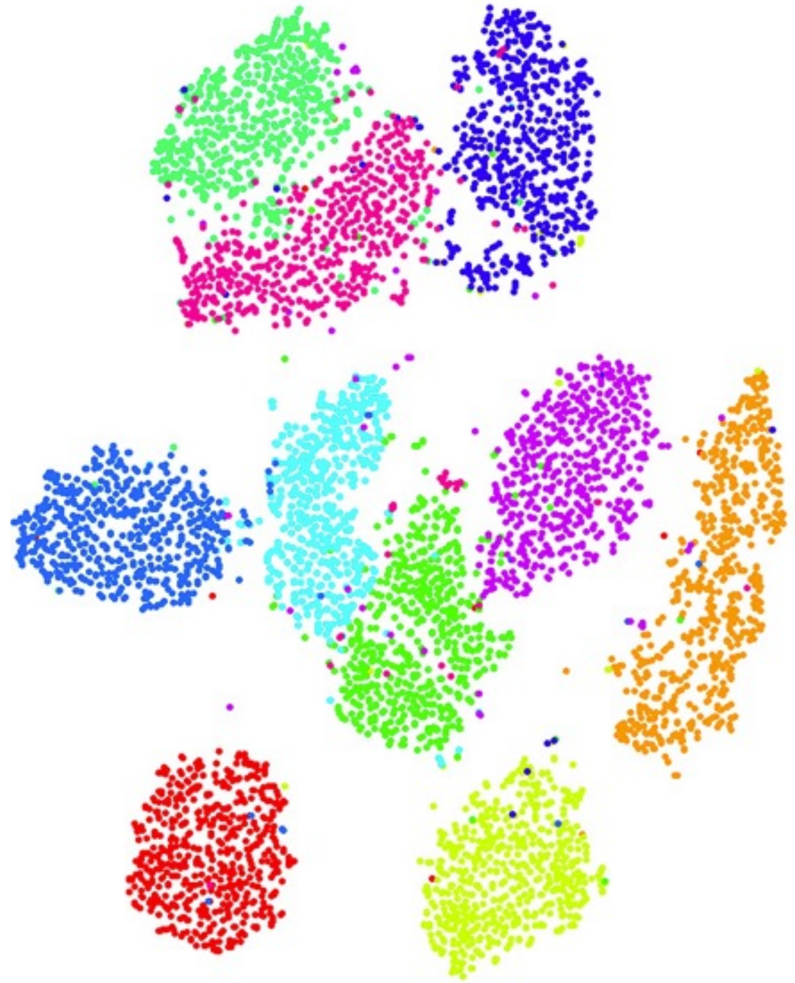
- Step 1
  - Determine neighboring points within a fixed radius based on the input space distance (Euclidean)
  - These neighborhood relations are represented as weighted graph *G* over the data points.
- Step 2
  - Estimate the geodesic distance between all pairs of points on the manifold by computing their shortest path distances on the graph *G*
- Step 3
  - Construct an embedding of the data in a *k* dimensional Euclidean space that best preserves the manifold geometry

t-Distributed Stochastic Neighbor Embedding (t-SNE)

- PCA tries to find a global structure
  - Low dimensional subspace
  - Can lead to local inconsistencies
    - Far away points can become neighbors
- t-SNE tries to preserve local structure
  - Local dimensional neighborhood should be the same as original neighborhood
  - Distance Preservation
  - Neighbor Preservation
- Unlike PCA almost only used for visualization

