"You can Compress!!!"

Dimensionality Reduction - Why? What? How?

By :- Manan Pal Singh, NSIT
Nilay Shrivastava, NSIT

Speakers

Manan Pal Singh

- CS undergrad at NSIT Delhi
- Interest domains: Computer Vision, Web Security, Cryptography, Mathematics.
- Past-time hobby: Solving math problems, reading AI and security blogs.
- My motive is to contribute to the progress in the AI field, and working on some missing theories in the field to develop a better understanding.

Speakers

Nilay Shrivastava

- CS undergrad at NSIT Delhi.
- Interest domains: Mathematics, Physics and Computer Science.
- Past-time hobby: Read about Modern Physics and AI.
- Keenly interested in the theoretical foundations of AI.
- My dream is to be a part of the community of people working to solve intelligence.

What lies ahead?

“An awesome presentation of course” - Aristotle

- What is dimensionality reduction, why do we need it
- Diving into PCA (basic intuition and formulation)
- Inside t-SNE
- Autoencoders
- When to use what

What is dimensionality reduction?

Dimensionality Reduction (D.R)

D.R is the process of reducing the degrees of freedom of given dataset.

In other words expressing some information (in this case dataset) in the most compact way without any significant loss of content (information).

The ‘information’ is described using a fixed number of variables. We intend to reduce this number (of variables).

What are the Variables?

Variables are the features of the dataset.

Every entry of the dataset is uniquely defined by a certain combination of these variables.

Just like a point in Cartesian Geometry, we can treat these features as co-ordinates of dataset ‘space’.

N-feature dataset => n co-ordinates => n dimensions

Why Dimensionality Reduction

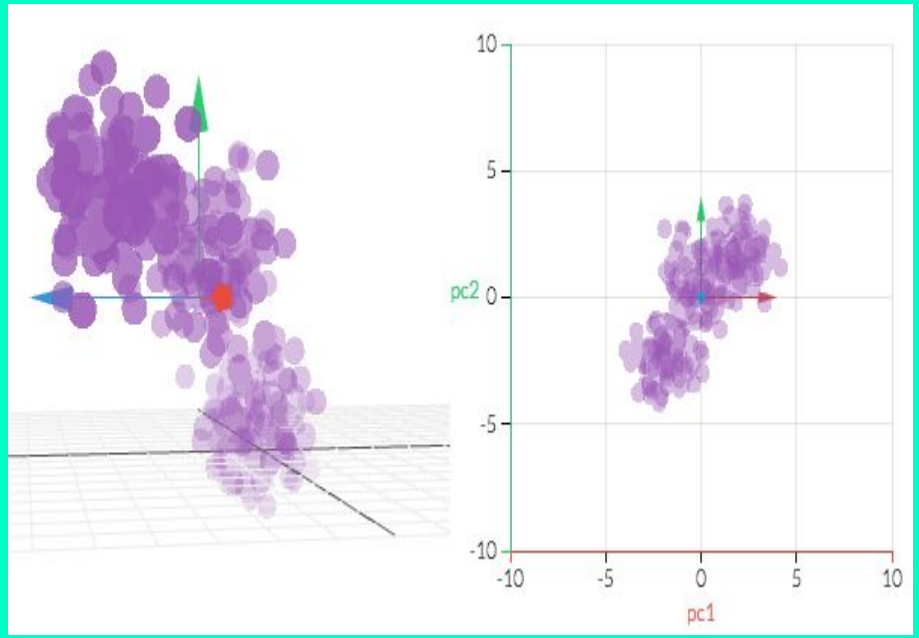
"Why on earth will anyone do that???" - Galileo

Reasons

- To dilute the ‘curse of dimensionality’
  - Reduce the computational complexity
  - Reduce sparsity
- For feature extraction
- For Visualization
  - D.R algorithms to reduce the dimensions to 2 or 3 for human visualisation
  - Necessary for debugging/understanding data algorithms and datasets themselves.

