Dimension Reduction: PCA and t-SNE

Presentation Subtitle

Ritadip Bharati & Sudip Kumar Kar

School of Physical Sciences,
National Institute of Science Education and Research

April 21, 2023

Part I: Principal Component Analysis

Part II: t-Distributed Stochastic Neighbor Embedding

Part I

Principal Component Analysis

Introduction

- Principal Component Analysis (PCA) [Mishra 2023] is a statistical technique used for reducing the number of dimensions in a dataset while retaining the variation in the data as much as possible.
- It involves identifying the underlying structure in the data by finding linear combinations of the original features i.e. the principal component that accounts for the most variance in the data.
- The resulting principal components are orthogonal to each other, and the first principal component explains the most variance in the data.
- PCA can be used for data visualization, feature extraction, and noise reduction in various fields such as finance, biology, engineering, and social sciences.
- PCA can be performed using various software packages, including Python's scikit-learn library, MATLAB, and R.

How does PCA work?

The primary motive in PCA is to find the axes that capture the variance in the data most effectively. The steps involved in the PCA of a two-dimensional data are shown in Fig. 1. It can be extended to any number of dimensions with ease.

- First, the data is centered about the origin.
- To start with, a random line is drawn, and the squared distances along the line are computed and added to give the SS score. Maximizing the SS score is the same as minimizing the sum of the square of perpendicular distance from the line since the distance from the origin is fixed. Thus, gradient descent is performed to find the line that minimizes the sum of squares of the perpendicular distances. This line is called the PC1 line
- This step is repeated in the orthogonal plane. In two dimensions, this step is simple as the choice of the axis is limited to only one, so that is what is chosen as the PC2 line. For d-dimensions, this process is repeated d times.
- Then the SS score of each PC line is plotted together; such a plot is called the 'Scree plot'. The axes with the lowest SS scores are eliminated to give a dataset with a reduced number of dimensions.

Principal Component Analysis

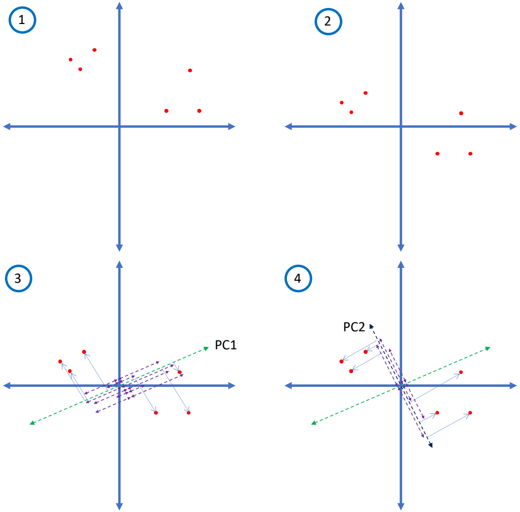

Figure. A visual representation of PCA in action, the first plot is the original data, and the subsequent plots show the process of PCA described before. The dotted blue lines are perpendicular distances to the line and the purple lines are the distances used for calculating the SS score.

IMPLEMENTATION OF PCA ON THE IRIS DATASET

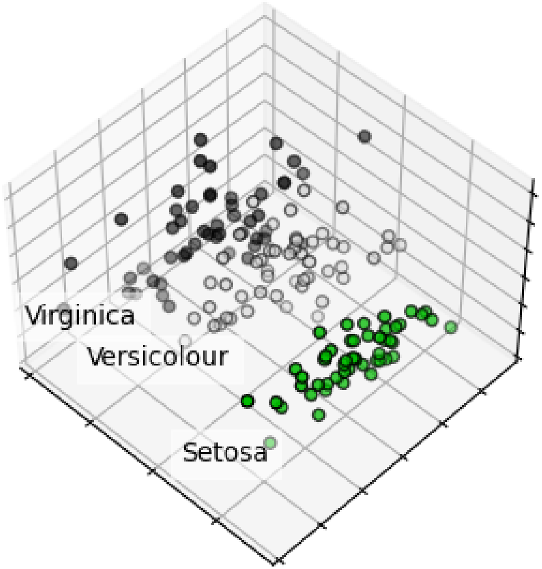

Figure. The figure shows a PCA implementation on the Iris dataset used to reduce the four-dimensional dataset to a three-dimensional one.