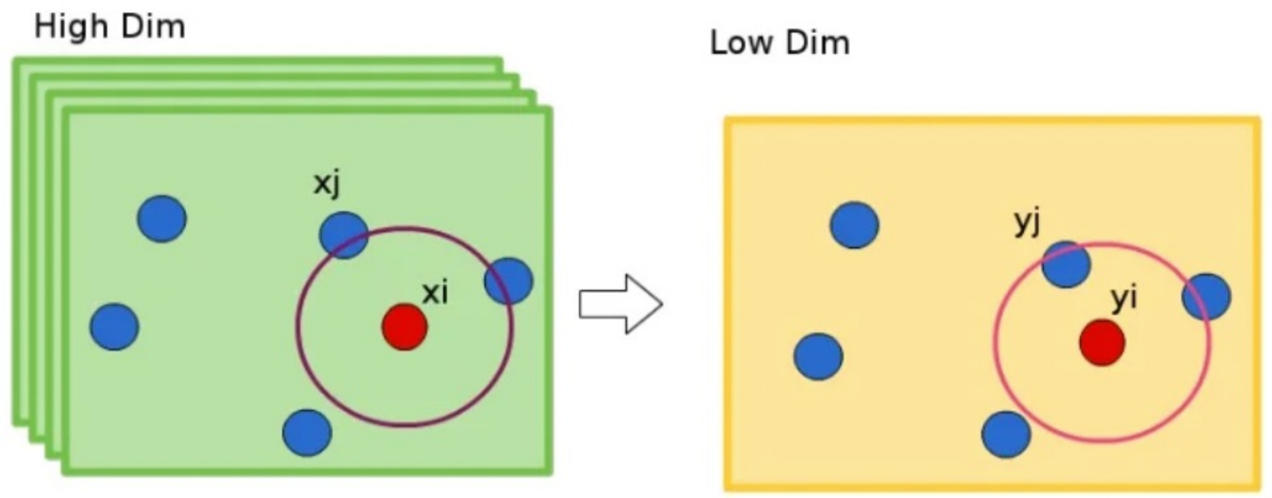
PCA vs t-SNE





SNE Intuition

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



Stochastic Neighbor Embedding (SNE)

- Encode high dimensional neighborhood information as a distribution
- Intuition: Random walk between data points.
  - High probability to jump to a close point
- Find low dimensional points such that their neighborhood distribution is similar.
- How do you measure distance between distributions?
  - Most common measure: KL divergence

Neighborhood Distributions

- Consider the neighborhood around an input data point x_i
- Imagine that we have a Gaussian distribution centered around x_i
- Then the probability that x_i chooses some other datapoint x_j as its neighbor is in proportion with the density under this Gaussian
- A point closer to x_i will be more likely than one further away

Probabilities

- This $p_{j|i}$ probability is the probability that point x_i chooses x_j as its neighbor

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

- The parameter sigma sets the size of the neighborhood
  - Very low sigma -> all the probability is in the newest neighbor
  - Very high sima -> uniform weights
- We set sigma differently for each data point
- Results depend heavily on sigma as it defines the neighborhood we are trying to preserve
- The final distribution over pairs is symmetrized $p_{ij} = 1/2N(p_{i|j} + p_{j|i})$

Perplexity

- For each distribution $p_{j|i}$ depends on sigma we define the perplexity
  - $perp(p_{j|i}) = 2^{H(pj|i)}$ where $H(p) = - \Sigma\, p \log(p)$ is the entropy
- If p is uniform over k elements perplexity is k
  - Smooth version of k in kNN
  - Low perplexity equals to small sigma
  - High perplexity equals to large sigma
  - Typically values of sigma between 5-50 work well
- Important parameter that can capture different scales in the data

SNE objective

- Given $x_1, \ldots, x_n \in R^m$ define the distribution p_{ij}
- Goal: find good embedding $y_1, .., y_n \in R^k$ *for* $k < n$
- How do we measure embedding quality?
- For points $y_1, .., y_n$ we can define distribution q similarly to the same (but not sigma and not symmetric)

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

- The idea is to optimize q to be close to p by minimizing the KL-divergence
- The embeddings $y_1, .., y_n$ are the parameters we are optimizing

KL-divergence

- Measures distance between two distributions, P and Q

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

- It is not a metric function as is not symmetric
- Based on the information theory intuition: if we are transmitting information distributed according to *p* then the optimal lossless compression will need to send on average *H(p)* bits
- Thus, *K(P||Q)* is the penalty for using a wrong distribution

Distances to Conditional Probabilities

- Converting the high-dimensional Euclidean distances into conditional probabilities that represent similarities
- Similarities 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 *C* using gradient descent