Principle Component Analysis

"WTF does that mean?" - DJ Trump

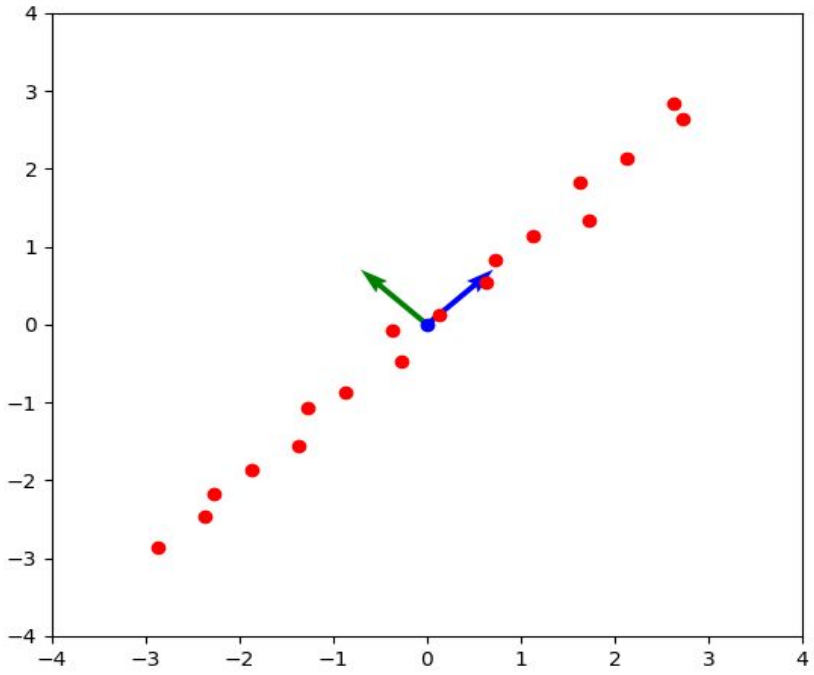


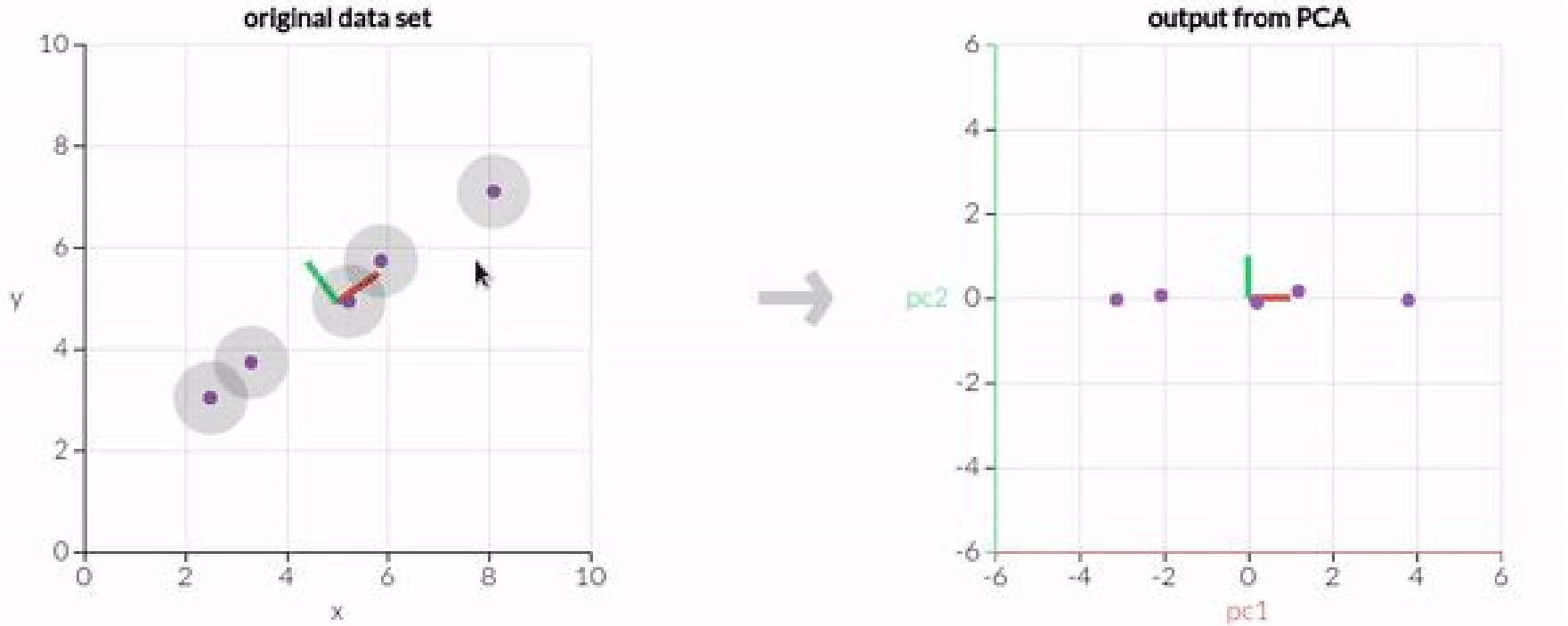
Basic Intuition Behind PCA

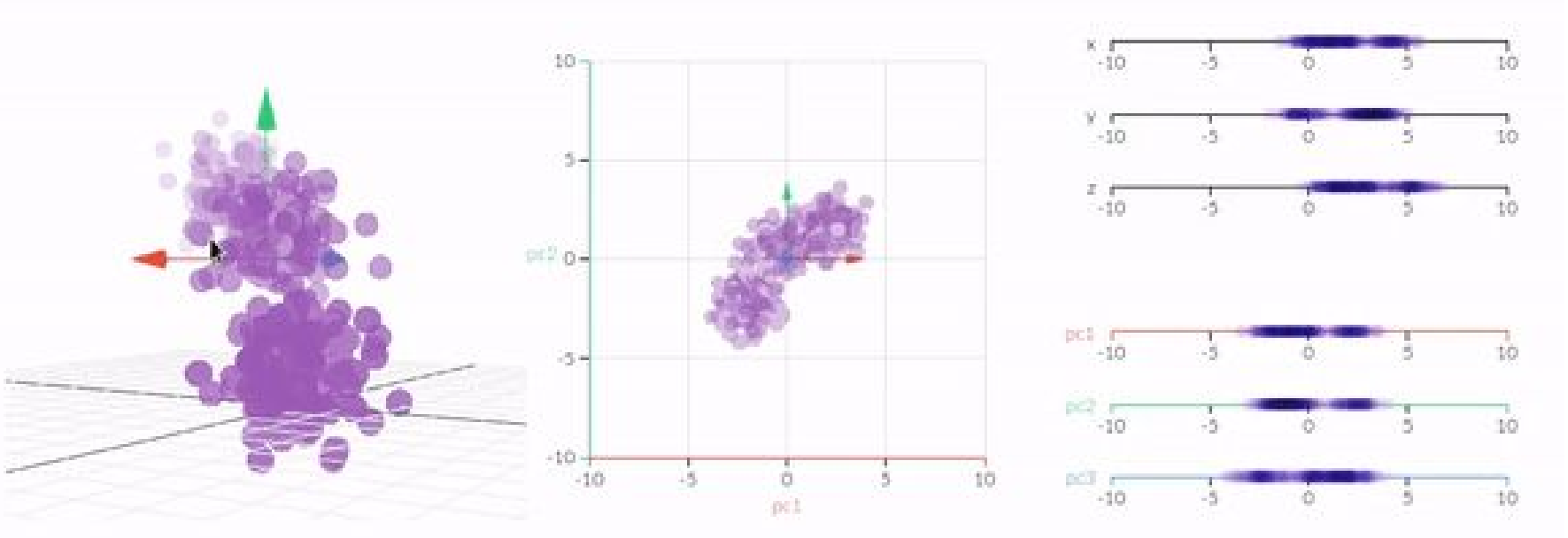
- PCA can be thought of as the process of choosing an appropriate set of axes. (Similar to what we do while solving Physics problems)
  - We rotate our original axes and then drop the component(s) on which the projection of our data-points is small
  - The above process is performed so that there is little loss of information.
- The axes are chosen in a way such that the maximum variance in data is retained
  - Of course the number of axes selected is less than the original dimension.

A SMALL VISUALISATION TO THE RESCUE!!

- The red dots are the data points (of a 2D dummy dataset)
- The blue and green arrows are the rotated axes.
- Finally we will drop the green arrow and represent data as projection on the blue arrow (principal component)

