Machine Learning II

Session 2 : Dimensionality Reduction

Mohamad GHASSANY

EFREI PARIS

Course Overview

Course Information

Format:

- Course sessions: 6 sessions of 5 hours each.
- Sessions are CTP.

Course chapters:

Session 1: Supervised Learning: Discriminant Analysis

Session 2: Dimensionality Reduction

Session 3:

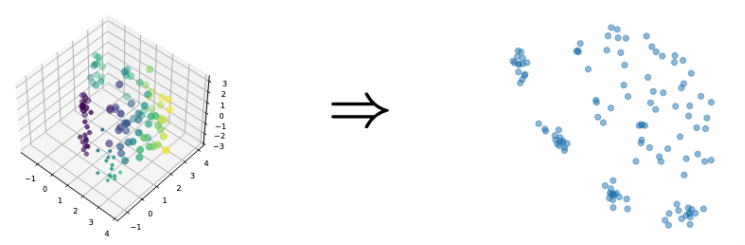
Session 4:

Session 5:

Session 6:

Definition

Dimensionality reduction, or dimension reduction, is the transformation of data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data, ideally close to its intrinsic dimension[1].



[1]The intrinsic dimension for a data set can be thought of as the number of variables needed in a minimal representation of the data.

Dimensionality Reduction methods

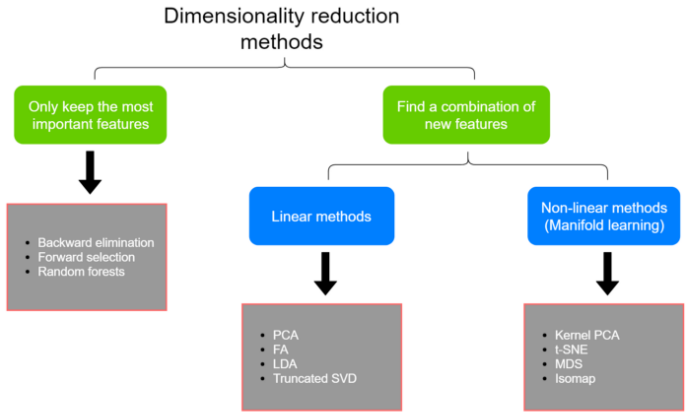


Image copyright: Rukshan Pramoditha

Dimensionality Reduction Methods: Principal Components Analysis (PCA)

PCA: Introduction

- Central idea: reduce the dimensionality of a data set consisting of a large number of interrelated variables, while retaining as much as possible of the ***variation*** present in the data set.
- How is this achieved: by *transforming* to a new set of variables, the principal components (PCs), which are uncorrelated, and which are ordered so that the first few retain most of the variation present in all of the original variables.
- Each of the dimensions found by PCA is a **linear combination of the** p **features**.

Suppose that we have a random vector of the features X with population variance-covariance matrix Σ

$$\mathbf{X} = \begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_p \end{pmatrix} \qquad \text{var}(\mathbf{X}) = \Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1p} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \dots & \sigma_p^2 \end{pmatrix}$$

Consider the linear combinations

$$\begin{aligned} Y_1 &= a_{11}X_1 + a_{12}X_2 + \cdots + a_{1p}X_p \\ Y_2 &= a_{21}X_1 + a_{22}X_2 + \cdots + a_{2p}X_p \\ &\vdots \\ Y_p &= a_{p1}X_1 + a_{p2}X_2 + \cdots + a_{pp}X_p \end{aligned}$$

The coefficients a_{ij} are collected into the vector

$$\mathbf{a}_i = \begin{pmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{ip} \end{pmatrix}$$

The coefficients a_{ij} are also called *loadings* of the principal component i and $\mathbf{a}_i$ is a principal component loading vector.

Note that Y_i is a function of our random data, and so is also random. Therefore it has a population variance

$$\text{var}(Y_i) = \sum_{k=1}^{p} \sum_{l=1}^{p} a_{ik} a_{il} \sigma_{kl} = \mathbf{a}_i^T \Sigma \mathbf{a}_i$$

Moreover, Y_i and Y_j will have a population covariance

$$\text{cov}(Y_i, Y_j) = \sum_{k=1}^{p} \sum_{l=1}^{p} a_{ik} a_{jl} \sigma_{kl} = \mathbf{a}_i^T \Sigma \mathbf{a}_j$$

and a correlation

$$\text{cor}(Y_i, Y_j) = \frac{\text{cov}(Y_i, Y_j)}{\sigma_i^2 \sigma_j^2}$$

Variation and trace

- The total variation of X is the *trace*[2] of the variance-covariance matrix Σ.
- The trace of Σ is the sum of the variances of the individual variables.
- $trace(\Sigma) = \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_p^2$

$$var(\mathbf{X}) = \Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1p} \\ \sigma_{21} & \sigma_2^2 & \dots & \sigma_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1} & \sigma_{p2} & \dots & \sigma_p^2 \end{pmatrix}$$

[2] The trace of a square matrix is the sum of its diagonal entries.