Shortcomings of PCA

- If the data is symmetrical around the principle axis, then PCA does not conserve variation well.
- If the data is not standardized, i.e, the mean of the values of points is not 0, or the standard deviation is not 1, then PCA displays biases towards a few dimensions or a few features.

So, we use t-SNE as a dimensionality reduction algorithm to avoid these biases.

Part II

t-distributed Stochastic Neighbor embedding

Introduction

In this section, we go over t-distributed stochastic neighbor embedding(t-SNE), which is an unsupervised machine learning algorithm that is primarily used for reducing higher dimensional data to lower dimensions for the purpose of visualization or otherwise. Other forms of dimension reduction algorithms tend to preserve global structure while ignoring local features. Stochastic neighbor embedding methods focus on the local structure to create maps to a lower dimension, thereby preserving local structure.

The model was created by Geoffrey Hinton [Maaten and G. Hinton 2008] and Laurens van der Maaten in 2008 as an attempt to improve the Stochastic Neighbor Embedding algorithm developed by Sam Roweis and Geoffrey HintonG. E. Hinton and Roweis 2002. The central idea was to solve the *crowding* problem that appears when a Gaussian is used for calculating the similarity between two points.

Stochastic Neighbor Embedding

The closeness of two data points x_i and x_j in some d-dimensional dataspace is defined by a joint probability, p_{ij} as,

$$p_{ij} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{j' \neq i} \exp(-||x_i - x_{j'}||^2/2\sigma_i^2)} \tag{1}$$

Each point has a specific *bandwidth* given by σ_i. This is done so that every point effectively has the same number of neighbors irrespective of density. The number of neighbors is a parameter that can be adjusted by *perplexity*.

In the original stochastic neighbor embedding, the closeness of two points, y_i and y_j, in the lower dimensional space is given by a Gaussian distribution,

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

Stochastic Neighbor Embedding

The discrepancy between the low-d distribution and the high-d distribution is quantified by the Kullback–Leibler(KL) divergence. This quantity signifies the information entropy of one distribution relative to another. The mathematical form is given as,

$$C = KL(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{3}$$

This quantity acts as the loss function for the gradient descent procedure due to its innate tendency to preserve the local structure. For example, if high p_{ij} is modeled by a low q_{ij}, the loss is high; on the other hand, low p_{ij} modeled by low q_{ij} does not incur a huge penalty. It should be noted that this loss function is not convex, so different initializations can lead to different mappings.

The prime drawback of this method is seen while reducing data that is intrinsically high dimensional e.g., vertices of a higher dimensional hypercube. The Gaussian has a sharp peak; thus, it has the tendency to pull everything towards the center in the gradient descent mechanism, and as a consequence, everything collapses at the center. This is called the *Crowding problem*.

SYMMETRIZING THE DISTRIBUTION

The first step towards t-SNE involves symmetrizing the distribution function. This is achieved by considering the conditional probability instead of the joint probability,

$$p_{j|i} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{j' \neq i} \exp(-||x_i - x_{j'}||^2/2\sigma_i^2)} \tag{4}$$

The joint probability is then connected to the conditional probability as,

$$p_{i|j} = \frac{p_{ij}}{p_j} = \frac{p_{ij}}{N} \qquad p_{j|i} = \frac{p_{ij}}{p_i} = \frac{p_{ij}}{N} \tag{5}$$

where the probability of selecting a point randomly from the set of points is $1/N$. Thus the joint probability is

$$p_{ij} = \frac{p_{i|j} + p_{j|i}}{2N}$$

Incorporating the t-distribution

The choice of probability in the lower-*d* space is where the student's t-distribution comes into the picture,

$$q_{ij} = \frac{(1 + ||y_i - y_j||^2)^{-1}}{\sum_k \sum_{l \neq k} (1 + ||y_i - y_j||^2)^{-1}} \tag{6}$$

This distribution has a heavier tail and hence allows dissimilar points to be far apart, thus solving the crowding problem.

With this consideration, the data points in the low-*d* space evolve using the gradient descent mechanism while trying to minimize the KL divergence.