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

SNE problems

- Not a convex problem! No guarantees, can use multiple restarts.
- Crowding problem
  - In high dim we have a lot of different neighbors
  - In 2 dimensions we have few neighbors at the same distance and far from each other
  - Thus, we do not have space to accommodate all neighbors

- t-SNE solution: change the Gaussian in *Q* to a heavy tailed distribution

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

Student-t Probability Density

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

```
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for $t=1$ to $T$ do
        compute low-dimensional affinities $q_{ij}$ (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left( \mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
    end
end
```

The basic algorithm …

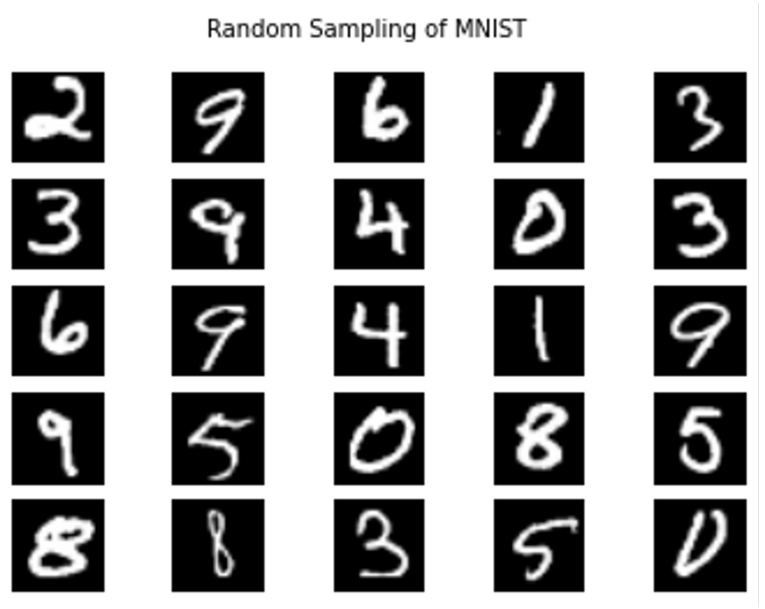
Complete slides available here: https://kawahara.ca/visualizing-data-using-t-sne-slides/

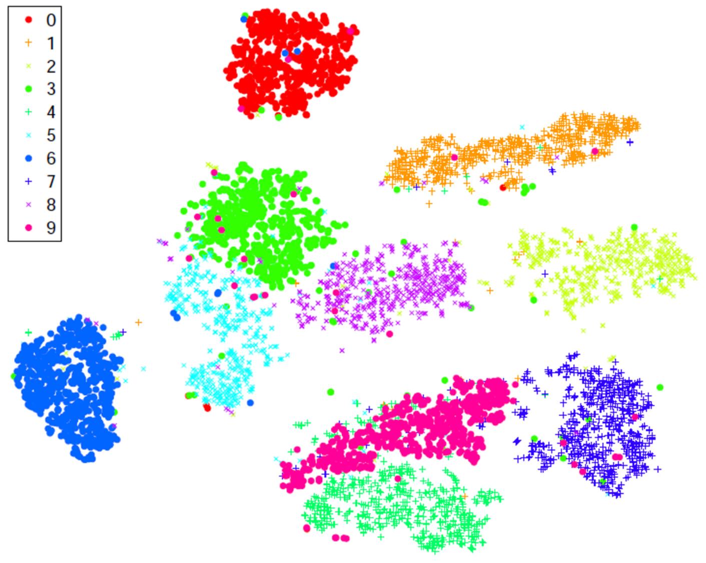
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

```
Data: data set $X = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for $t=1$ to $T$ do
        compute low-dimensional affinities $q_{ij}$ (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left( \mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
    end
end
```

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

```
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for $t$=$1$ to $T$ do
        compute low-dimensional affinities $q_{ij}$ (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
    end
end
```