First Principal Component (PC_1): Y_1

- The *first principal component* is the *normalized* linear combination of the features $X_1, X_2, \ldots, X_p$ that has maximum variance (among all linear combinations), so it accounts for as much variation in the data as possible.
- Specifically we will define coefficients $a_{11}, a_{12}, \ldots, a_{1p}$ for that component in such a way that its variance is maximized, subject to the constraint that the sum of the squared coefficients is equal to one (that is what we mean by *normalized*). This constraint is required so that a unique answer may be obtained.
- More formally, select $a_{11}, a_{12}, \ldots, a_{1p}$ that maximizes

$$\text{var}(Y_1) = \mathbf{a}_1^T \Sigma \mathbf{a}_1 = \sum_{k=1}^{p} \sum_{l=1}^{p} a_{1k} a_{1l} \sigma_{kl}$$

subject to the constraint that

$$\sum_{j=1}^{p} a_{1j}^2 = \mathbf{a}_1^T \mathbf{a}_1 = 1$$

Second Principal Component (PC_2): Y_2

- The *second principal component* is the linear combination of the features $X_1, X_2, \ldots, X_p$ that accounts for as much of the remaining variation as possible, with the constraint that the **correlation between the first and second component is 0**.
- To compute the coefficients of the second principal component, we select $a_{21}, a_{22}, \ldots, a_{2p}$ that maximizes the variance of this new component $\text{var}(Y_2) = \mathbf{a}_2^T \Sigma \mathbf{a}_2$ subject to:
  - The constraint that the sums of squared coefficients add up to one, $\sum_{j=1}^{p} a_{2j}^2 = \mathbf{a}_2^T \mathbf{a}_2 = 1$.
  - Along with the additional constraint that these two components will be uncorrelated with one another: $\text{cov}(Y_1, Y_2) = 0$

i^{th} Principal Component (PC_i): Y_i

- All subsequent principal components have this same property: they are linear combinations that account for as much of the remaining variation as possible and they are not correlated with the other principal components.
- We select $a_{i1}, a_{i2}, \ldots, a_{ip}$ that maximizes $\text{var}(Y_i)$

subject to the constraint that the sums of squared coefficients add up to one, along with the additional constraint that this new component will be uncorrelated with all the previously defined components:

$$\sum_{j=1}^{p} a_{ij}^2 \mathbf{a}_i^T \mathbf{a}_i = \mathbf{a}_i^T \mathbf{a}_i = 1$$

$$\text{cov}(Y_1, Y_i) = 0, \quad \text{cov}(Y_2, Y_i) = 0, \quad \ldots \quad , \quad \text{cov}(Y_{i-1}, Y_i) = 0$$

Therefore all principal components are uncorrelated with one another.

How do we find the coefficients a_{ij}?

- How do we find the coefficients a_{ij} (*loadings*) for a principal component?
- The solution involves the **eigenvalues** and **eigenvectors** of the variance-covariance matrix Σ.

How do we find the coefficients a_{ij}?

- How do we find the coefficients a_{ij} (*loadings*) for a principal component?
- The solution involves the **eigenvalues** and **eigenvectors** of the variance-covariance matrix Σ.
- Let $\lambda_1, \ldots, \lambda_p$ denote the eigenvalues of the variance-covariance matrix Σ. These are ordered so that λ_1 has the largest eigenvalue and λ_p is the smallest: $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_p$
- We are also going to let the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_p$ denote the corresponding eigenvectors.

How do we find the coefficients a_{ij}?

- How do we find the coefficients a_{ij} (*loadings*) for a principal component?
- The solution involves the **eigenvalues** and **eigenvectors** of the variance-covariance matrix Σ.
- Let $\lambda_1, \ldots, \lambda_p$ denote the eigenvalues of the variance-covariance matrix Σ. These are ordered so that λ_1 has the largest eigenvalue and λ_p is the smallest: $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_p$
- We are also going to let the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_p$ denote the corresponding eigenvectors.
- **It turns out that the elements for these eigenvectors will be the coefficients of the principal components**.