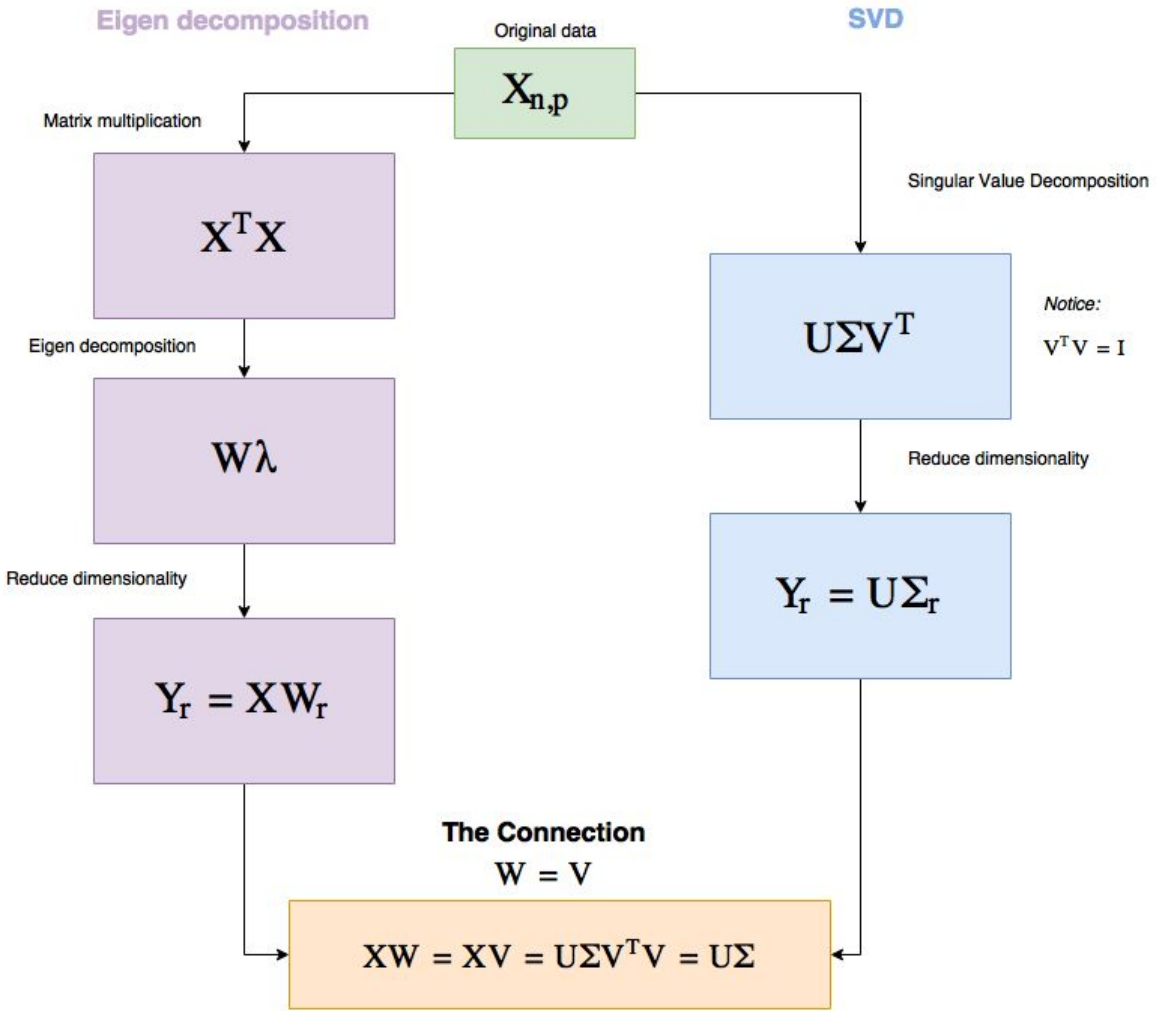



original data set
10
8
6
4
2
0
y
0
2
4
6
8
10
x
output from PCA
6
4
2
0
-2
-4
-6
pc2
-6
-4
-2
0
2
4
6
pc1


10
5
0
-5
-10
pc2
-10
-5
0
5
10
pc1
x
y
z
pc1
pc2
pc3

So how to calculate these directions?

- Decide how many dimensions to compress your data (say k) into.
- Calculate the “eigenvectors” of the “covariance” matrix of the dataset.
- Sort the eigenvectors in the decreasing order of their “eigenvalues”.
- Choose the top k eigenvectors (These are your new axes).
- Project your data-points on these eigenvectors.


Eigen decomposition
Original data
SVD
$X_{n,p}$
Matrix multiplication
Singular Value Decomposition
X^TX
$U\Sigma V^T$
Notice:
$V^TV = I$
Eigen decomposition
$W\lambda$
Reduce dimensionality
Reduce dimensionality
$Y_r = XW_r$
$Y_r = U\Sigma_r$
The Connection
$W = V$
$XW = XV = U\Sigma V^TV = U\Sigma$

Isn't this looking eerily similar to Linear Regression?

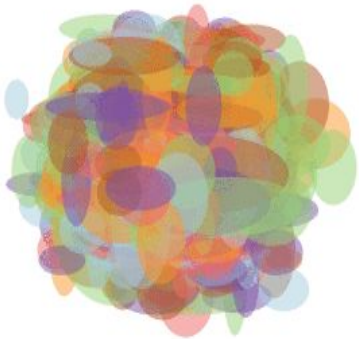
- Both PCA and Linear Regression are framed as Optimization problems.
- Both try to minimize the mean squared loss.
- Loss function of PCA :- "Find the axes for which the sum of squared distance of data-points from these axes is minimum."
- The algorithm is actually the solution of this optimization.

Some Points on pca

- Deterministic
  - Apply the algorithm on same dataset any number of times, will get the same output.
  - Technically PCA is returning a function.
- Computation:
  - Efficient with linear algebra libraries like Numpy, LAPACK, Eigen.
- Visualisation:
  - Preserves variance of the dataset.
  - Good for linearly correlated datasets.
  - Local neighbourhood isn’t preserved to a large extent (clustering not apparent).

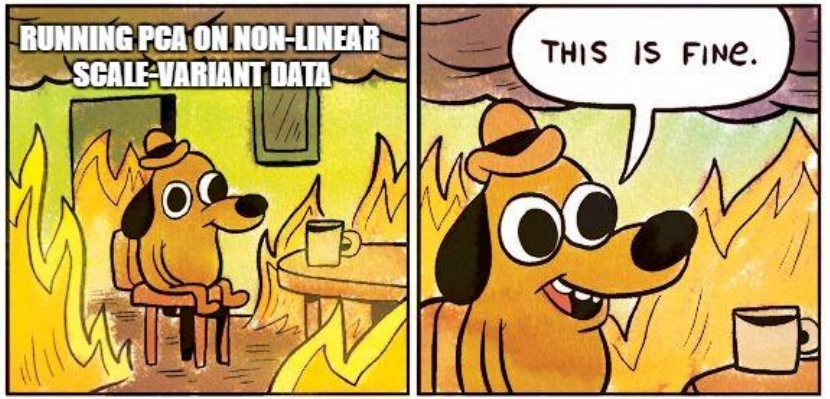
T-SNE

T-distributed Stochastic Neighborhood Embedding



Why t-SNE?

