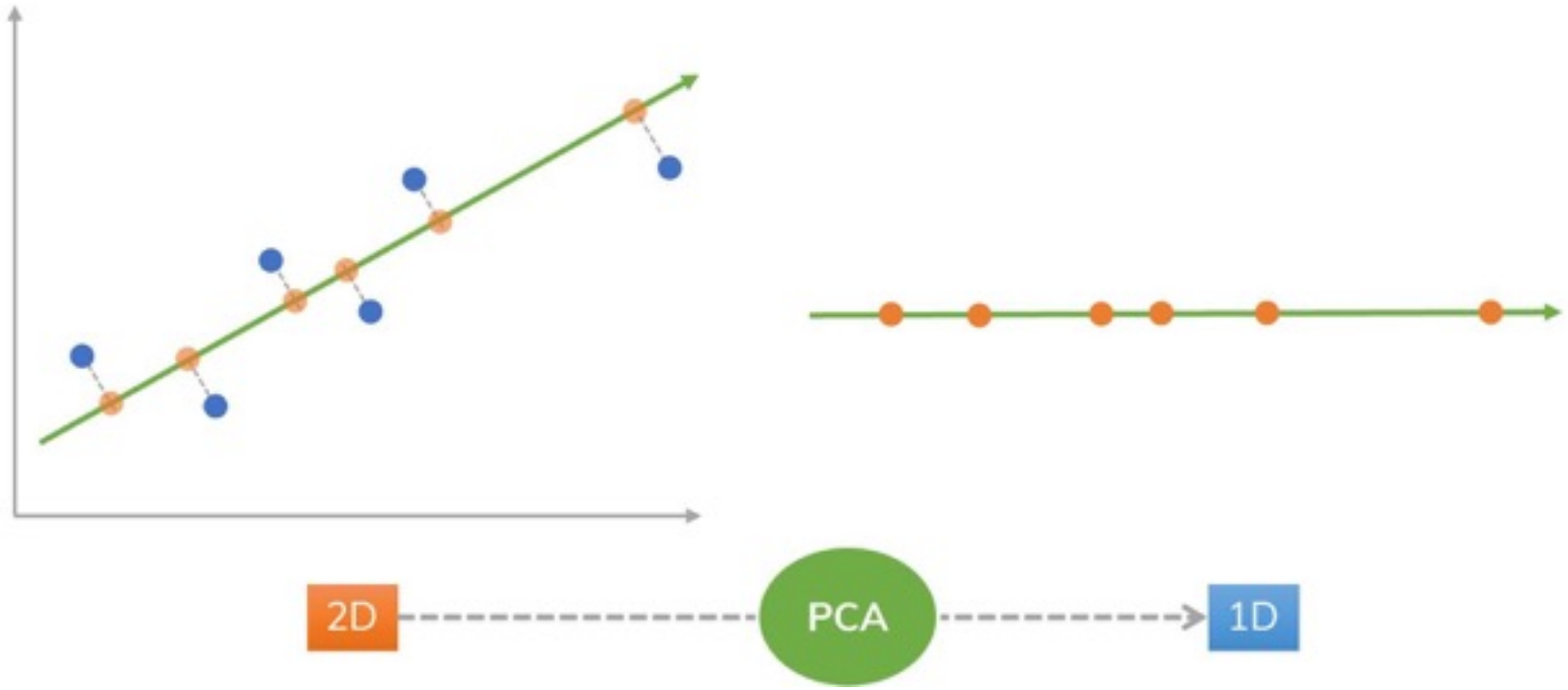
Dimensionality Reduction

Seyed Abbas Hosseini
Sharif University of Technology

Most slides are adopted from Soleymani CE-717

Dimensionality Reduction

- **Dimensionality 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.



Feature Selection vs. Extraction

- Feature Selection
  - select a subset of a given feature set
- Feature Extraction
  - A linear or nonlinear transformation on original feature set

$$\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \rightarrow \begin{bmatrix} x_{i_1} \\ \vdots \\ x_{i_{d'}} \end{bmatrix}$$

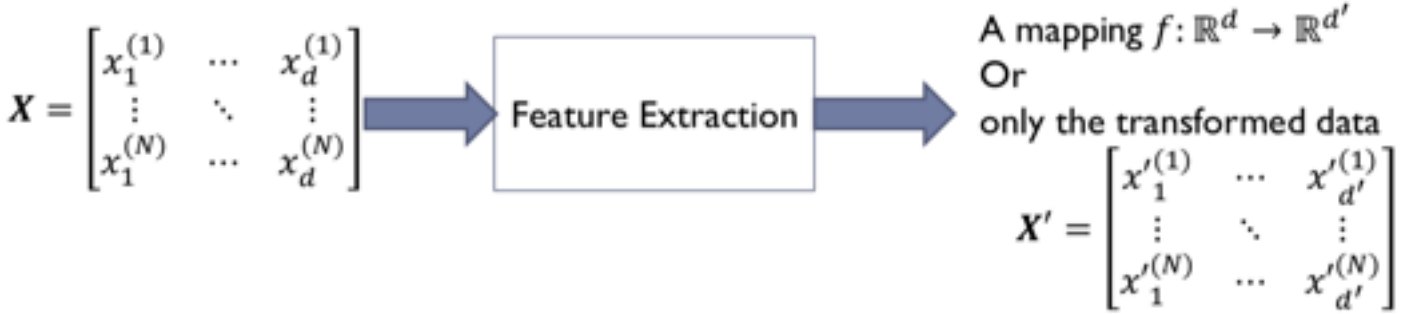
Feature Selection $(d' < d)$

$$\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \rightarrow \begin{bmatrix} y_1 \\ \vdots \\ y_{d'} \end{bmatrix} = f\left(\begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix}\right)$$

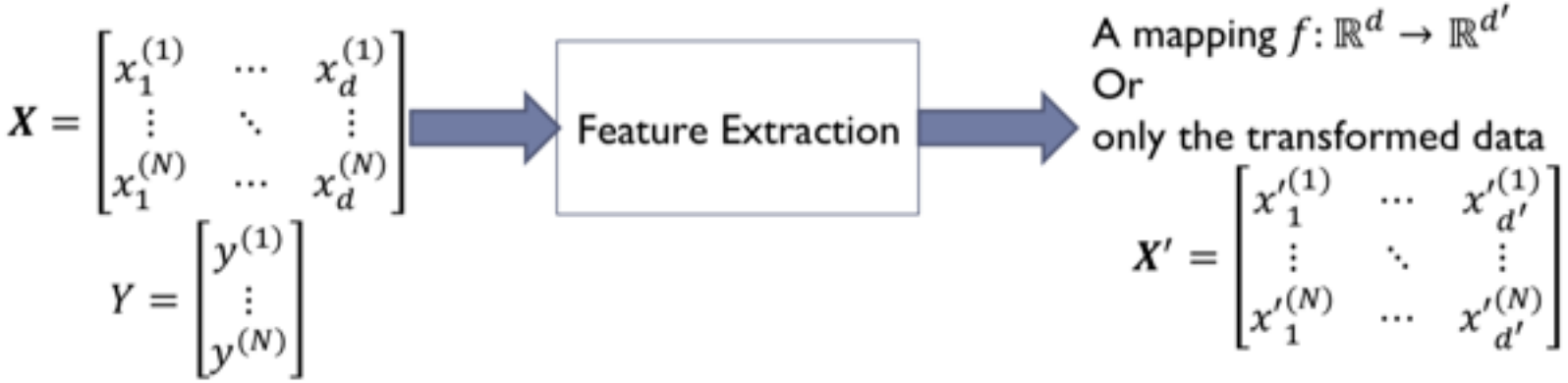
Feature Extraction

Feature Selection vs. Extraction

- Unsupervised Feature Extraction

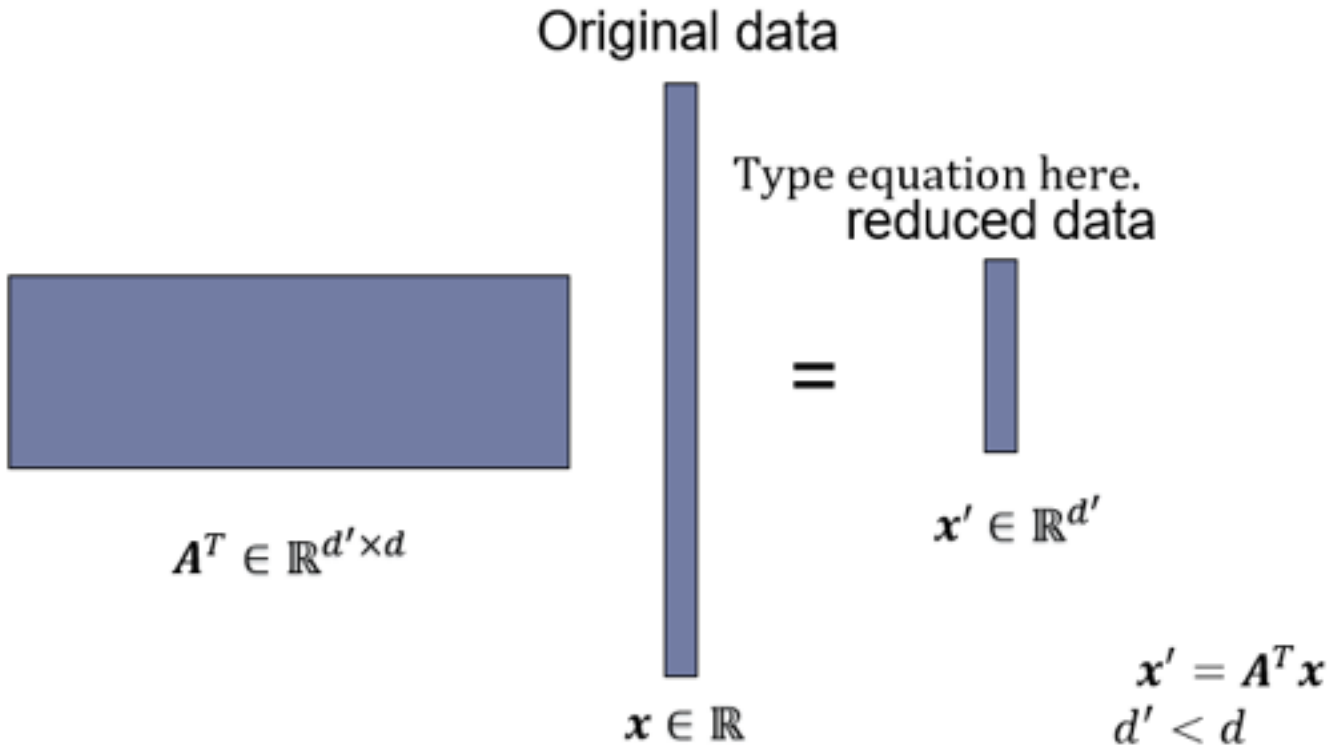


- Supervised Feature Extraction



Linear Dimensionality Reduction

- Finding a linear transformation (matrix) that directly maps data from high dimensional feature space to a low dimensional feature space



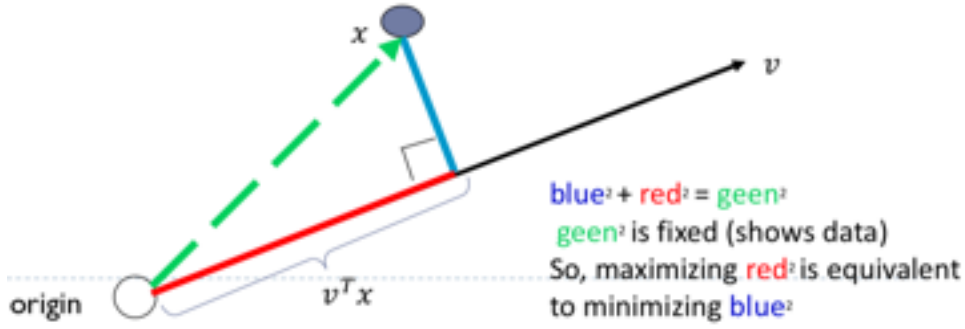
Principal Component Analysis (PCA)

- Goal:
  - Reducing the dimensionality of data by a **linear transformation** while preserving important aspects of the data
- Two equivalent views: find the direction for which
  - the variation presents in the dataset is as much as possible

$$\underset{v}{\operatorname{argmax}} \frac{1}{N} \sum_{n=1}^{N} \left(v^T x^{(n)}\right)^2$$