How do we find the coefficients a_{ij}?

- How do we find the coefficients a_{ij} (*loadings*) for a principal component?
- The solution involves the **eigenvalues** and **eigenvectors** of the variance-covariance matrix Σ.
- Let $\lambda_1, \ldots, \lambda_p$ denote the eigenvalues of the variance-covariance matrix Σ. These are ordered so that λ_1 has the largest eigenvalue and λ_p is the smallest: $\lambda_1 \geqslant \lambda_2 \geqslant \cdots \geqslant \lambda_p$
- We are also going to let the vectors $\mathbf{a}_1, \ldots, \mathbf{a}_p$ denote the corresponding eigenvectors.
- **It turns out that the elements for these eigenvectors will be the coefficients of the principal components**.
- The variance for the i-th principal component is equal to the i-th eigenvalue.

$$\textbf{var}(Y_i) = \text{var}(a_{i1}X_1 + a_{i2}X_2 + \ldots a_{ip}X_p) = \lambda_i$$

- Moreover, the principal components are uncorrelated with one another: $\text{cov}(Y_i, Y_j) = 0$

- Earlier in the chapter we defined the total variation of X as the trace of the variance-covariance matrix. **This is also equal to the sum of the eigenvalues** as shown below:

$$\begin{aligned} \text{trace}(\Sigma) &= \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_p^2 \\ &= \lambda_1 + \lambda_2 + \cdots + \lambda_p \end{aligned}$$

- This will give us an interpretation of the components in terms of the amount of the full variation explained by each component.

[3] In other words, the ith principal component explains the following proportion of the total variation

- Earlier in the chapter we defined the total variation of X as the trace of the variance-covariance matrix. **This is also equal to the sum of the eigenvalues** as shown below:

$$\begin{aligned} \text{trace}(\Sigma) &= \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_p^2 \\ &= \lambda_1 + \lambda_2 + \cdots + \lambda_p \end{aligned}$$

- This will give us an interpretation of the components in terms of the amount of the full variation explained by each component.
- The proportion of variation explained by the ith principal component[3] is then going to be defined as: $\dfrac{\lambda_i}{\lambda_1 + \lambda_2 + \cdots + \lambda_p}$

[3] In other words, the ith principal component explains the following proportion of the total variation

Proportion of the total variation

- Earlier in the chapter we defined the total variation of X as the trace of the variance-covariance matrix. **This is also equal to the sum of the eigenvalues** as shown below:

$$\begin{aligned} \text{trace}(\Sigma) &= \sigma_1^2 + \sigma_2^2 + \cdots + \sigma_p^2 \\ &= \lambda_1 + \lambda_2 + \cdots + \lambda_p \end{aligned}$$

- This will give us an interpretation of the components in terms of the amount of the full variation explained by each component.
- The proportion of variation explained by the ith principal component[3] is then going to be defined as: $\dfrac{\lambda_i}{\lambda_1 + \lambda_2 + \cdots + \lambda_p}$
- A related quantity is the proportion of variation explained by the first k principal component: $\dfrac{\lambda_1 + \lambda_2 + \cdots + \lambda_k}{\lambda_1 + \lambda_2 + \cdots + \lambda_p}$
- Naturally, if the proportion of variation explained by the first k principal components is large, then not much information is lost by considering only the first k principal components.

[3] In other words, the ith principal component explains the following proportion of the total variation

- Compute the eigenvalues $\hat{\lambda}_1, \hat{\lambda}_2, \ldots, \hat{\lambda}_p$ of the sample variance-covariance matrix $\mathbf{S}$, and the corresponding eigenvectors $\hat{\mathbf{a}}_1, \hat{\mathbf{a}}_2, \ldots, \hat{\mathbf{a}}_p$.
- Then we will define our estimated principal components using the eigenvectors as our coefficients:

$$\begin{aligned} \hat{Y}_1 &= \hat{a}_{11}X_1 + \hat{a}_{12}X_2 + \cdots + \hat{a}_{1p}X_p \\ \hat{Y}_2 &= \hat{a}_{21}X_1 + \hat{a}_{22}X_2 + \cdots + \hat{a}_{2p}X_p \\ &\vdots \\ \hat{Y}_p &= \hat{a}_{p1}X_1 + \hat{a}_{p2}X_2 + \cdots + \hat{a}_{pp}X_p \end{aligned}$$

- This can be written for all observations and all the principal components using the matrix formulation $\hat{\mathbf{Y}} = \hat{\mathbf{A}}X$ where $\hat{\mathbf{A}}$ is the matrix of the coefficients $\hat{a}_{ij}$.
- Generally, we only retain the first k principal component which explain a "large" proportion of the total variation.