Barnes-Hut SNE

Gradient descent of KL-divergence involves computing the joint probability of every pair, and this takes $\mathcal{O}(N^2)$ time, which might be quite significant for a large dataset.

To speed up this process, the Barnes-Hut algorithm[van der Maaten 2013] is used. This involves calculating the ‘center of mass’ of various clusters to quickly evaluate the ‘force’ that drives the gradient descent algorithm. This approximation results in a time complexity of $\mathcal{O}(N \log N)$.

t-SNE on some examples

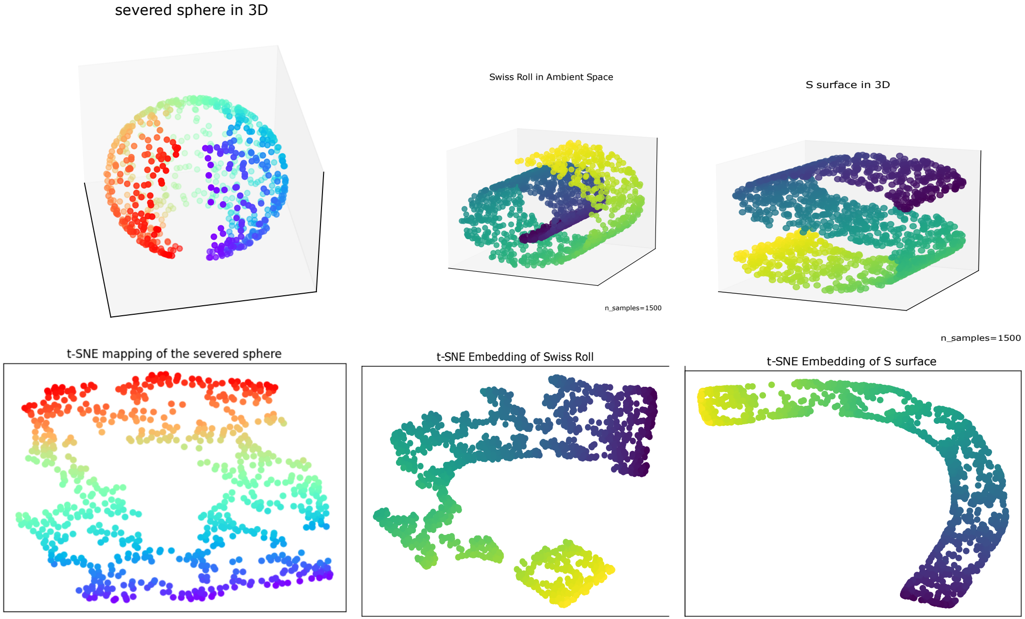

Figure. t-SNE on some 3-d datasets; the colors in the higher dimensional space are faithfully reproduced in 2-d

CAVEATS

The t-SNE result is highly dependent on the value of perplexity(usually chosen between 5 to 50). A low perplexity can give rise to non-existent clustering in the 2-d map, and a high perplexity might hide the separation between natural clusters. The t-SNE procedure ignores density by design. Thus, it is important to keep in mind that the density of points in a t-SNE embedding does not mean anything. To understand the topology of the data, it is always wise to experiment with different perplexity values and then derive conclusions.[Wattenberg, Viégas, and Johnson 2016]

References I

- Hinton, Geoffrey E and Sam Roweis (2002). "Stochastic Neighbor Embedding". In: *Advances in Neural Information Processing Systems*. Ed. by S. Becker, S. Thrun, and K. Obermayer. Vol. 15. MIT Press. URL: https://proceedings.neurips.cc/paper_files/paper/2002/file/6150ccc6069bea6b5716254057a194ef-Paper.pdf.
- Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86, pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.
- Mishra, Subhankar (2023). *Unsupervised Learning - Dimensionality reduction PCA*.
- van der Maaten, Laurens (Jan. 2013). "Barnes-Hut-SNE". In: *arXiv e-prints*, arXiv:1301.3342, arXiv:1301.3342. DOI: 10.48550/arXiv.1301.3342. arXiv: 1301.3342 [cs.LG].
- Wattenberg, Martin, Fernanda Viégas, and Ian Johnson (2016). "How to Use t-SNE Effectively". In: *Distill*. DOI: 10.23915/distill.00002. URL: http://distill.pub/2016/misread-tsne.