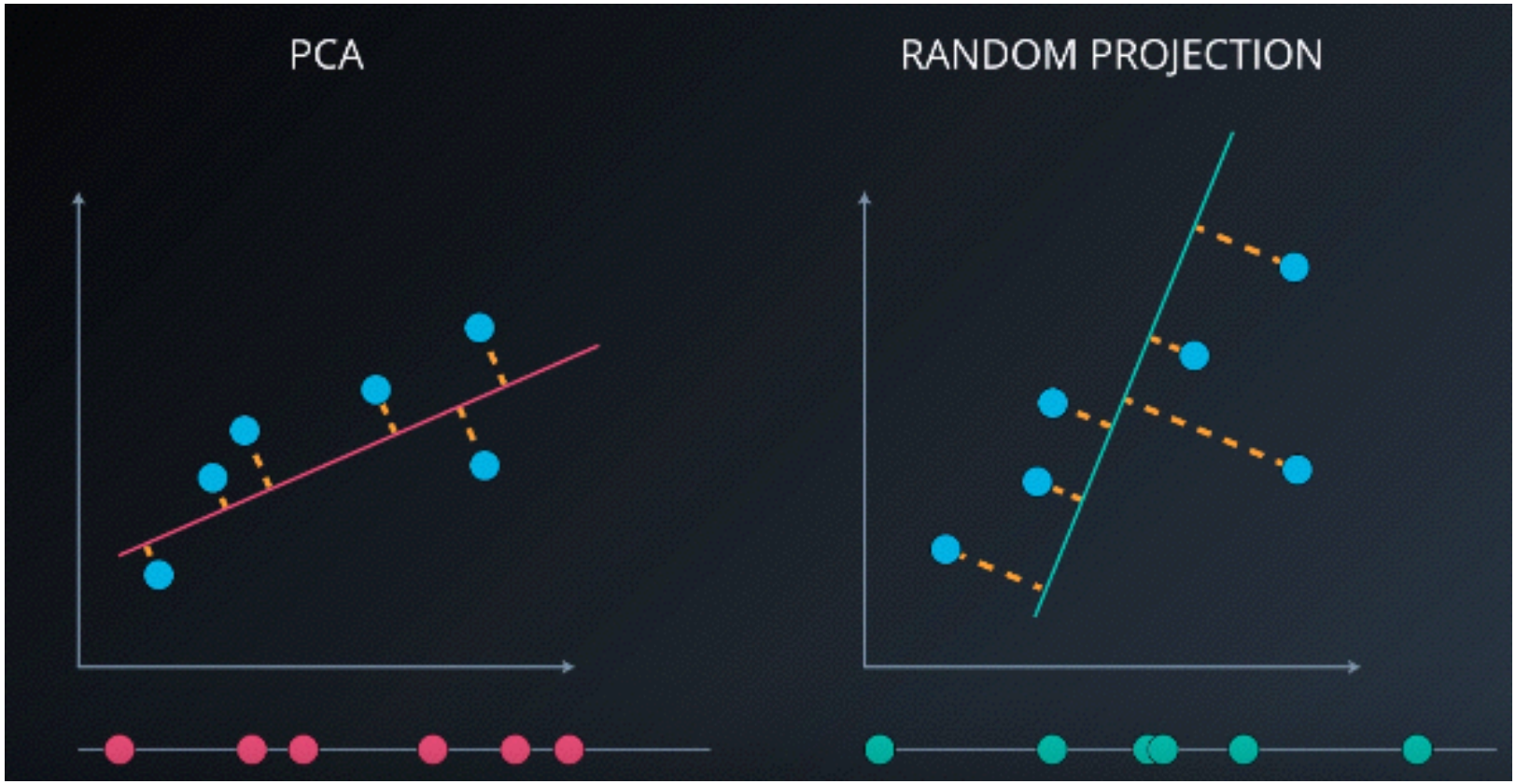
  - the reconstruction error is minimized

$$\underset{v}{\operatorname{argmin}} \sum_{n=1}^{N} \left\| x^{(n)} - \left(v^T x^{(n)}\right) v \right\|^2$$



PCA vs. Random Projection

PCA finds directions that preserve the information in data and original data can be reconstructed by minimum error.



Principal Component Analysis (PCA)

- Goal:
  - Reducing the dimensionality of data by a linear transformation while preserving important aspects of the data
- Two equivalent views: find the direction for which
  - the variation presents in the dataset is as much as possible
  - the reconstruction error is minimized
- Method: Mapping each data onto first few ***Principal components***

Principal Components

- **Principal Components:** ***Orthonormal basis*** for the vector space of data that are ordered by the fraction of the total ***information (variation)*** in the corresponding directions
- **Claim: PCs** are the eigenvectors of the **covariance Matrix** of the data points.

$$\boldsymbol{\mu_x} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_d \end{bmatrix} = \begin{bmatrix} E(x_1) \\ \vdots \\ E(x_d) \end{bmatrix}$$

$$\boldsymbol{\Sigma} = E[(\boldsymbol{x} - \boldsymbol{\mu_x})(\boldsymbol{x} - \boldsymbol{\mu_x})^T]$$

- ML estimate of covariance matrix from $\{\boldsymbol{x}^{(i)}\}_{i=1}^{N}$ data points :

$$\boldsymbol{S} = \frac{1}{N}\sum_{i=1}^{N}(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}})(\boldsymbol{x}^{(i)} - \bar{\boldsymbol{x}})^T = \frac{1}{N}(\tilde{\boldsymbol{X}}^T\tilde{\boldsymbol{X}})$$

$$\tilde{\boldsymbol{X}} = \begin{bmatrix} \tilde{\boldsymbol{x}}^{(1)} \\ \vdots \\ \tilde{\boldsymbol{x}}^{(N)} \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^{(1)} - \bar{\boldsymbol{x}} \\ \vdots \\ \boldsymbol{x}^{(N)} - \bar{\boldsymbol{x}} \end{bmatrix} \qquad \bar{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N}\boldsymbol{x}^{(i)}$$

First Principal Component

Find vector $\boldsymbol{v1}$ that maximizes sample variance

$$\max_{v_1} \frac{1}{N} \sum_{n=1}^{N} \left(v_1^T x^{(n)} - v_1^T \bar{x}\right)^2$$

$$= \frac{1}{N} \sum_{n=1}^{N} v_1^T \left(x^{(n)} - \bar{x}\right)\left(x^{(n)} - \bar{x}\right)^T v_1$$

$$= v_1^T \left(\frac{1}{N} \sum_{n=1}^{N} \left(x^{(n)} - \bar{x}\right)\left(x^{(n)} - \bar{x}\right)^T \right) v_1 = v_1^T S v_1$$

$$\text{s.t. } v_1^T v_1 = 1$$

First Principal Component

$$\max_{v} \frac{1}{N} \sum_{n=1}^{N} \left(v_1^T x^{(n)} - v_1^T \bar{x} \right)^2 = v_1^T S v_1$$

$$\text{s.t. } v_1^T v_1 = 1$$

$$L(v_1, \lambda_1) = v_1^T S v_1 + \lambda_1 (1 - v_1^T v_1)$$

$$\frac{\partial L}{\partial v_1} = 0 \Rightarrow 2Sv_1 - 2\lambda_1 v_1 = 0$$

$$\Rightarrow Sv_1 = \lambda_1 v_1$$

Eigenvector with maximum eigenvalue maximizes the objective

Second Principal Component

$$\max_{v_2} v_2^T S v_2$$
$$\text{s.t. } v_2^T v_2 = 1$$
$$v_2^T v_1 = 0$$

$$L(v_2, \lambda_2, \alpha) = v_2^T S v_2 + \lambda_2(1 - v_2^T v_2) - \alpha v_2^T v_1$$

Finding α:

$$\frac{\partial L}{\partial v_2} = 0 \Rightarrow 2Sv_2 - 2\lambda_2 v_2 - \alpha v_1 = 0$$
$$\Rightarrow 2v_1^T S v_2 - 2\lambda_2 v_1^T v_2 - \alpha v_1^T v_1 = 0$$
$$\Rightarrow 2\lambda_1 v_1^T v_2 - 2\lambda_2 \times 0 - \alpha = 0$$
$$\Rightarrow \alpha = 0$$

Second Principal Component

$$\max_{v_2} v_2^T S v_2$$
$$\text{s.t. } v_2^T v_2 = 1$$
$$v_2^T v_1 = 0$$

$$L(v_2, \lambda_2, \alpha) = v_2^T S v_2 + \lambda_2(1 - v_2^T v_2) - \alpha \overbrace{v_2^T v_1}^{0}$$

Finding λ_2:

$$\frac{\partial L}{\partial v_2} = 0 \Rightarrow 2Sv_2 - 2\lambda_2 v_2 = 0$$
$$\Rightarrow Sv_2 = \lambda_2 v_2$$

v**2** is the second largest eigenvalue

PCA Steps

- Input: $N \times d$ data matrix $\boldsymbol{X}$ (each row contain a d dimensional data point)
  - $\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{i=1}^{N} \boldsymbol{x}^{(i)}$
  - $\widetilde{\boldsymbol{X}} \leftarrow$ Mean value of data points is subtracted from rows of $\boldsymbol{X}$
  - $\boldsymbol{S} = \frac{1}{N}\widetilde{\boldsymbol{X}}^T\widetilde{\boldsymbol{X}}$ (Covariance matrix)
  - Calculate eigenvalue and eigenvectors of $\boldsymbol{S}$
  - Pick d' eigenvectors corresponding to the largest eigenvalues and put them in the columns of $\boldsymbol{A} = [\boldsymbol{v}_1, \dots, \boldsymbol{v}_{d'}]$
  - $\boldsymbol{X}' = \boldsymbol{X}\boldsymbol{A}$

First PC d'-th PC

Reconstruction

- Original data can be reconstructed using the PCs and the transformed data

$$x' = \begin{bmatrix} v_1^T x \\ \vdots \\ v_{d'}^T x \end{bmatrix}$$

$$A = [v_1, \dots, v_{d'}]$$

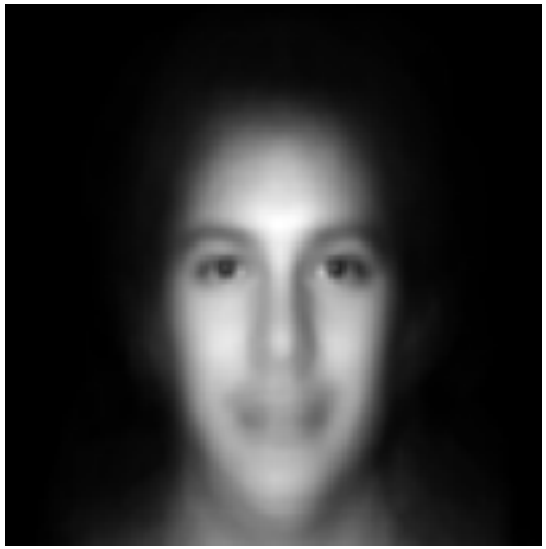
$$x' = A^T(x - \bar{x})$$

$$\Rightarrow \hat{x} = \bar{x} + Ax' = \bar{x} + AA^T(x - \bar{x})$$

if we use all PCs, then original data can be found without any error.