Standardization of the features

- If we use the raw data, the principal component analysis will tend to give more emphasis to the variables that have higher variances than to those variables that have very low variances.

Standardization of the features

- If we use the raw data, the principal component analysis will tend to give more emphasis to the variables that have higher variances than to those variables that have very low variances.
- If the variables either have different units of measurement they should be **standardized** (*scaled*) before a principal components analysis is carried out:

$$Z_{ij} = \frac{X_{ij} - \bar{x}_j}{\sigma_j}$$

Note: Z_j has mean $= 0$ and variance $= 1$.

- If we use the raw data, the principal component analysis will tend to give more emphasis to the variables that have higher variances than to those variables that have very low variances.
- If the variables either have different units of measurement they should be **standardized** (*scaled*) before a principal components analysis is carried out:

$$Z_{ij} = \frac{X_{ij} - \bar{x}_j}{\sigma_j}$$

Note: Z_j has mean $= 0$ and variance $= 1$.

Variance-covariance matrix and correlation matrix

- The variance-covariance matrix of the standardized data is equal to the correlation matrix for the unstandardized data.
- Therefore, principal component analysis using the standardized data is equivalent to principal component analysis using the correlation matrix.

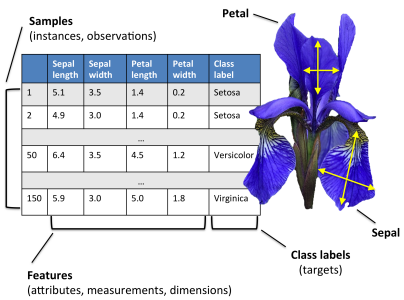
The ***Iris flower dataset*** or ***Fisher's Iris dataset*** is a multivariate data set introduced by the British statistician and biologist Ronald Fisher in his **1936** paper. The data set consists of 50 samples from each of three species of Iris. Four features were measured from each sample.

The three species in the Iris dataset are:

1. *Iris-setosa* ($n_1 = 50$)
2. *Iris-versicolor* ($n_2 = 50$)
3. *Iris-virginica* ($n_3 = 50$)

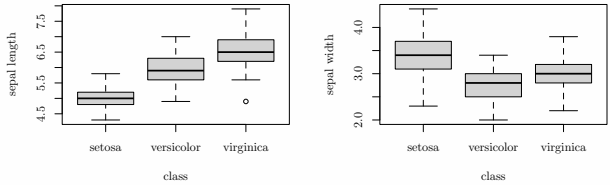
And the four features in Iris dataset are:

1. *sepal length* in cm
2. *sepal width* in cm
3. *petal length* in cm
4. *petal width* in cm