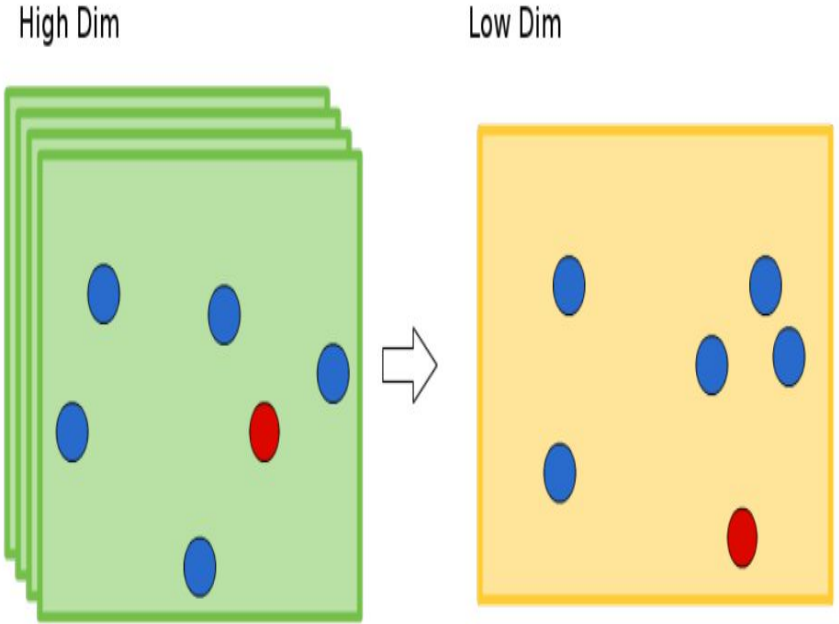
- t-SNE specializes in preserving local neighbourhood relation. (Methods like PCA don’t).
- t-SNE visualisation is therefore one of the best for clustered datasets
- One can tune t-SNE parameters to fit the visualisation according to need.



Local Neighbourhood Problem

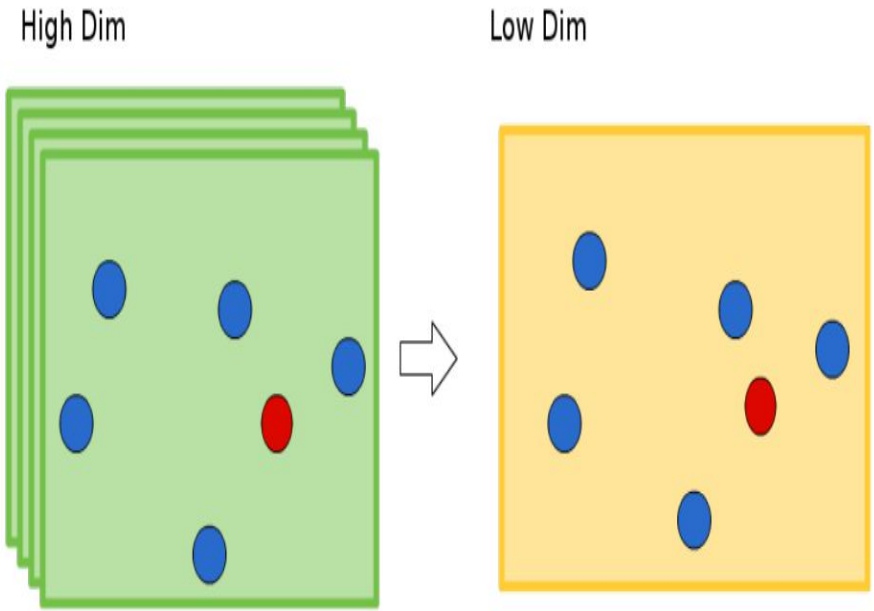
Neighbourhood relations (clusters) may or may not be maintained.

This is a common issue in PCA when applied to categorical data.