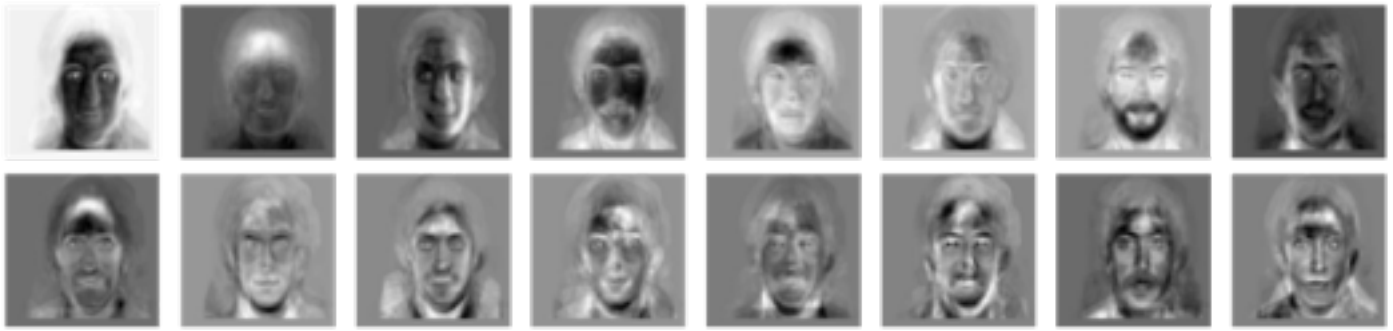
PCA on faces: Eigenfaces

Display regular faces of dataset

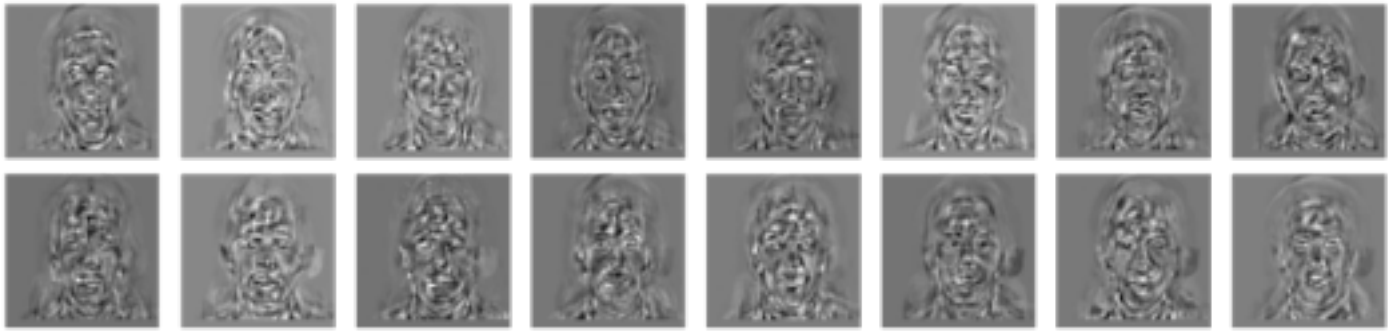


Mean face

Eigenfaces: PCs of faces



First few eigen faces **vs.** Last few eigenfaces



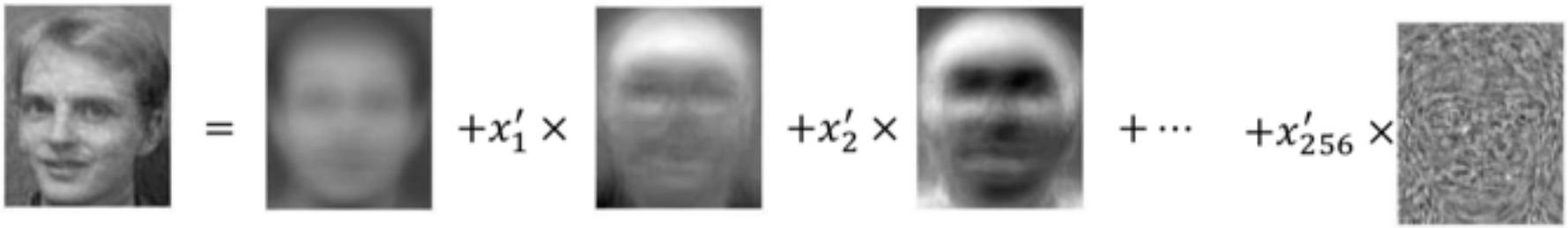
Eigenfaces: Reconstructing images

$\boldsymbol{x}$ is a $112 \times 92 = 10304$ dimensional vector containing intensity of the pixels of this image and $\tilde{\boldsymbol{x}} = \boldsymbol{x} - \bar{\boldsymbol{x}}$

Feature vector= $[x'_1, x'_2, \ldots, x'_{d'}]$

$x'_i = \boldsymbol{v}_i^T \tilde{\boldsymbol{x}}$ ⟶ The projection of $\boldsymbol{x}$ on the i-th PC

$$\hat{\boldsymbol{x}} = \bar{\boldsymbol{x}} + \sum_{i=1}^{d'} x'_i \times \boldsymbol{v}_i \qquad \hat{\boldsymbol{x}} = \bar{\boldsymbol{x}} + \sum_{i=1}^{d'} (\boldsymbol{v}_i^T \tilde{\boldsymbol{x}}) \times \boldsymbol{v}_i$$

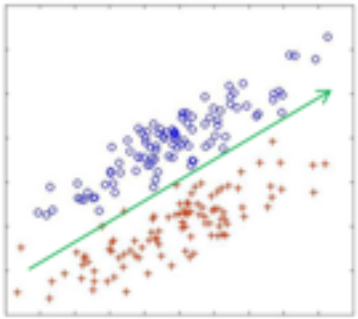


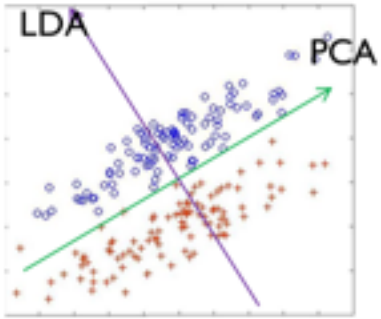
Average

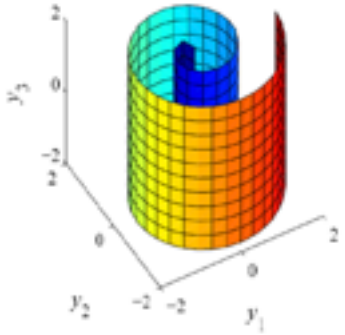
Eigenfaces: Reconstructing images

Pros & Cons

- No parameter tuning
- PCA is deterministic and fast
- Can be a preprocessing step specially for noise reduction in data
- However:
  - Data of different classes may not be separable after PCA
  - **Distance among data** and data **topology** may not be preserved



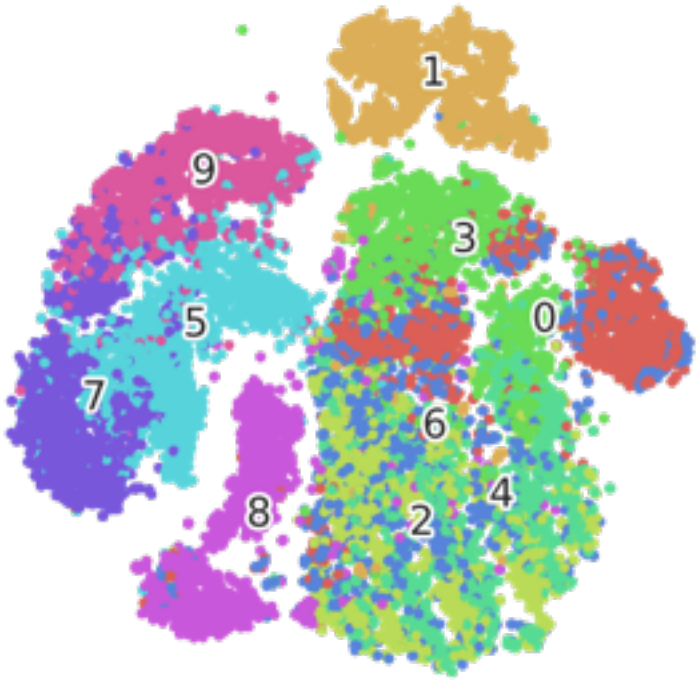




t-SNE

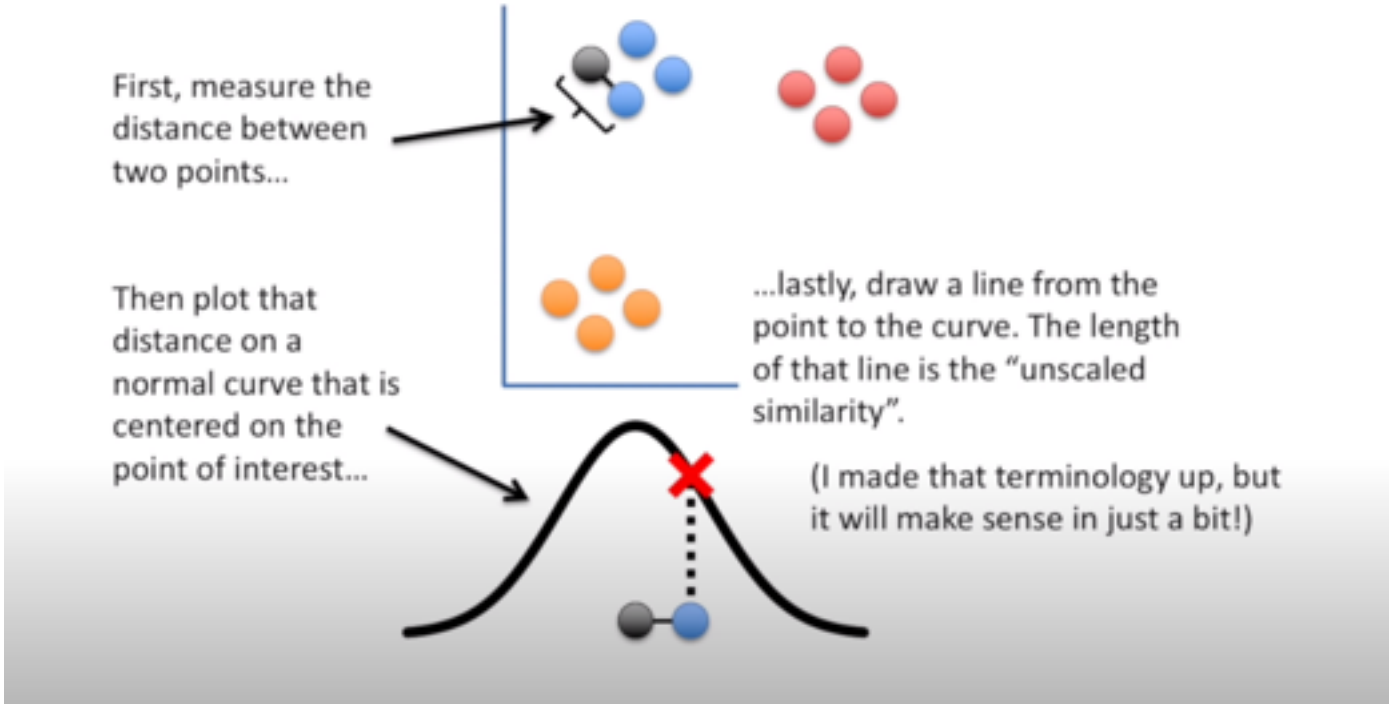
t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) is an **unsupervised, non-linear** technique primarily used for data exploration and **visualizing** high-dimensional data. The technique is a variation of Stochastic Neighbor Embedding (SNE).



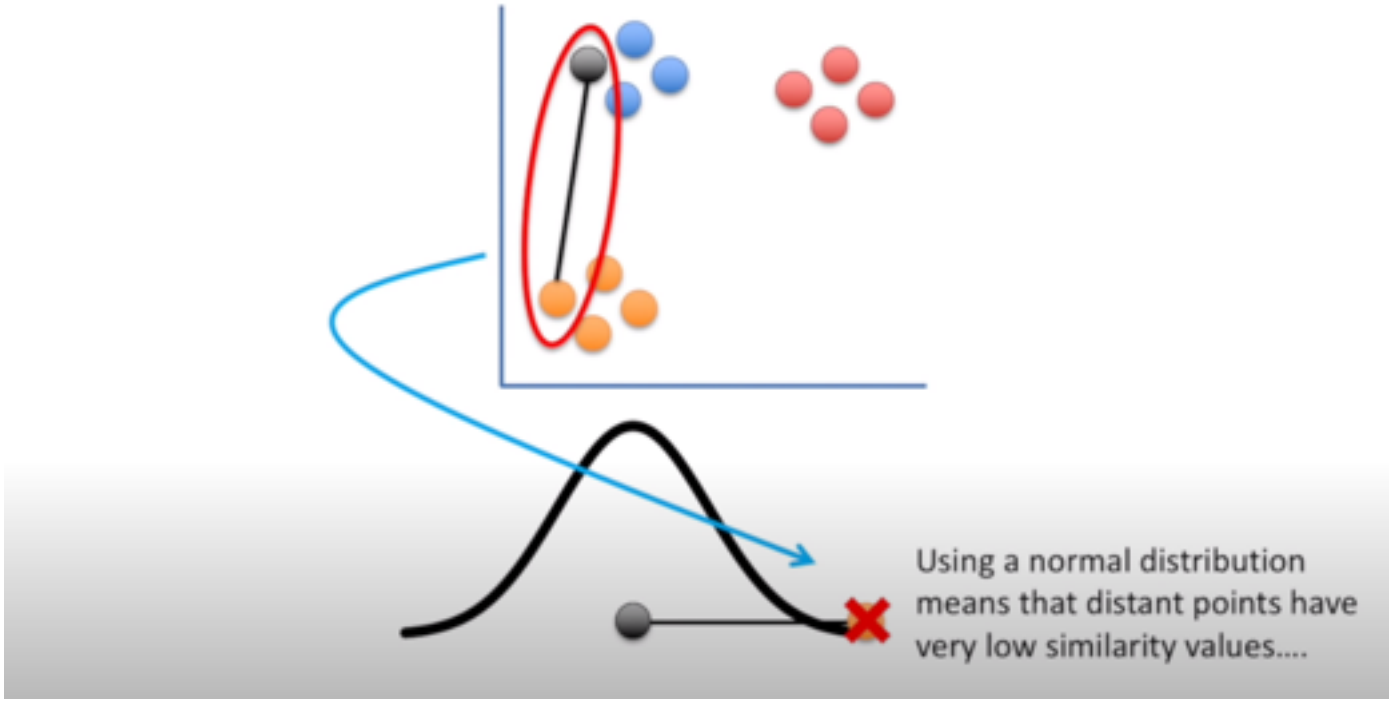
Intuition

overview of working of t-SNE:
The algorithms starts by calculating the probability of similarity of points in high-dimensional space and calculating the probability of similarity of points in the corresponding low-dimensional space.