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

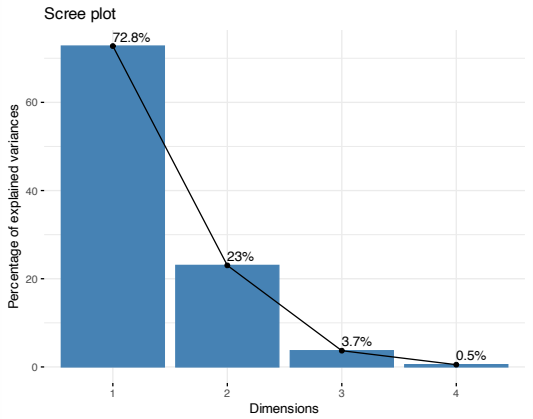
Case Study: the Iris Dataset



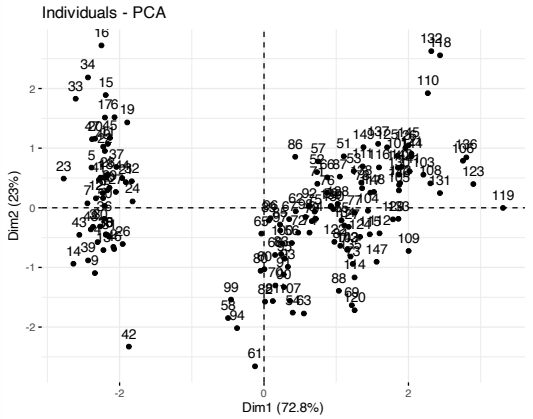
```
pcairis=princomp(iris[,-5], cor=T)

summary(pcairis)
```

```
## Importance of components:
##                           Comp.1    Comp.2     Comp.3      Comp.4
## Standard deviation     1.7061120 0.9598025 0.38386622 0.143553848
## Proportion of Variance 0.7277045 0.2303052 0.03683832 0.005151927
## Cumulative Proportion  0.7277045 0.9580098 0.99484807 1.000000000
```

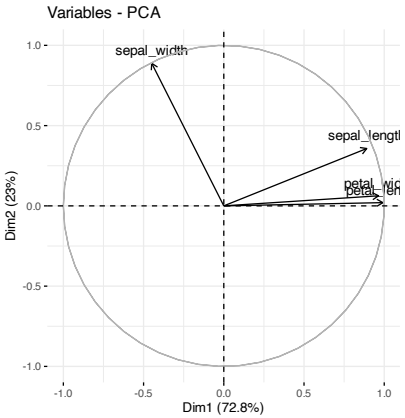
```
library(factoextra)
fviz_eig(pcairis, addlabels = TRUE)
```



```
fviz_pca_ind(pcairis)
```

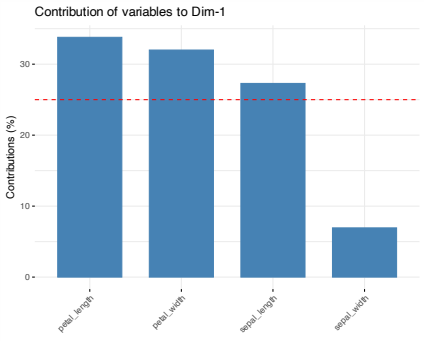


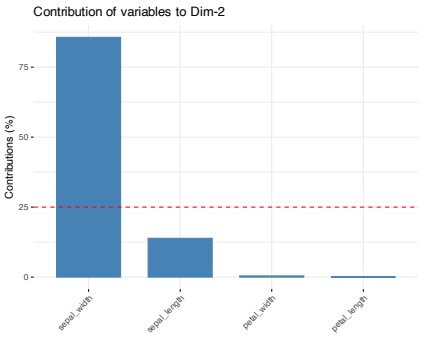
```
fviz_pca_var(pcairis, col.var = "black")
```

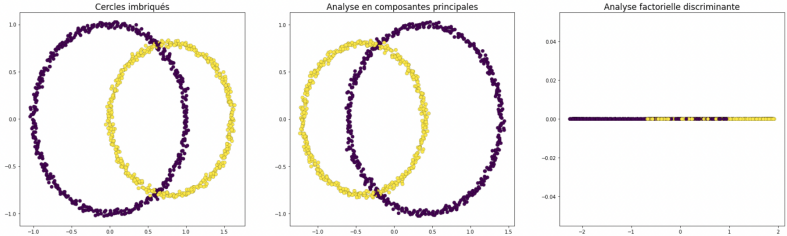


```
fviz_pca_biplot(pcairis)
```

Case Study: Contribution of variables







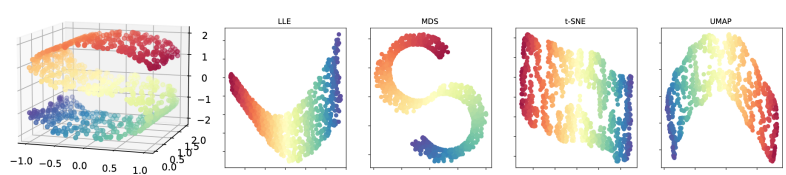
- There is no linear transformation that could separate the two circles.
- $\Rightarrow$ non linear methods.

Nonlinear dimension reduction methods (manifold learning)

What is a manifold ?

- In non technical terms, a manifold is a continuous geometrical structure having finite dimension: a line, a curve, a plane, a surface, a sphere, a ball, a cylinder, a torus, a "blob"...
- It is a generic term used by mathematicians to say "a curve" (dimension 1) or "surface" (dimension 2), or a 3D object (dimension 3)... for any possible finite dimension n
- A manifold is often described by an equation the set of points (x, y) such as $x^2 + y^2 = 1$ is a one dimensional manifold (a circle).
- In ML, "manifold hypothesis" says "high dimensional data are points in a low dimensional manifold with high dimensional noise added".