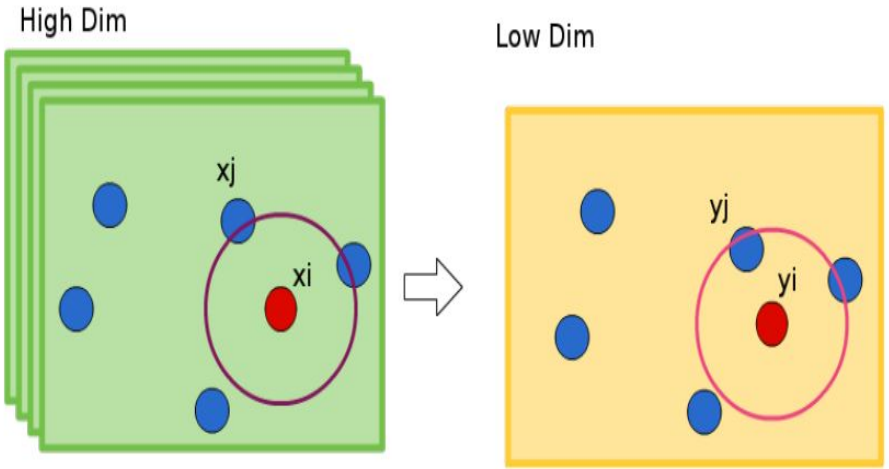
Preserving the Neighbourhood relationship

That’s what we want!!!



How t-SNE solves it

t-SNE creates a gaussian based probability distribution in high dimension giving the probability ‘p’ that a data point i selects another data point j.



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

.. CONTINUED

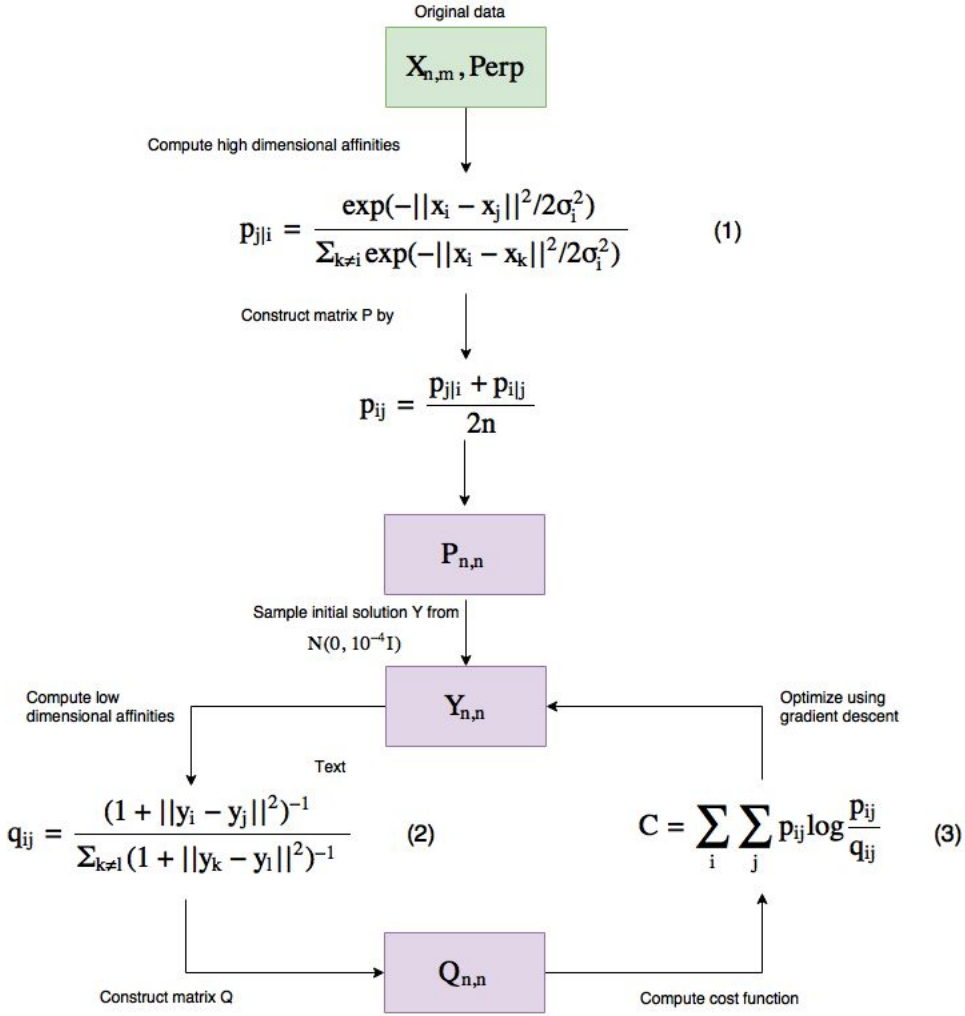
It creates a similar distribution for low dimension. Therefore

Similarity in higher dimension $$p_{ij} = \frac{exp(-||x_i - x_j||^2/2\sigma^2)}{\sum_{k \neq l} exp(-||x_l - x_k||^2/2\sigma^2)}$$