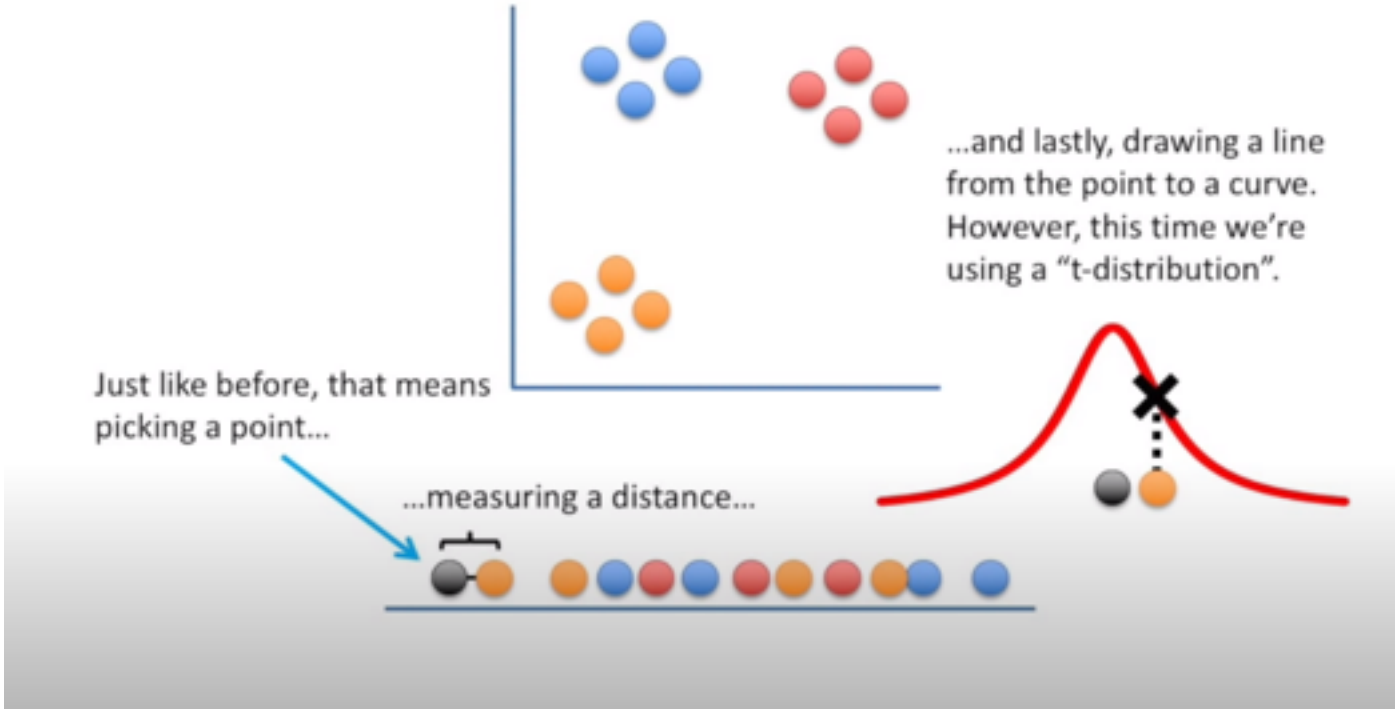
Intuition

The similarity of points is calculated as the conditional probability that a point A would choose point B as its neighbor if neighbors were picked in proportion to their probability density under a Gaussian (normal distribution) centered at A.

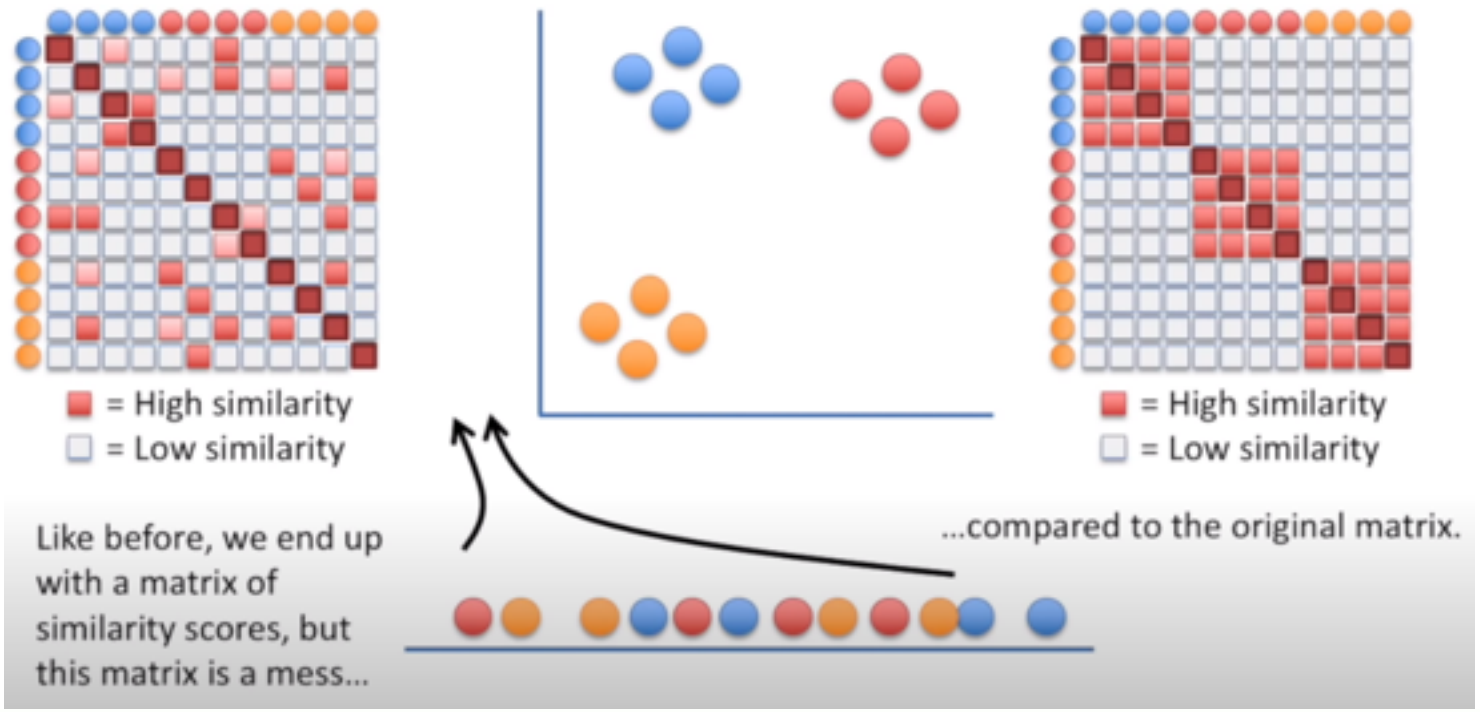


Intuition

It then tries to minimize the difference between these conditional probabilities (or similarities) in higher-dimensional and lower-dimensional space for a perfect representation of data points in lower-dimensional space.



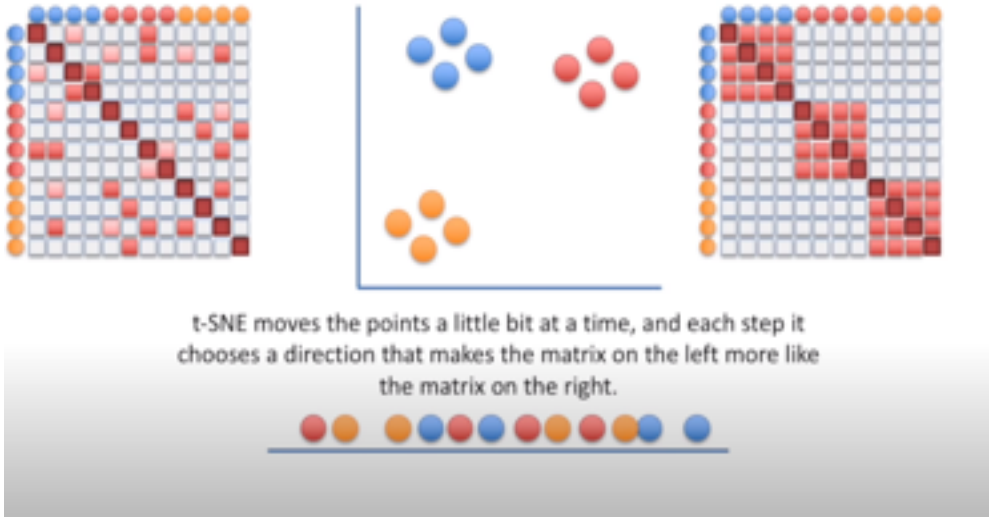
Intuition


= High similarity
= Low similarity
= High similarity
= Low similarity
Like before, we end up with a matrix of similarity scores, but this matrix is a mess...
...compared to the original matrix.

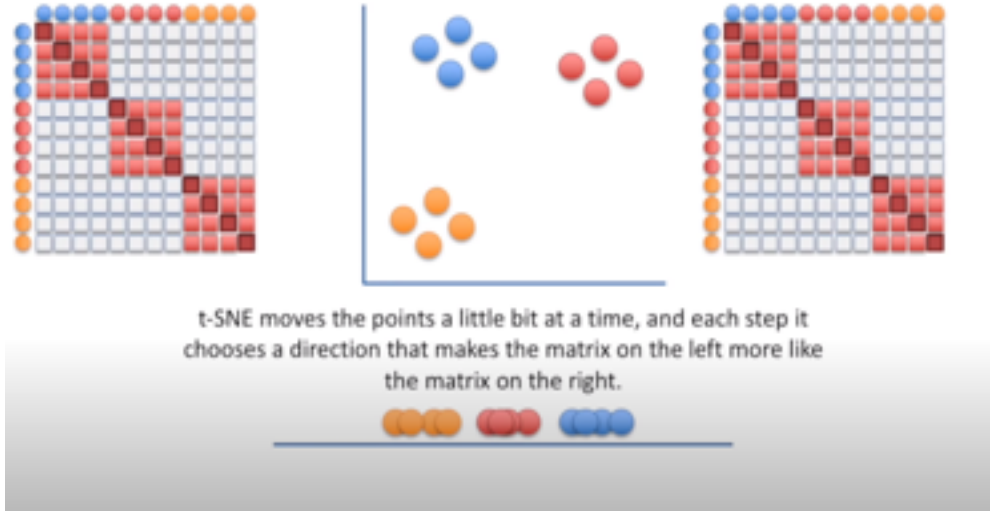

1
0.5
0
−0.5
−1
0
2
4
6
8
10

Intuition

In simpler terms, t-SNE gives you a feel or intuition of how the data is arranged in a high-dimensional space. It was developed by Laurens van der Maatens and Geoffrey Hinton in 2008.



Geoffrey Hinton



Laurens van der Maaten

SNE pair-wise similarities

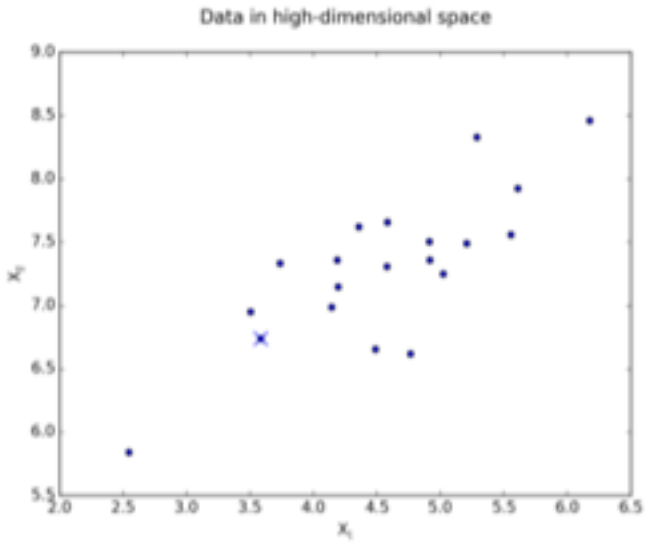
dimensionality reduction methods convert the high-dimensional data set X = $\{x_1, \ldots, x_n\}$ into two or three-dimensional data Y = $\{y_1, \ldots, y_n\}$ that can be displayed in a scatterplot. Stochastic Neighbor Embedding (SNE) starts by converting the high-dimensional Euclidean distances between datapoints into conditional probabilities that represent similarities.

$$p_{j|i} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)} \qquad p_{j|j} = 0$$

For nearby datapoints, $P_{j|i}$ is relatively high, whereas for widely separated datapoints, $P_{j|i}$ will be almost infinitesimal (for reasonable values of the variance of the Gaussian, σ). We set the variance of the Gaussian that is employed in the computation of the conditional probabilities $P_{j|i} = \frac{1}{\sqrt{2}}$.