What is manifold learning ?



Nonlinear subspaces

- The dataset often lies in a nonlinear subspace (a manifold) of $\mathbb{R}^p$.
- Manifold learning method aim at recovering this low dimensional manifold.
- Example above of 2D manifold in a 3D ambient space and the projection of the samples in 2D for different methods (colors only to check that the relation between samples are preserved).

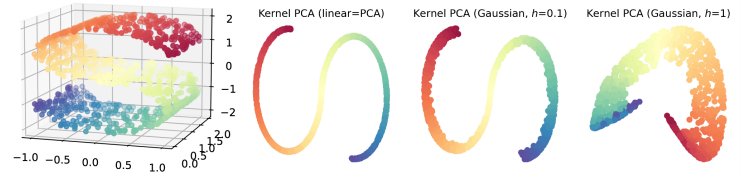
Manifold learning problems

- **Projection**: Project dataset in low dimension (visualization).
- **Inductive**: Learn a nonlinear projection function.
- **Inductive+Invertible**: Learn both projection and reconstruction nonlinear functions.

Manifold Learning methods

Contents [hide]

Kernel PCA (KPCA)



Principle

- Perform PCA in a high-dimensional non-linear embedding $\phi(x)$ of the data.
- Embedding is implicit, thanks to the use of a kernel $k(x, x\prime) =< \phi(x), \phi(x\prime) >$, only the kernel matrix between samples is necessary. This is called the "kernel trick" (used also for SVM classification).
- Inductive method, reconstruction is possible but requires solving an inverse problem.
- Classical kernels are linear kernel (equivalent to PCA) and Gaussian kernel (Radial Basis Function RBF in Sckikit-learn).
- LLE and ISOMAP are actually special cases of KPCA with specifically designed kernels.
- Scikit-learn implementation: `sklearn.decomposition.KernelPCA`.

T-distributed Stochastic Neighbor Embedding (t-SNE)

Definition

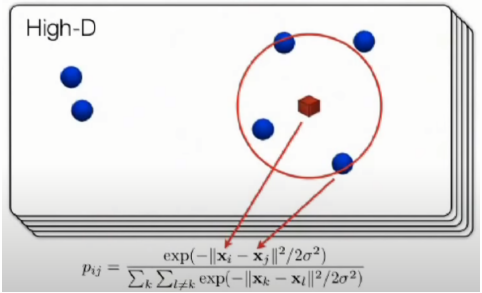
- t-SNE converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data.
- t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.

Year	Method	Author	Summary
1901	PCA	Karl Pearson	First dimensionality reduction technique
2000	Isomap	Tenenbaum, de Silva, and Langford	First non-linear dimensionality reduction technique
2002	SNE	Hinton and Roweis	Original SNE algorithm
2008	tSNE	Maaten and Hinton	Addressed the crowding issue of SNE, $O(N^2)$
2014	BHt-SNE	Maaten	Using BarnesHut approximation to achieve $O(N \log(N))$
2017		Linderman and Steinerberger	First step towards theoretical guarantee for t-SNE
2017	Fit-SNE	Linderman et al.	Acceleration to $O(N)$
2018		Arora et al.	Theoretical guarantee for t-SNE
2018		Verma et al.	Generalization of t-SNE to manifold

Brief history of SNE

- Suppose we have high-dimensional data set $X = \{x_1, x_2, \ldots, x_n\}$, and we want to reduce the dimension into two or three-dimensional data $Y = \{y_1, y_2, \ldots, y_n\}$ that can be displayed in a scatterplot.
- The low-dimensional data representation Y is referred as a **map**, and to the low-dimensional representations y_i of individual data points as **map points**.
- Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities.
- The similarity of datapoint x_j to datapoint x_i is the conditional probability, $p_{j|i}$, that x_i would pick x_j as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian centered at x_i
- For nearby datapoints, $p_{j|i}$ is relatively high, whereas for widely separated datapoints, $p_{j|i}$ will be almost infinitesimal.