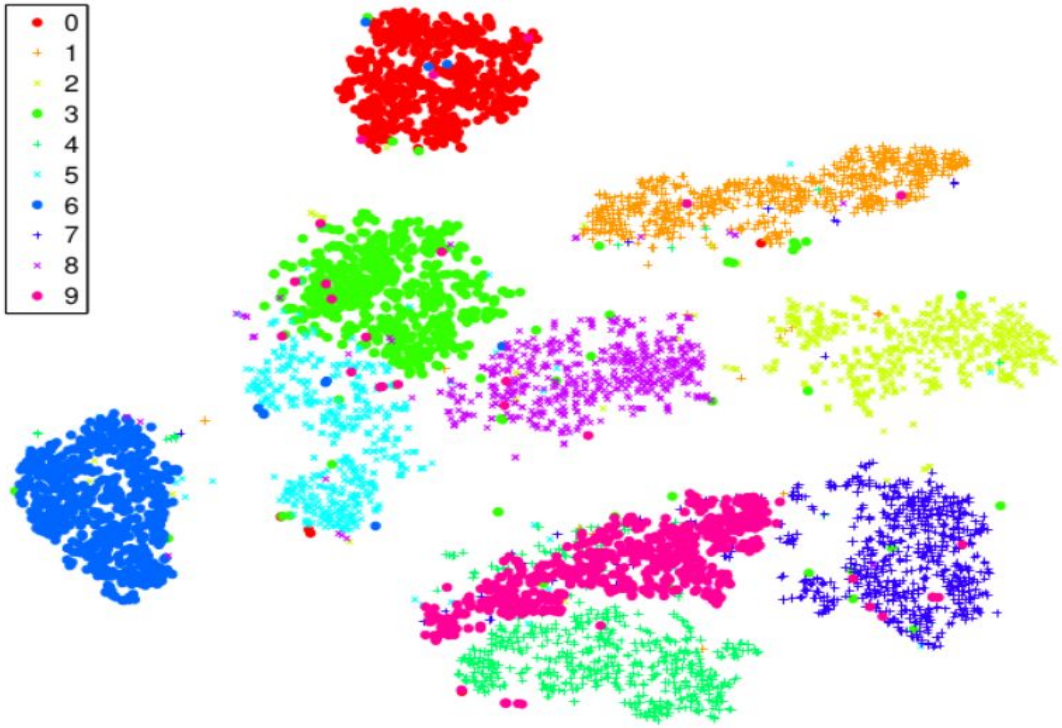
Similarity in lower dimension $$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l}(1 + ||y_k - y_l||^2)^{-1}}$$

Here y's are the lower dimensions representations that are learned using gradient descent.

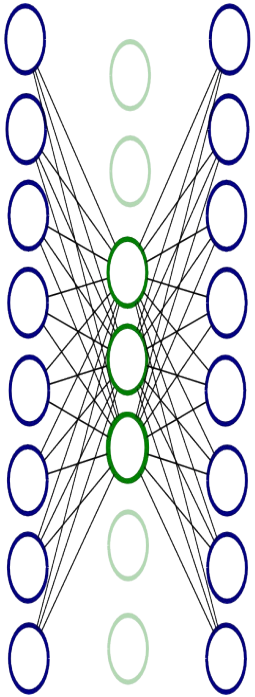
The cost function is the KL(p||q) $$C = KL(P||Q) = \sum_i \sum_{j \neq i} p_{ij} log \frac{p_{ij}}{q_{ij}}$$

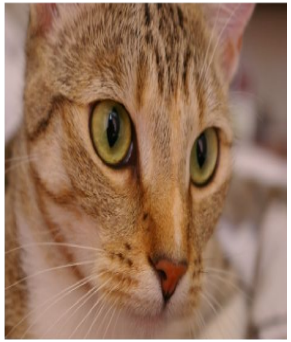

Original data
$X_{n,m}$, Perp
Compute high dimensional affinities
$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k \neq i} \exp(-||x_i - x_k||^2/2\sigma_i^2)}$ (1)
Construct matrix P by
$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2n}$
$P_{n,n}$
Sample initial solution Y from
$N(0, 10^{-4}I)$
$Y_{n,n}$
Compute low dimensional affinities
Optimize using gradient descent
Text
$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_{k \neq l} (1 + ||y_k - y_l||^2)^{-1}}$ (2)
$C = \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}}$ (3)
$Q_{n,n}$
Construct matrix Q
Compute cost function

Results on mnist (t-SNE)



Results on MNIST (PCA)