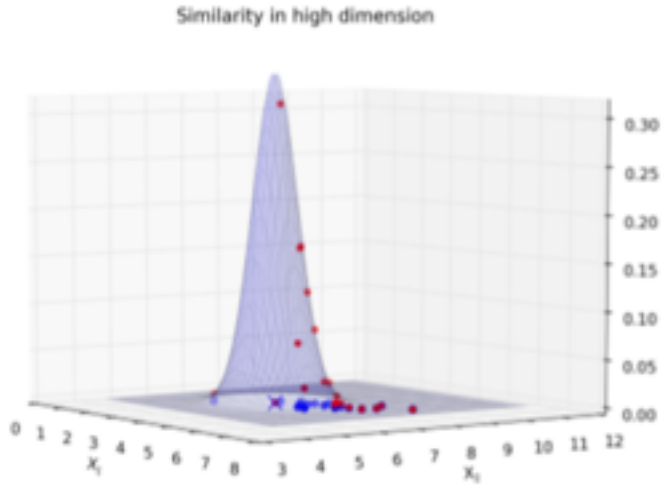
$$q_{j|i} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq i} \exp\left(-\|y_i - y_k\|^2\right)} \qquad q_{j|j} = 0$$

pair-wise similarities stay the same

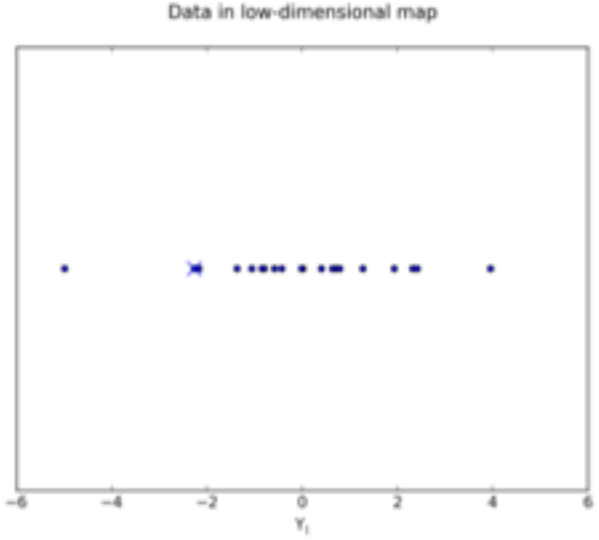


$p_{j|i}$

$\Leftrightarrow$

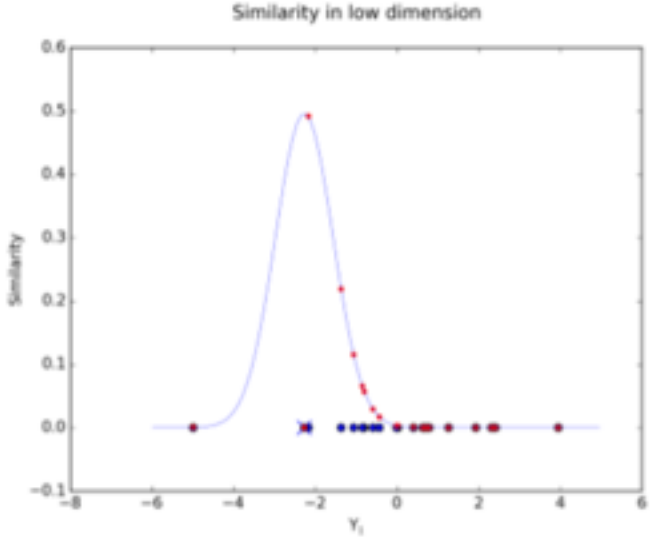


$\Downarrow$

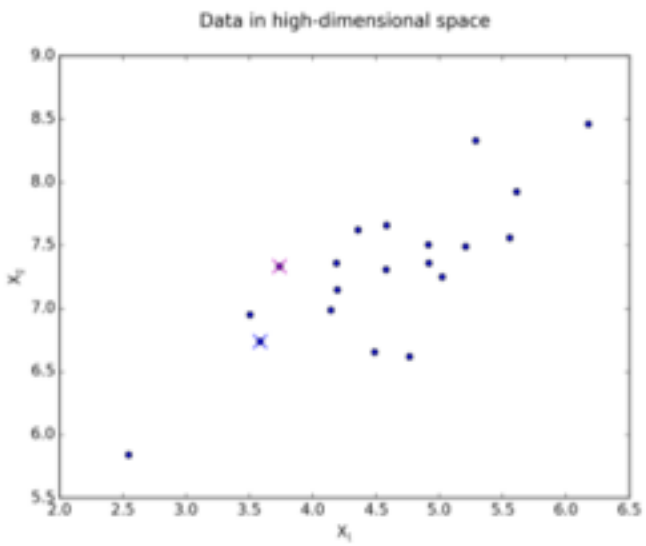


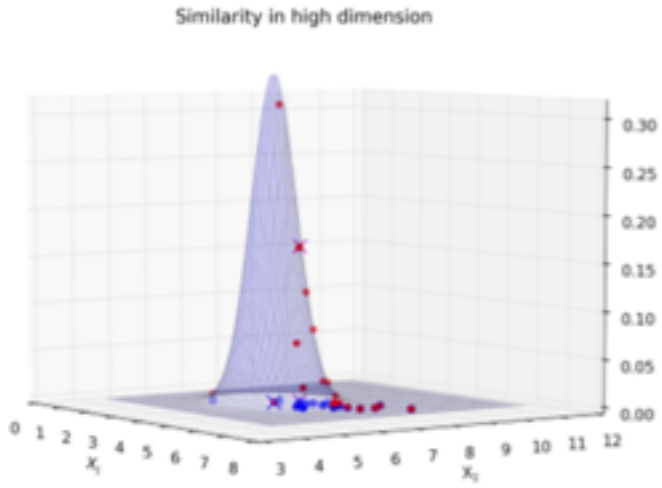
$q_{j|i}$

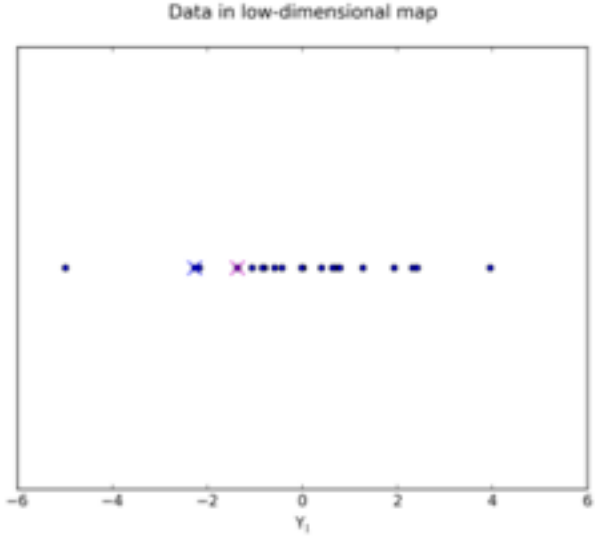
$\Leftrightarrow$

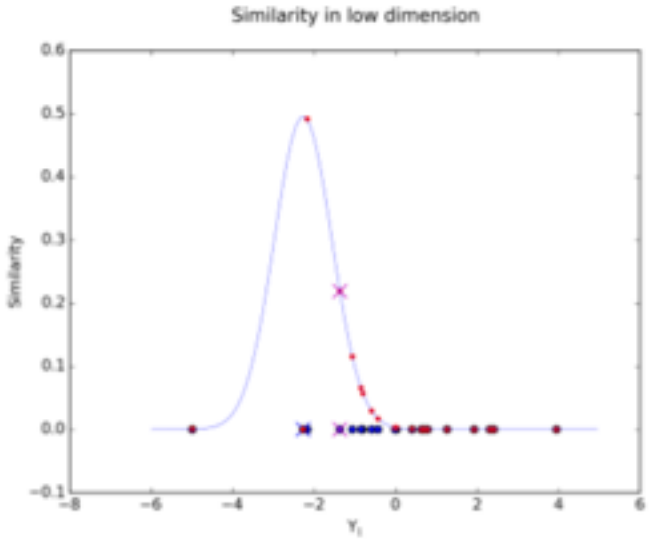


pair-wise similarities stay the same

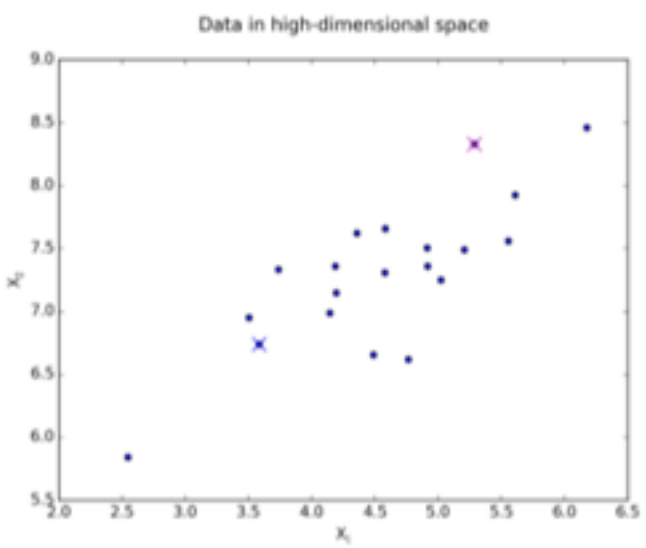


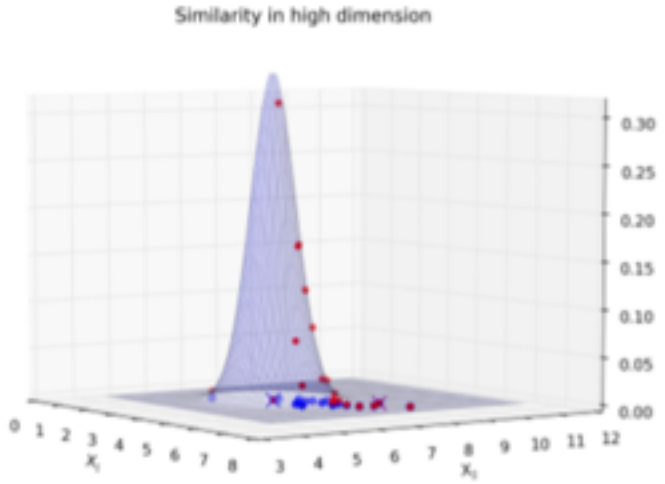


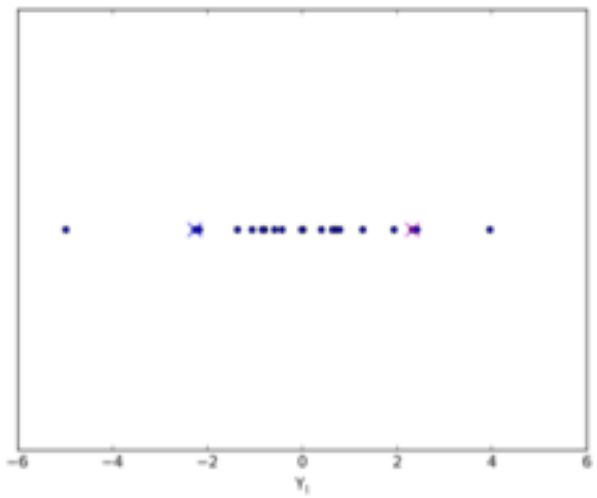


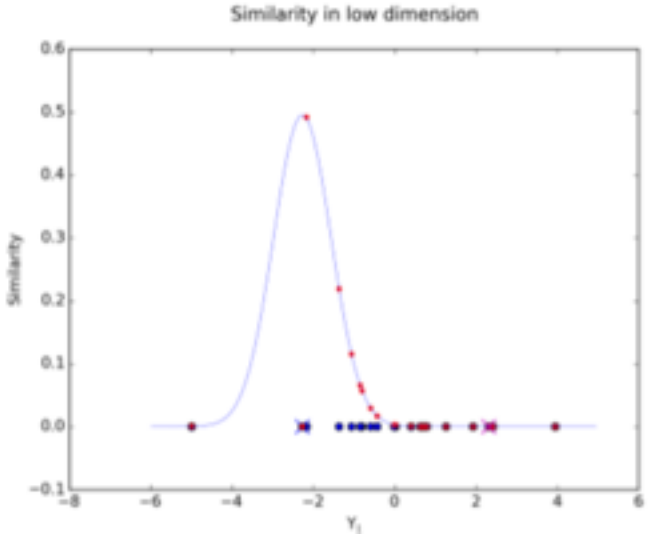


pair-wise similarities stay the same

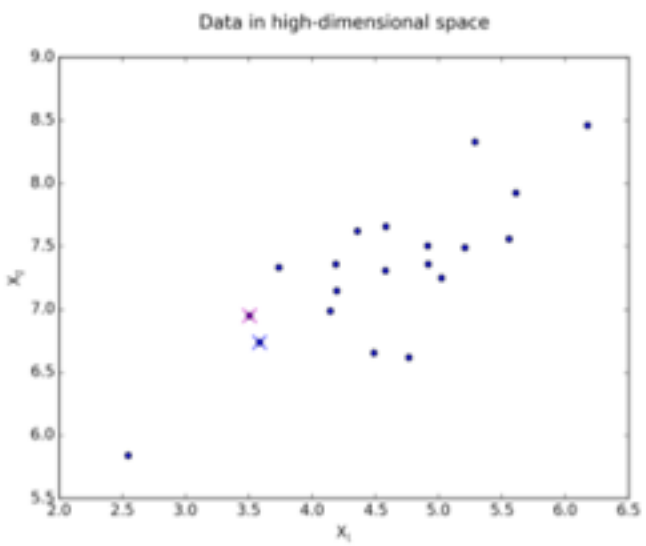


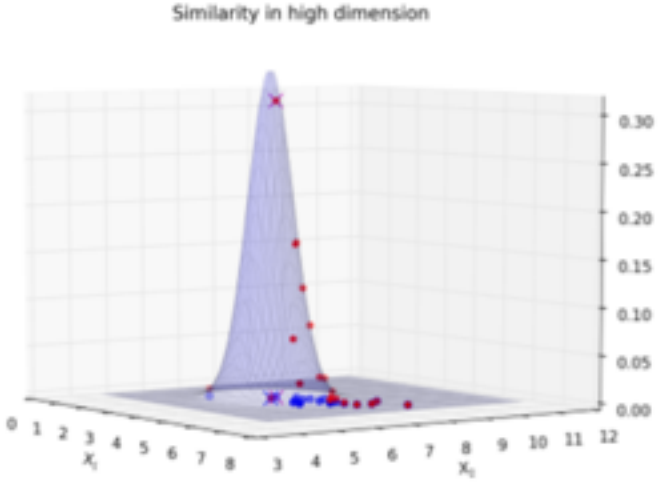


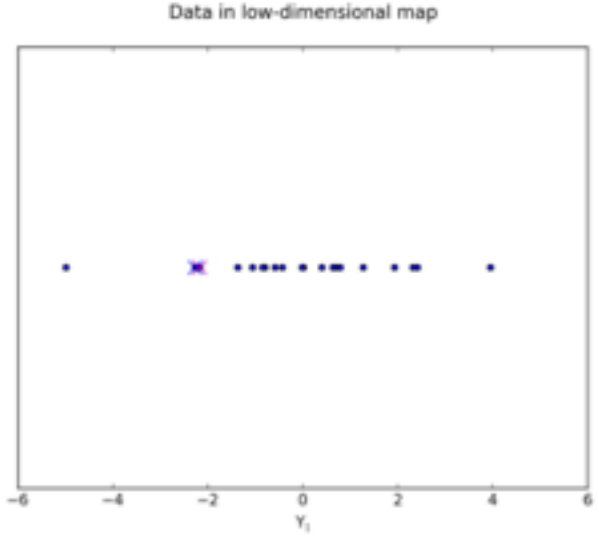


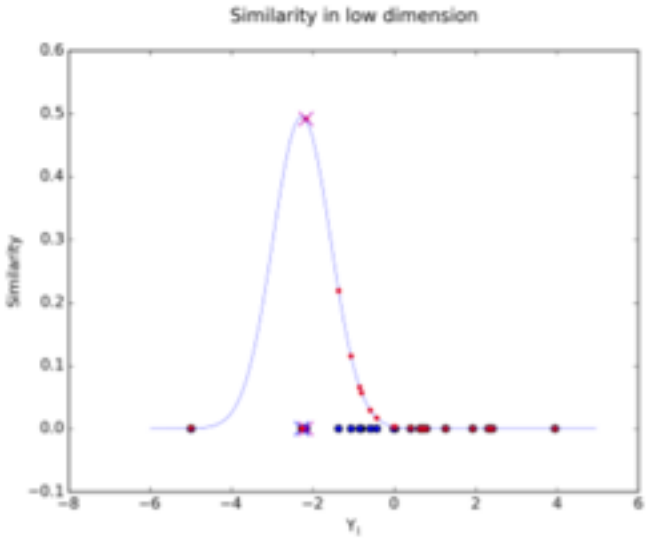


pair-wise similarities stay the same









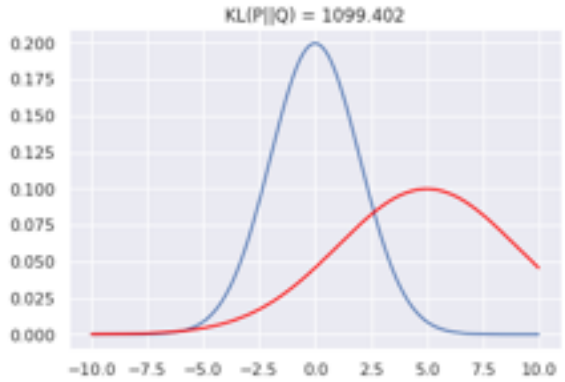
SNE

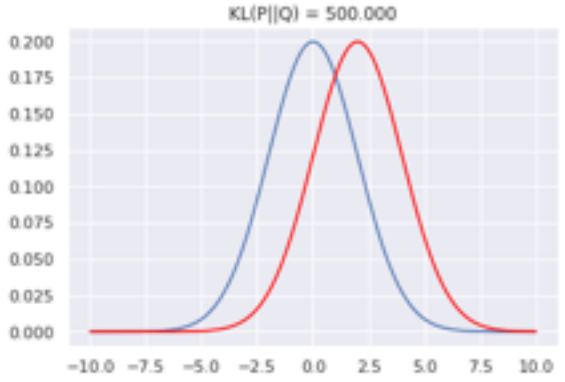
If the map points y_i and y_j correctly model the similarity between the high-dimensional datapoints x_i and x_j, the conditional probabilities $p_{j|i}$ and $q_{j|i}$ will be equal.

A natural measure of the faithfulness with which $q_{j|i}$ models $p_{j|i}$ is the **KullbackLeibler divergence** (which is in this case equal to the cross-entropy up to an additive constant). The Kullback-Leibler divergence, D_{KL}, is a measure of how one probability distribution is different from a second, reference probability distribution.