Brief history of SNE



- σ_i is the variance of the Gaussian that is centered on datapoint x_i.
- The similarity of map point y_j to map point y_i is modelled by: $q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)}$
- SNE aims to find a low-dimensional data representation that minimizes the mismatch between $p_{j|i}$ and $q_{j|i}$.

Brief history of SNE

- SNE minimizes the sum of Kullback-Leibler divergences over all datapoints using a gradient descent method. The cost function C is given by: $C = \sum_i KL(P_i \| Q_i) = \sum_i \sum_j p_{j|i} \log \frac{p_{j|i}}{q_{j|i}}$
- $P_i = \{p_{1|i}, p_{2|i}, \ldots, p_{n|i}\}$ and $Q_i = \{q_{1|i}, q_{2|i}, \ldots, q_{n|i}\}$ are the distributions on the neighbors of datapoint i.

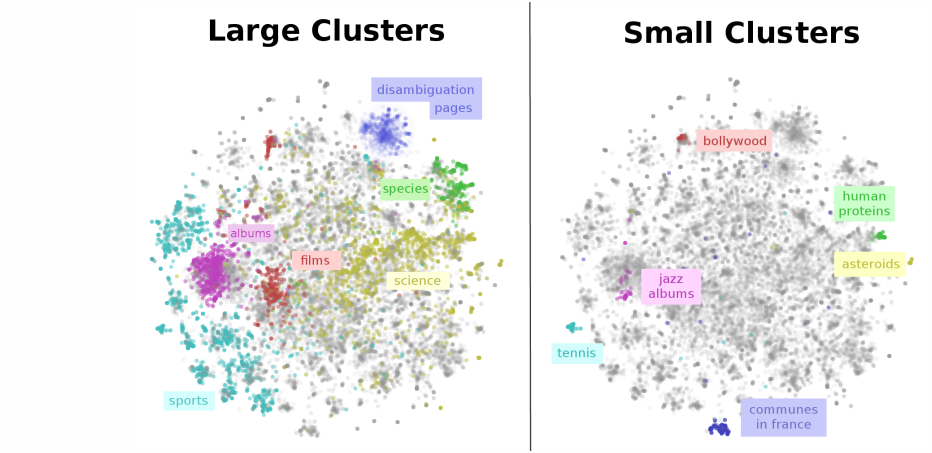
Selection of σ_i

- In dense regions, a smaller value of σ_i is usually more appropriate than in sparser regions.
- SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity that is specified by the user. The perplexity is defined as: $Perp(P_i) = 2^{H(P_i)}$ where $H(P_i)$ is the Shannon entropy of P_i: $H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i}$
- Therefore, for denser data, greater perplexity should be chosen, which would result in a smaller σ_i and neighborhood size.
- As well as SNE preserves local relationships, it suffers from the "crowding problem": The area of the 2D map that is available to accommodate moderately distant data points will not be large enough compared with the area available to accommodate nearby data points.

From SNE to t-SNE

- To address the crowding problem and make SNE more robust to outliers, t-SNE was introduced.
- Compared to SNE, t-SNE has two main changes:
  1. a symmetrized version of the SNE cost function with simpler gradients
  2. a Student-t distribution rather than a Gaussian to compute the similarity in the low-dimensional space to alleviate the crowding problem.

SNE	⇒	Symmetric SNE	⇒	t-SNE
Modelisation: $p_{j\|i} = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)}$ $q_{j\|i} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k\neq i}\exp(-\|y_i - y_k\|^2)}$		Modelisation: $p_{ij} = \frac{p_{j\|i} + p_{i\|j}}{2n}$ $q_{ij} = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k\neq l}\exp(-\|y_k - y_l\|^2)}$		Modelisation: $p_{ij} = \frac{p_{j\|i} + p_{i\|j}}{2n}$ $q_{ij} = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|y_k - y_l\|^2)^{-1}}$
Cost Function: $C = \sum_i KL(P_i\|\|Q_i)$		Cost Function: $C = KL(P\|\|Q)$		Cost Function: $C = KL(P\|\|Q)$
Derivatives: $\frac{\delta C}{\delta y_i} = 2\sum_j (p_{j\|i} - q_{j\|i} + p_{i\|j} - q_{i\|j})(y_i - y_j)$		Derivatives: $\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)$		Derivatives: $\frac{\delta C}{\delta y_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)(1+\| y_i - y_j \|^2)^{-1}$
		▸ **Faster Computation**		▸ **Even Faster Computation** ▸ **Better Behaviour**



from http://colah.github.io/