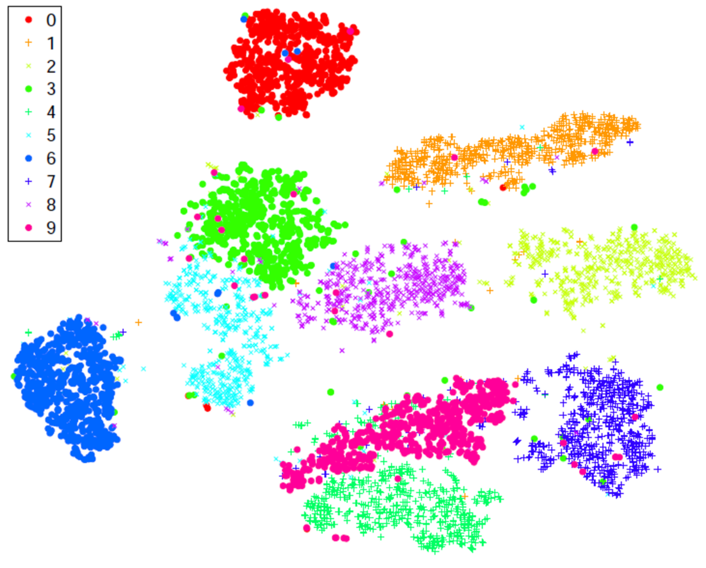




(a) Visualization by t-SNE.

```
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for $t=1$ to $T$ do
        compute low-dimensional affinities $q_{ij}$ (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
    end
end
```

Compute probabilities ***P*** that ***xi*** and ***xj*** are neighbors (based on Euclidian distance in **high-d** space)

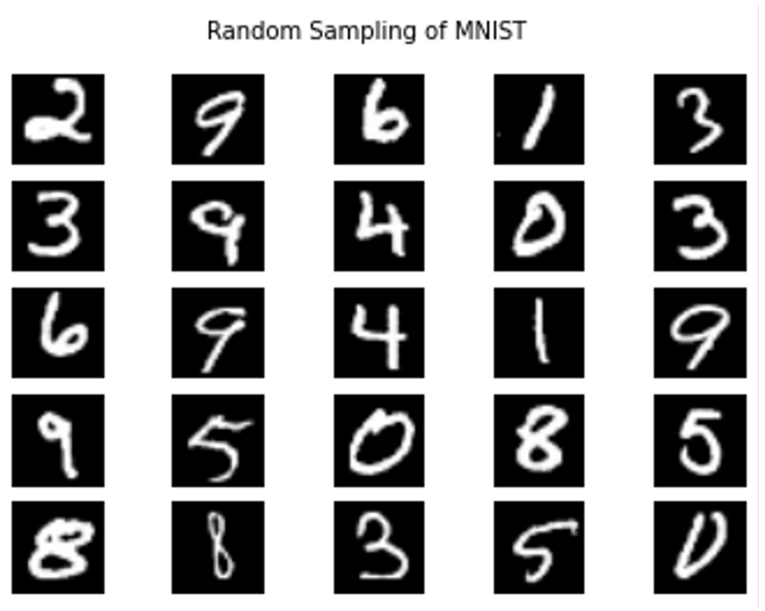


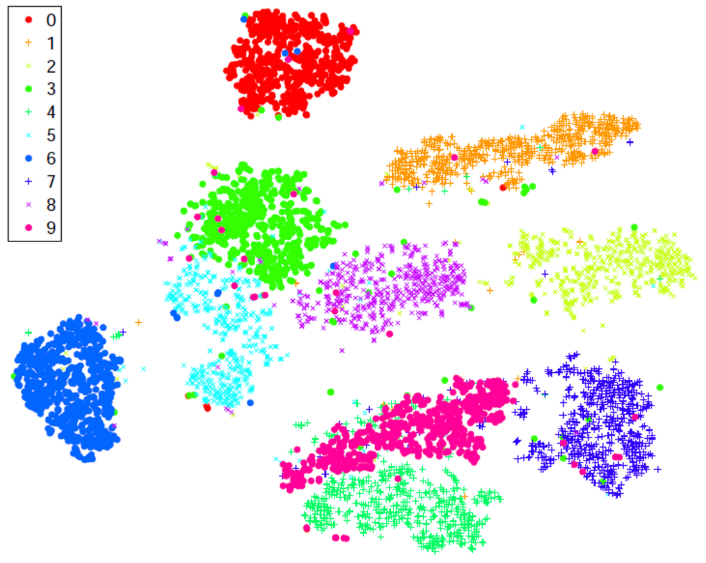
(a) Visualization by t-SNE.

Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

```
Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, ..., y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, ..., y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for $t$=1 to $T$ do
        compute low-dimensional affinities $q_{ij}$ (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$
    end
end
```

Key assumption is that the **high-d** *P* and the **low-d** *Q* probability distributions should be the same





(a) Visualization by t-SNE.

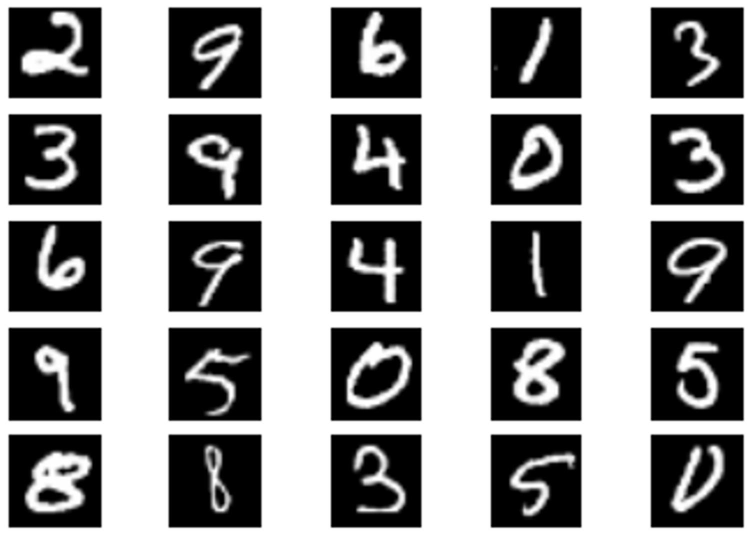
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

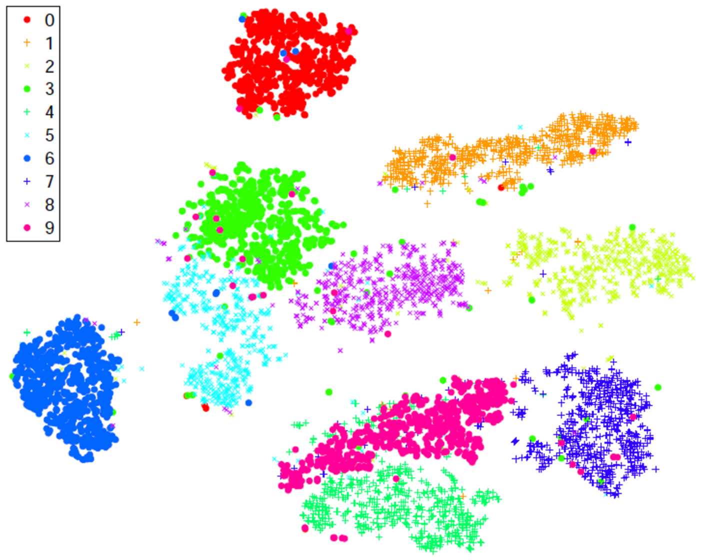
```
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i}+p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for t=1 to T do
        compute low-dimensional affinities $q_{ij}$ (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left( \mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
    end
end
```

Find a **low-d** map that minimizes the difference between the ***P*** (high-d) and ***Q*** (low-d) distributions