$$D_{\mathrm{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log\left(\frac{P(x)}{Q(x)}\right)$$

Lower the KL divergence value, the better we have matched the true distribution with our approximation.





SNE

SNE minimizes the sum of Kullback-Leibler divergences over all datapoints using a gradient descent method.

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

P_i represents the conditional probability distribution over all other datapoints given datapoint x_i , and Q_i represents the conditional probability distribution over all other map points given map point y_i.

It is not likely that there is a single value of σ_i that is optimal for all datapoints in the data set because the density of the data is likely to vary. 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.

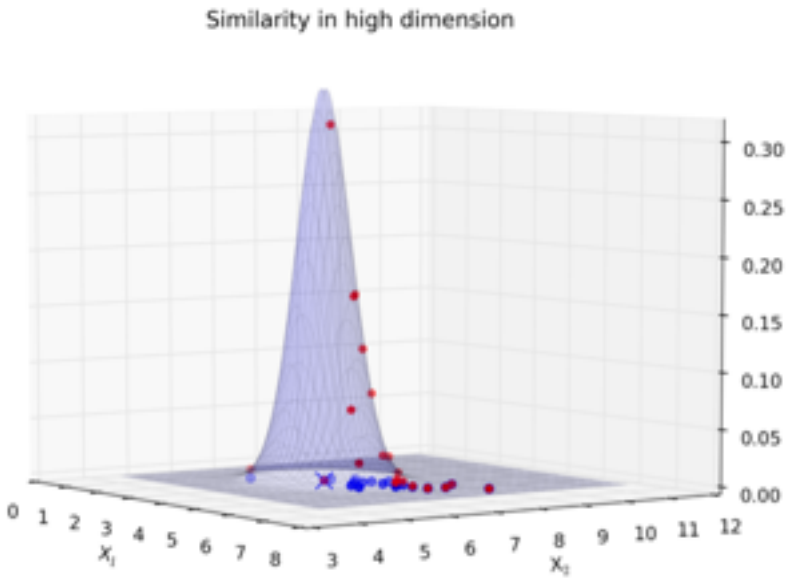
$$H(P_i) = -\sum_j p_{j|i} \log_2 p_{j|i} \qquad Perp(P_i) = 2^{H(P_i)}$$

Some questions

- **Why radial basis function (exponential)?**

  Focus on **local geometry**.

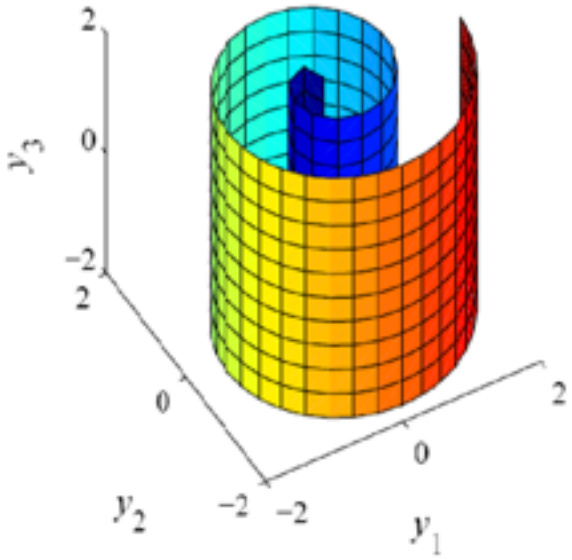
  This is why t-SNE can be interpreted as **topology-based.**



- **Why probabilities?**

  Small distance does not mean proximity on manifold.

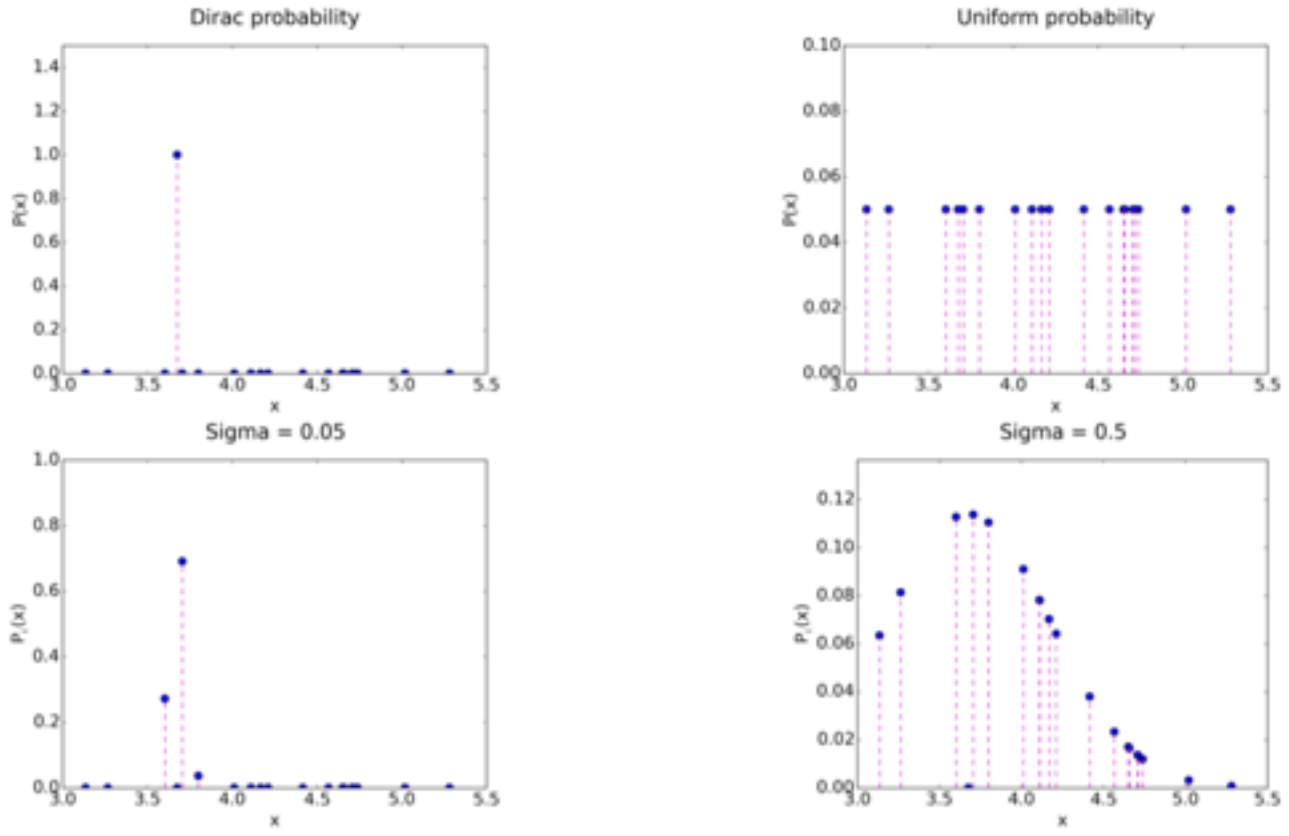
  Probabilities are appropriate to model this **uncertainty**.



Some questions

- **How do you choose σi?**

The entropy of Pi increases with σi.