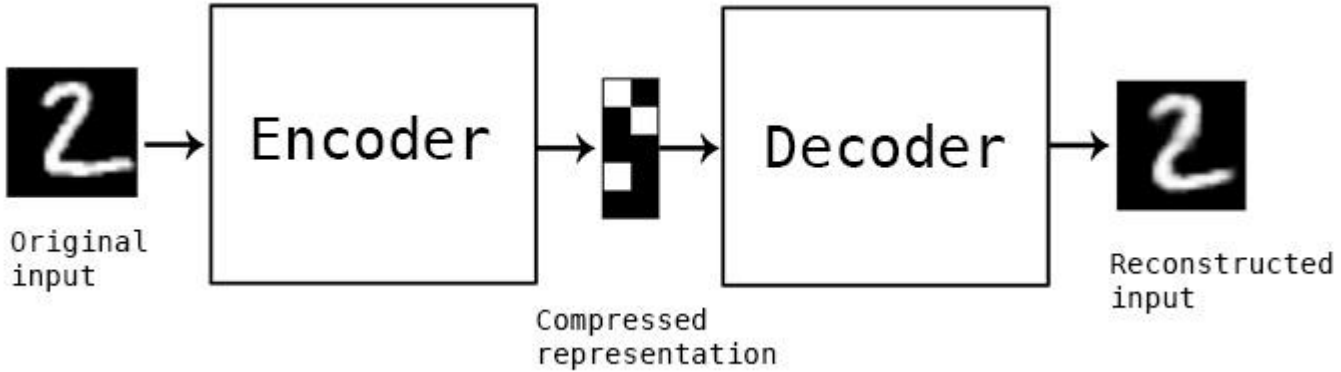


Auto-Encoder

Autoencoders

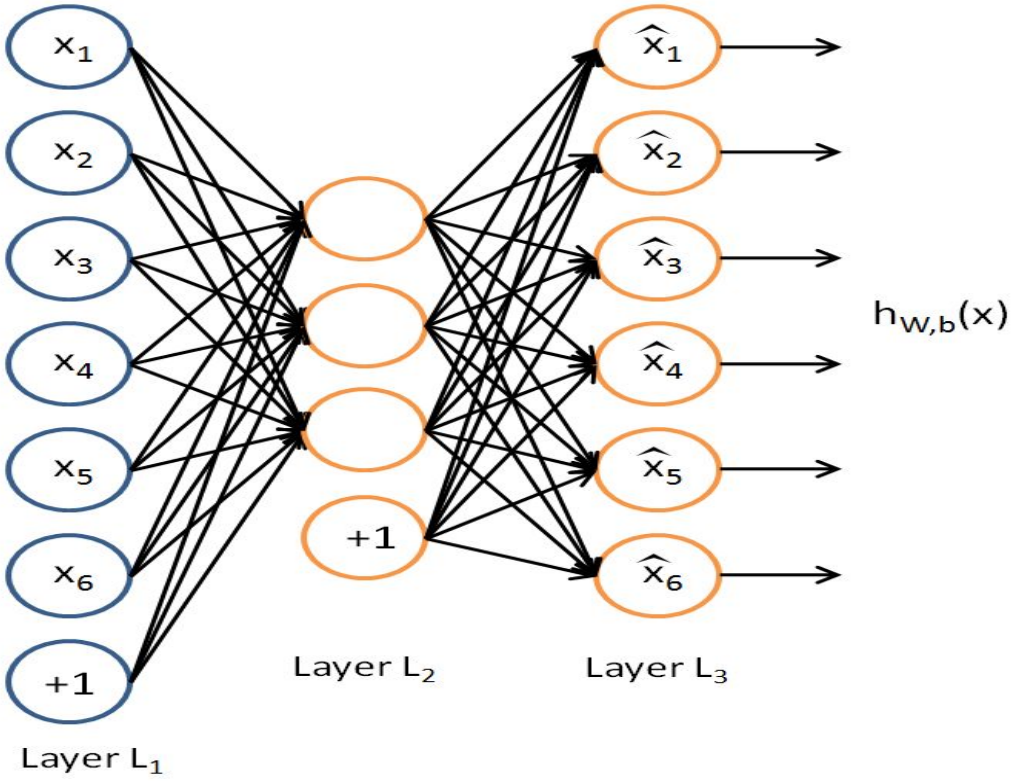
- Simplest encoder-decoder models.
- Train a 1-(hidden) layer Neural Network to learn the Identity function
  - Give x as input, and the network to give x as output
  - The input and output layer are of same size.
  - The middle layer is (considerably) smaller.
- After training we remove the output (decoder) and use the input to project the data to smaller dimension.

Like This ...


Encoder
Decoder
Original input
Compressed representation
Reconstructed input

By constraining the size of the hidden layer, we force the autoencoder to learn efficient representation of the data.

Interestingly the autoencoder projection to latent space is similar to PCA.



When should You Use Autoencoders?

- Autoencoders are used for denoising images.
- Project your data to auto-encoder latent space and use it for nearest neighbour classification.
- Forcibly making a neural network to learn hidden representation ensures that the latent space projections have less redundancy and are a better representation of data, use this as input for SVMs or any other algorithm.

Looking Ahead

Rule(s) of thumb ...

- Use t-SNE for visualisation.
- Use PCA for cases where deterministic solutions are desirable.
- Use PCA when it is known that there exists some correlation in data.
- Use PCA for new unseen data in a given problem.
  - PCA provides the eigenvectors for projection.
  - No such framework present in t-SNE.

This is not the end

- There are other algorithms like Linear Discriminant Analysis, Generalized Discriminant Analysis.
- Using Convolutional Neural Networks as feature extractors (like autoencoders).
- Using autoencoders as a lossy compression algorithm.

Thank You!!!