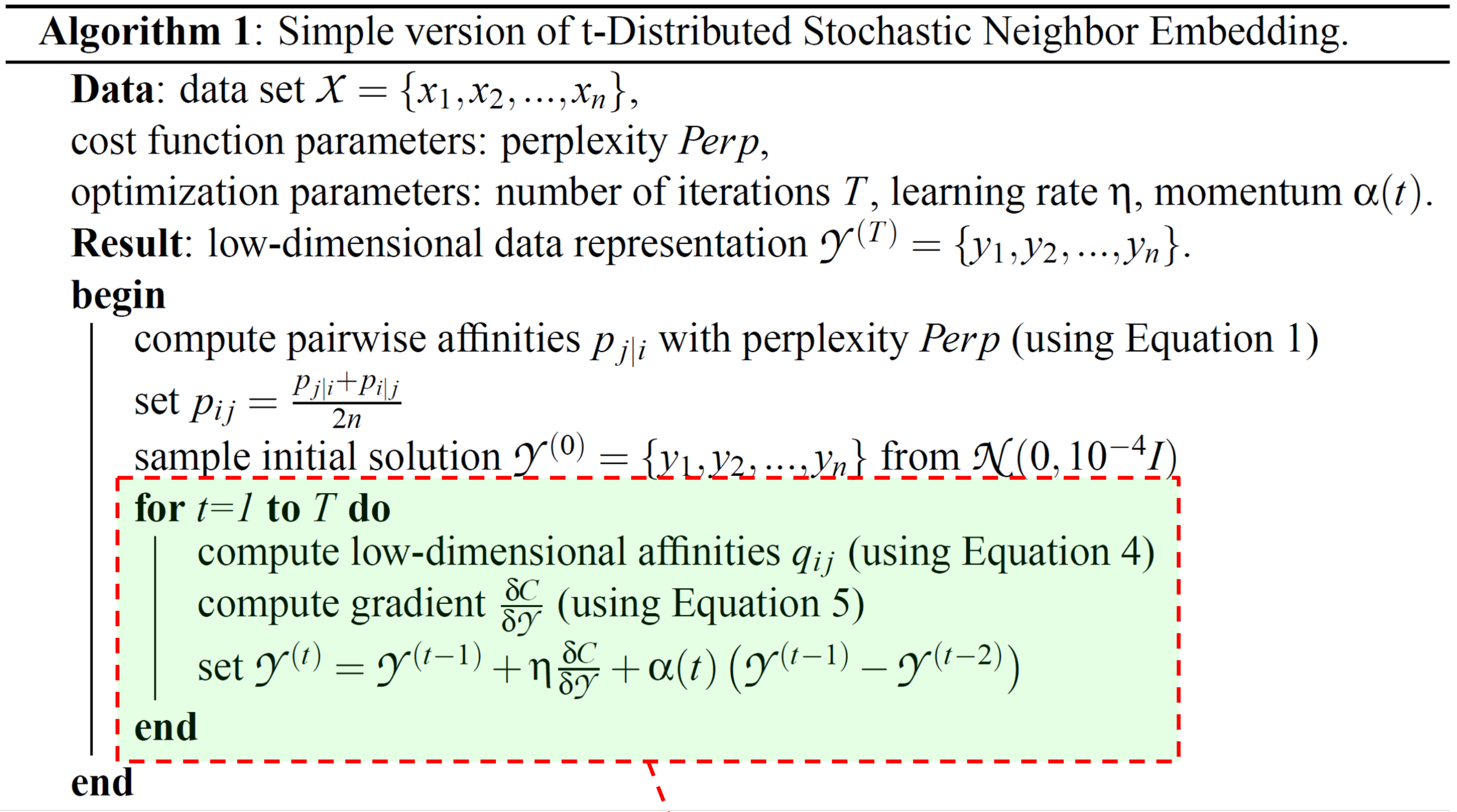
(if *xi,xj* has high probability of being neighbors in **high-d**, then *yi,yj* should have high probability in **low-d**)