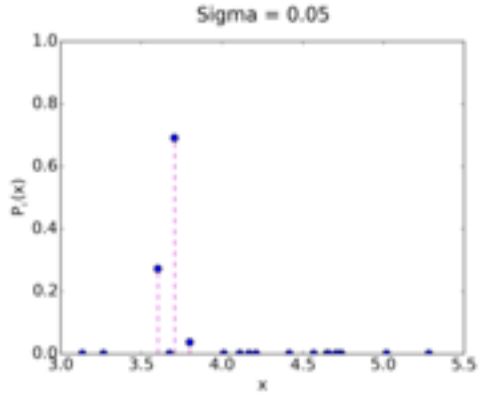
Some questions

- **How do you choose σi?**

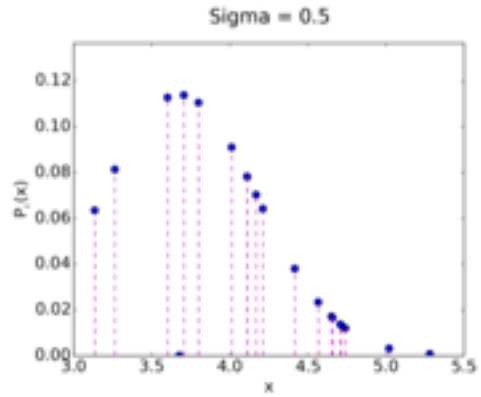
Perplexity, a smooth measure of the number of neighbors.

$$H(P) = -\sum \text{Pi} * \log_2 Pi$$

$$Prep(p) = 2^{H(p)}$$



⇒ Entropy of 1.055
Perplexity of 2.078



⇒ Entropy of 3.800
Perplexity of 13.929

SNE

The minimization of the cost function in Equation 2 is performed using a gradient descent method.

$$\frac{\delta C}{\delta y_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

The current gradient is added to an exponentially decaying sum of previous gradients in order to determine the changes in the coordinates of the map points at each iteration of the gradient search.

$$\mathcal{Y}^{(t)} = \mathcal{Y}^{(t-1)} + \eta \frac{\delta C}{\delta \mathcal{Y}} + \alpha(t)\left(\mathcal{Y}^{(t-1)} - \mathcal{Y}^{(t-2)}\right)$$

t-SNE

In this section, we first discuss the symmetric version of SNE

As an alternative to minimizing the sum of the Kullback-Leibler divergences between the conditional probabilities $P_{j|i}$ and $Q_{j|i}$, it is also possible to minimize a single Kullback-Leibler divergence between a joint probability distribution, P, in the high-dimensional space and a joint probability distribution, Q, in the low-dimensional space

$$C = KL(P||Q) = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

We refer to this type of SNE as symmetric SNE, because it has the property that $P_{ij} = P_{ji}$ and $q_{ij} = q_{ji}$ for ∀ i, j.

$$q_{ij} = \frac{\exp\left(-\|y_i - y_j\|^2\right)}{\sum_{k \neq l} \exp\left(-\|y_k - y_l\|^2\right)} \qquad p_{ij} = \frac{\exp\left(-\|x_i - x_j\|^2 / 2\sigma^2\right)}{\sum_{k \neq l} \exp\left(-\|x_k - x_l\|^2 / 2\sigma^2\right)}$$

t-SNE

But this causes problems when a high-dimensional datapoint xi is an outlier.

We set $P_{ij} = \frac{P_{i|j} + P_{j|i}}{2n}$. This ensures that $\sum_j P_{ij} > \frac{1}{2n}$ for all datapoints xi, as a result of which each datapoint xi makes a significant contribution to the cost function.

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

The main advantage of the symmetric version of SNE is the simpler form of its gradient, which is faster to compute.

SNE

In t-SNE, we employ a Student t-distribution with one degree of freedom as the heavy-tailed distribution in the low-dimensional map.

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

The gradient of the Kullback-Leibler divergence between P and the Student-t based joint probability distribution Q:

$$\frac{\delta C}{\delta y_i} = 4 \sum_j (p_{ij} - q_{ij})(1 + \|y_i - y_j\|^2)^{-1}(y_i - y_j)$$

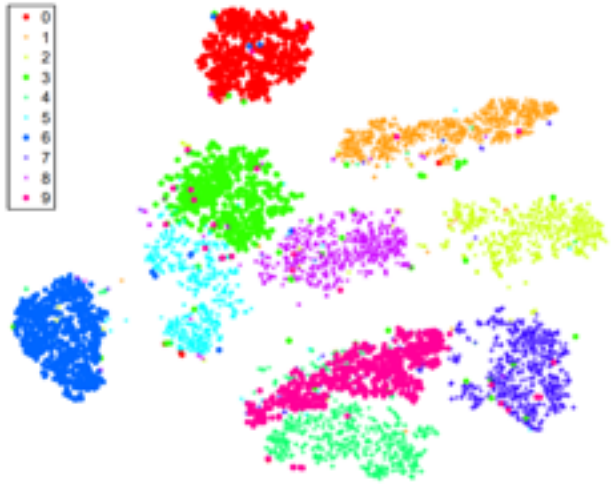
- *What are the differences between SNE and t-SNE?*
- *Why we should use t-SNE instead of SNE?*

Pseudo Code

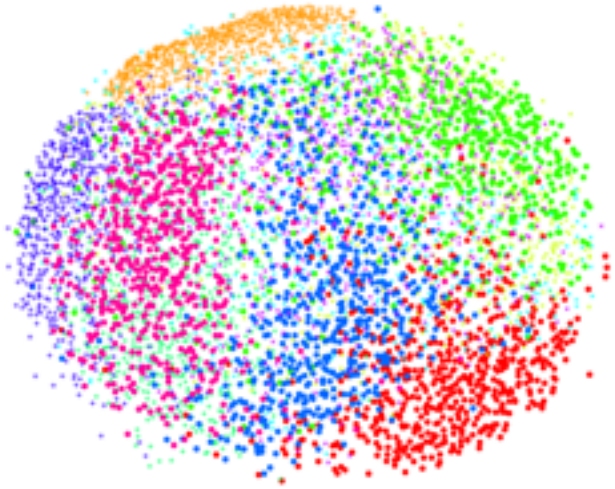
Algorithm 1: Simple version of t-Distributed Stochastic Neighbor Embedding.

Data: data set $\mathcal{X} = \{x_1, x_2, ..., x_n\}$,
cost function parameters: perplexity *Perp*,
optimization parameters: number of iterations T, learning rate η, 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

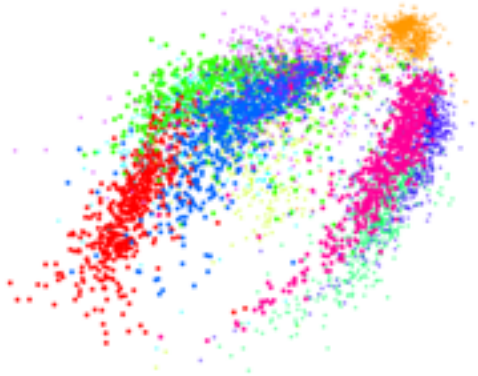
Visualizations of 6,000 handwritten digits from the MNIST data set.



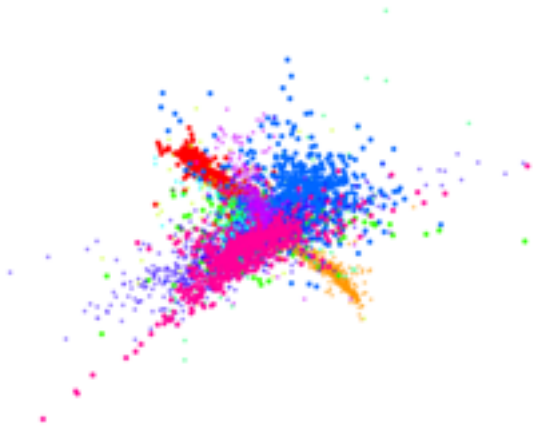
(a) Visualization by t-SNE.



(b) Visualization by Sammon mapping.



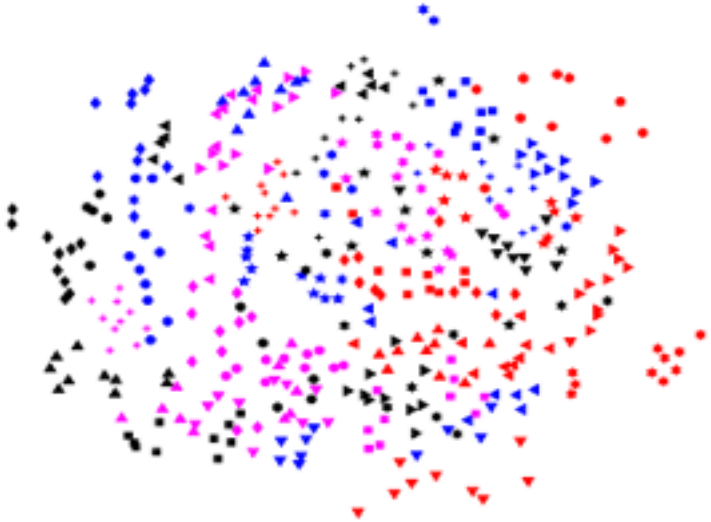
(a) Visualization by Isomap.



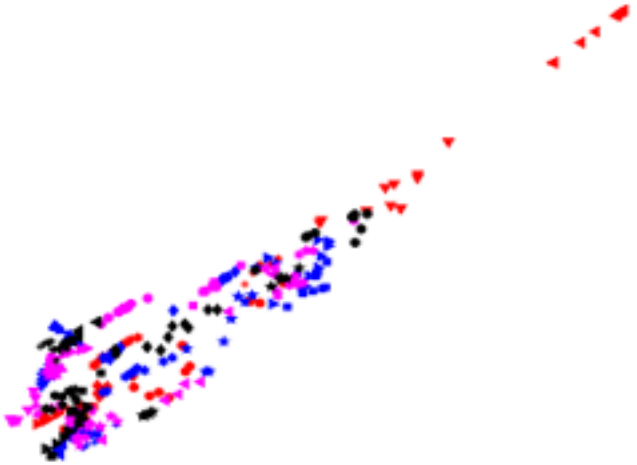
(b) Visualization by LLE.

Visualizations of the Olivetti faces data set.

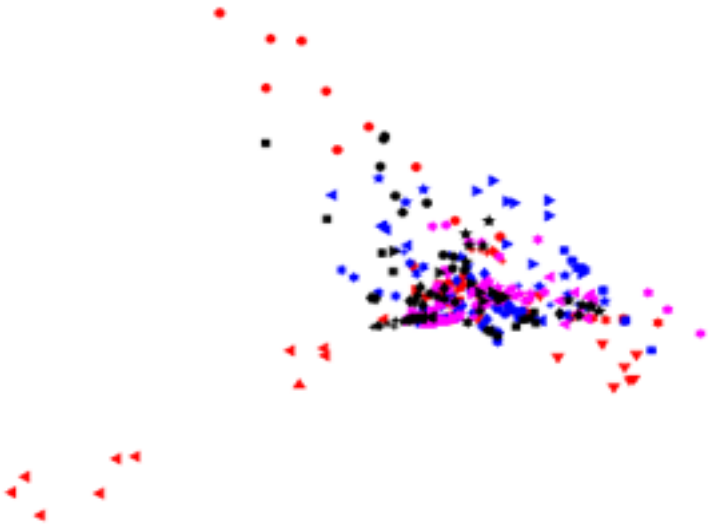
(a) Visualization by t-SNE.



(b) Visualization by Sammon mapping.



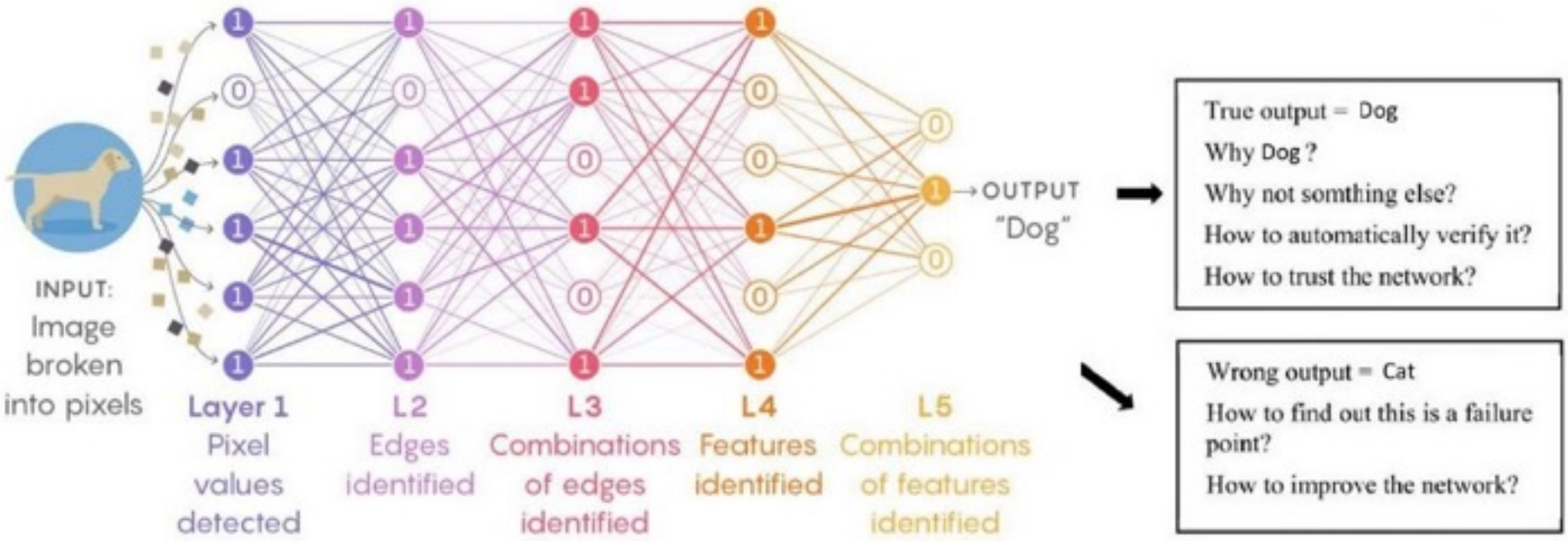
(c) Visualization by Isomap.