Random Sampling of MNIST





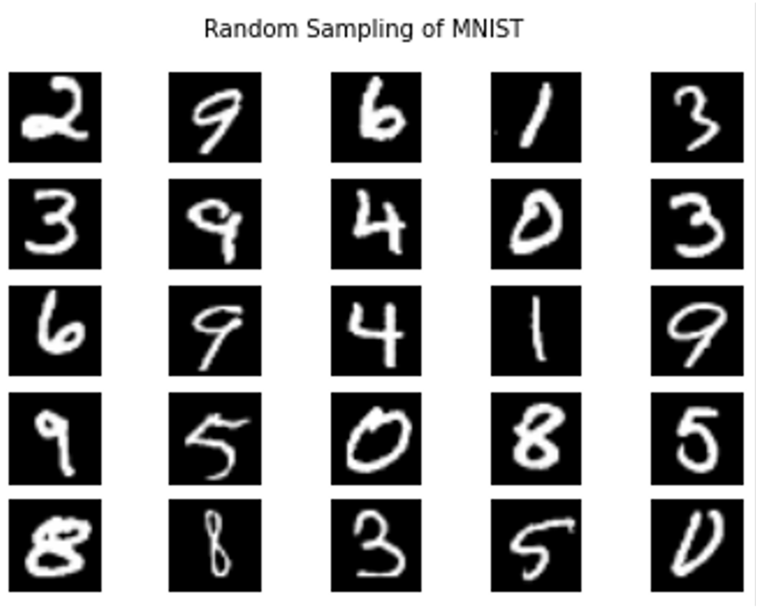
(a) Visualization by t-SNE.

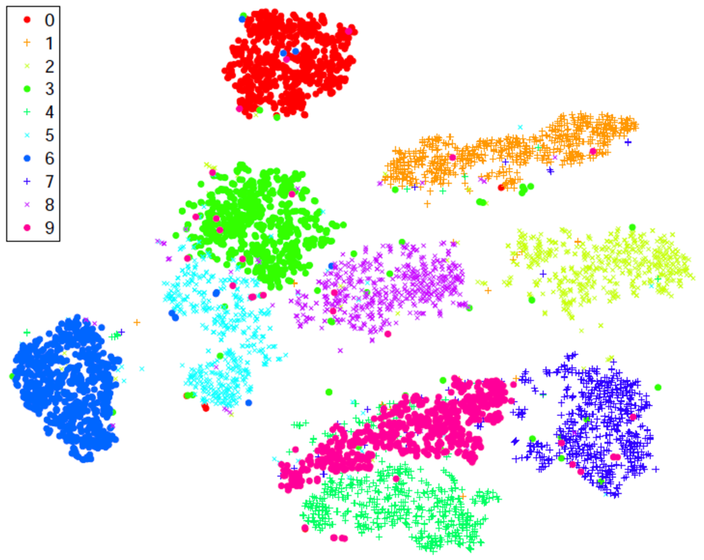


Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

```
Data: data set $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$,
cost function parameters: perplexity $Perp$,
optimization parameters: number of iterations $T$, learning rate $\eta$, momentum $\alpha(t)$.
Result: low-dimensional data representation $\mathcal{Y}^{(T)} = \{y_1, y_2, \ldots, y_n\}$.
begin
    compute pairwise affinities $p_{j|i}$ with perplexity $Perp$ (using Equation 1)
    set $p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
    sample initial solution $\mathcal{Y}^{(0)} = \{y_1, y_2, \ldots, y_n\}$ from $\mathcal{N}(0, 10^{-4}I)$
    for $t=1$ to $T$ do
        compute low-dimensional affinities $q_{ij}$ (using Equation 4)
        compute gradient $\frac{\delta C}{\delta \mathcal{Y}}$ (using Equation 5)
        set $\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t) \left( \mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)} \right)$
    end
end
```

We will minimize the difference between the **high-d** and **low-d** maps using **gradient descent**





(a) Visualization by t-SNE.

References

- Dimensionality Reduction. Appendix B. Introduction to Data Mining.
- Cox, T.F.; Cox, M.A.A. (2001). Multidimensional Scaling. Chapman and Hall.
- Maaten, L. van der, & Hinton, G. (2008). Visualizing Data using t-SNE. Journal of Machine Learning Research (JMLR), 9, 2579–2605. [pdf]
- J. B. Tenenbaum, V. de Silva, J. C. Langford, A Global Geometric Framework for Nonlinear Dimensionality Reduction, Science 290, (2000), 2319–2323. [pdf]