(d) Visualization by LLE.

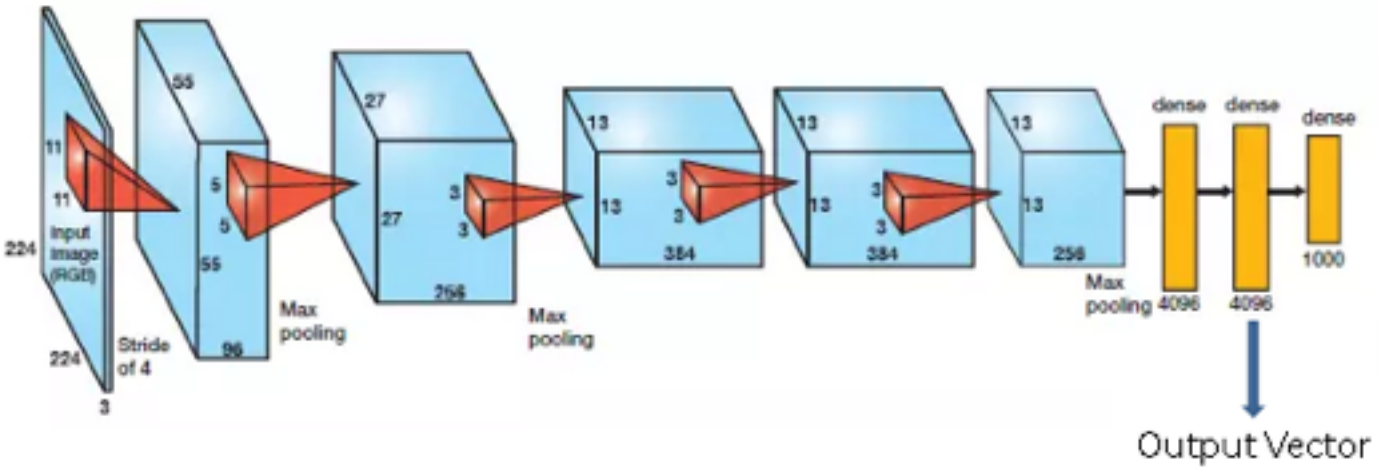
Important Application

The Deep Learning Algorithms mimic the cognitive capabilities of the human brain and the Interpretability of Deep Learning Algorithms has been an important area of research since the start of the Deep Learning era



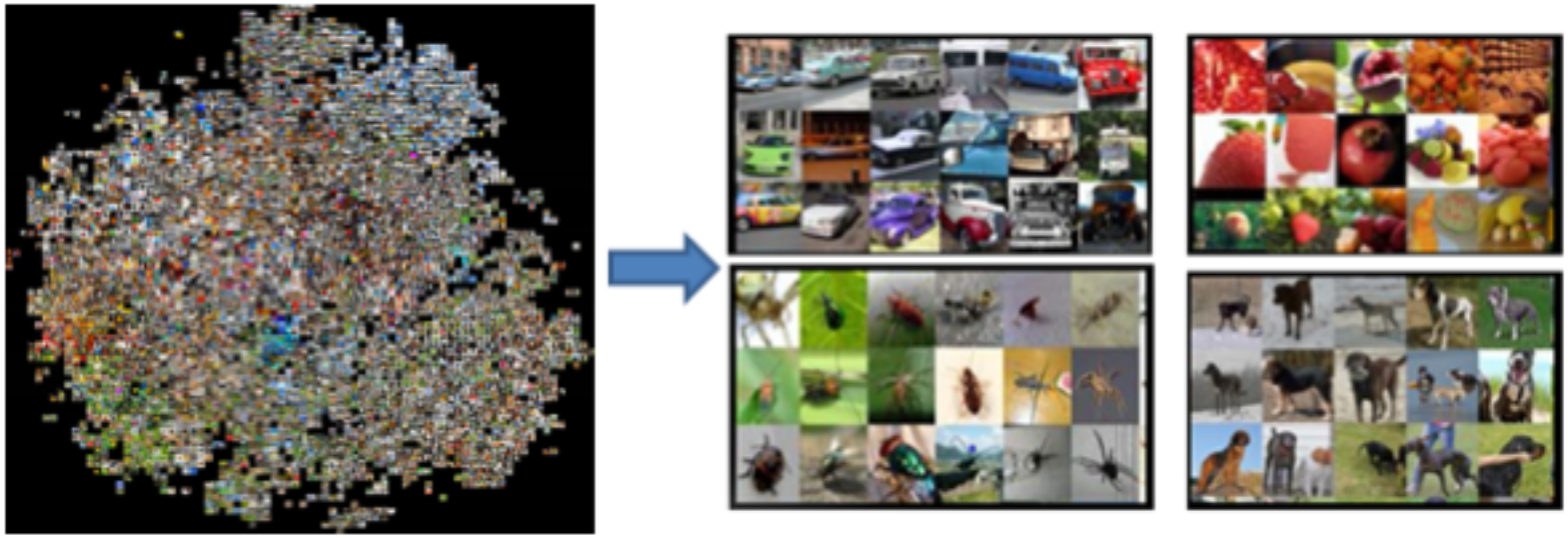
Important Application

ConvNets can be interpreted as gradually transforming the images into a representation in which the classes are separable by a linear classifier. We can get a rough idea about the topology of this space by embedding images into low dimensions and t-SNE is one powerful technique that can be used for embedding high-dimensional vectors in a low-dimensional space while preserving the pairwise distances of the points.



Important Application

To produce an embedding, take a set of images and Forward propagating each image through a trained ConvNet to extract a vector for each image class (e.g. in AlexNet the 4096-dimensional vector right before the classifier can be used for embedding) and then plug the vectors into t-SNE and get a 2-dimensional vector for each image.



Important Application

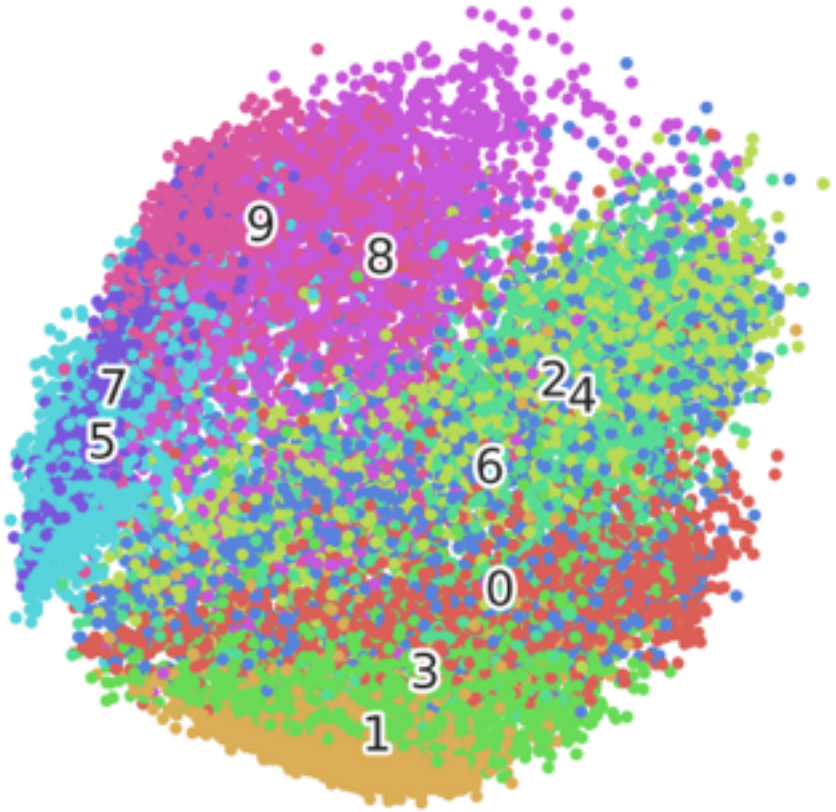
Images that are nearby each other are also close in the CNN representation space, which implies that CNN “sees” them as being very similar.

Cluster containing black cats.

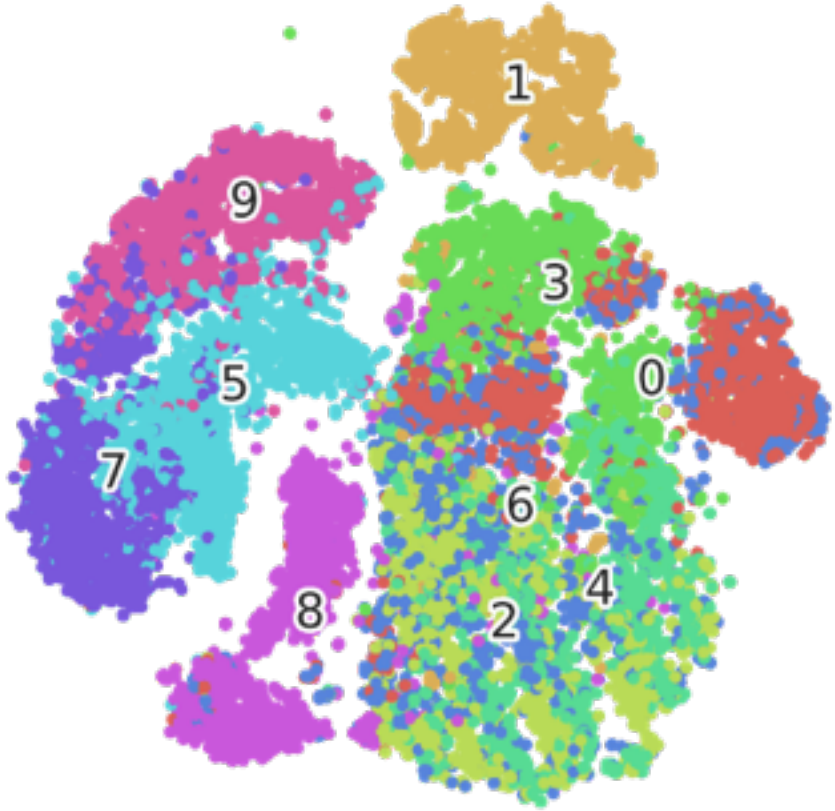
Cluster containing gray cats.

PCA vs t-SNE

you will see the output of PCA on the Fashion-MNIST dataset to compare it with t-SNE.



PCA



t-SNE

PCA vs t-SNE

some key differences between PCA and t-SNE can be noted as follows:

•t-SNE is computationally expensive and can take several hours on million-sample datasets where PCA will finish in seconds or minutes.

•PCA it is a mathematical technique, but t-SNE is a probabilistic one.

•Linear dimensionality reduction algorithms, like PCA, concentrate on placing dissimilar data points far apart in a lower dimension representation. But in order to represent high dimension data on low dimension, non-linear manifold, it is essential that similar data points must be represented close together, which is something t-SNE does not PCA.

•Sometimes in t-SNE different runs with the same hyperparameters may produce different results hence multiple plots must be observed before making any assessment with t-SNE, while this is not the case with PCA.

•Since PCA is a linear algorithm, it will not be able to interpret the complex polynomial relationship between features while t-SNE is made to capture exactly that.

References

- CE-717: Machine Learning _ Sharif University of Technology _ Fall 2020 _ Soleymani
- www.en.wikipedia.org/wiki/Principal_component_analysis
- www.cs.tau.ac.il/~rshamir/abdbm/pres/17/PCA.pdf
- https://heartbeat.fritz.ai/understanding-the-mathematics-behind-principal-component-analysis-efd7c9ff0bb3
- https://www.youtube.com/watch?v=NEaUSP4YerM
- https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding
- https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a
- https://www.datacamp.com/community/tutorials/introduction-t-sne
- https://www.researchgate.net/post/Does-Random-Projection-have-advantages-over-PCA
- https://medium.com/@lwj.liuwenjing/how-to-get-eigenfaces-a9caeeba8767
- https://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf?fbclid=IwAR0Bgg1eA5TFmqOZeCQXsloL6PKrVXUFaskUKtg6yBhVXAFFvZA6yQiYx-M
- https://medium.com/analytics-vidhya/deep-learning-visualization-and-interpretation-of-neural-networks-2f3f82f501c5

Any Questions?!