Development of unsupervised learning transformations through supervised learning methods.

Author: Patricia Cortajarena Sauca

Ponente: Carlos Roberto del Blanco Adán

Tutor: Pedro Morales

Trabajo Fin de Grado

ETSIT UPM

Madrid. January, 2018

Abstract

The aim of this project is

Acknowledgements

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Working with large datasets and high-dimensional data in nowadays' problems has encouraged the use of dimensionality reduction algorithms which try to preserve as much information as possible even decreasing the number of features needed to describe that same dataset. Thus, time and memory in huge implementations could be saved.
Taking into account that this turns into a difficult task, we can find that numerous approaches have been proposed.
Although they look forward to achieve more or less the same performance, they differ from one another and we can not reassure which would suite for a specific problem or even if the behaviour of the algorithm is going to reach the results we expected or needed.
The first point to take into account is the existence of parametric and non parametric algorithms, and secondly, in both of them we can find different models proposed depending on what to optimize, yet not everything is going to be preserved as well as in the original dataset, so we need to prioritize some aspects.
So our decision of which to implement depends on the previous study of our data, the performance requirements and the later purpose and usage of the reduced data.

We propose the research and then base our study in the next dimensionality reduction algorithms:

- PCA (Principal Component Analysis)
- MDS (Multidimensional Scaling)
- TSNE (T-Stochastic Neighbour Embedding)

1.1 PCA: Principal Component Analysis

Principal Component Analysis algorithm is based on reducing the number of features by processing the correlations between the features of the datapoints. The aim is to eliminate this correlations by transforming the matrix $\mathbf{X} \in \Re^{mxn}$ (with m being the number of data points and n de number of features) into an orthogonal basis. By omiting the correlation between columns of the matrix $\mathbf{X}$ we are capable of doing away with redundancies.
The model starts by computing de covariance matrix, which results in a $\Re^{nxn}$ symmetric matrix. We obtain it by using the next expresion:

$$\text{cov}(\mathbf{X}) = \frac{1}{m-1}\,\mathbf{X}^T\mathbf{X}$$

Because the aim of the PCA is to eliminate the correlations, the covariance matrix of the result $\mathbf{Y}$ should be a diagonal matrix with just the variances of the columns.
PCA is famous because of a great advantage: we can find a linear transformation ($\mathbf{Y = XP}$), which makes this a parametric model, easy to reuse and quite computationally simple because of some covariance matrix calculation approaches.
For symmetric matrices ($\mathbf{X}$) we can find eigenvalue decomposition with a diagonal matrix ($\mathbf{Y}$), matching exactly with our linear problem with $\mathbf{X}$ and $\mathbf{Y}$.

$$\mathbf{Y = XP}$$

$$cov(\mathbf{Y}) = \frac{1}{m-1}\mathbf{Y}^T\mathbf{Y} = \frac{1}{m-1}(\mathbf{XP})^T\mathbf{XP} = \mathbf{P}^T cov(\mathbf{X})\mathbf{P}$$

$$\mathbf{D} = \mathbf{V}^T\mathbf{AV}$$

$$\mathbf{A} = cov(\mathbf{X}); \mathbf{P} = \mathbf{V}^T; \mathbf{D} = cov(\mathbf{Y})$$

With the previous expresions we get to the point that computing the eigenvectors of the covariance matrix $\mathbf{X}$ we can get a linear transformation from space $\mathbf{X}$ to space $\mathbf{Y}$. The eigenvalues matrix obtained ($\text{cov}(\mathbf{Y})$) sorted decreasingly would be the orthogonal basis values. Choosing the $\mathbf{N}$ first values of this matrix, being $\mathbf{N}$ the desired output dimension, and computing the corresponding eigenvectors, we would obtain our reduced dimensionally points, as a basis transformation of our datapoints from the original dataset, by the new basis coordinates.

1.2 MDS: Multidimensional Scaling

Multidimensional Scaling in dimensionality reduction tries to create a map which displays the relative distances between the data points. This focuses on getting a one, two or, at most, three dimensional map, keeping as much distance information as possible.
MDS calculates a metric or non-metric solution depending on the data provided, which has to be a *proximity* matrix. This *proximity* matrix quantifies how close the datapoints are. In the one hand for metric solutions, this *proximity* matrix has to be a true distance matrix, while in the other, both dissimilarities or correlations could be the input to the problem's matrix.

MDS algorithm is based on some ideas explained in the previous section. As we can see, the *proximity* matrix is always symmetric and somehow describes the relations between the features, so basically we can treat our problem as a variation of the PCA algorithm.
Metric MDS performs the same steps as in PCA, with the modification of being a distance matrix the one computed in this problem.
In non metric MDS, we assume a less strict relation and we compute the observed distances as a function of the real distance plus some meassure error. The usable information in this case would be the rank order of the previous matrix, which could be the input for the model.
The main distinction between PCA and MDS is the fact that, because of the need to compute a pairwise distance or proximity matrix, there is no linear transformation that suits both the distance computations plus the matrix operations, so MDS turns to be non-parametric.

1.2.1 Stress metric

As in every data problem, we need a metric which shows how well is the performance given a particular dataset. In Multidimensional Scaling we compute the *stress* measure that compares the predicted distances with the original ones. Note that obviously this depends on the number of dimensions we want to keep, yet if the dimensions lower, the stress will get higher, because we are representing the same distances relations in a lower dimensional space.

$$stress = \sqrt{\frac{\sum (d_{ij} - d'_{ij})^2}{\sum d_{ij}^2}}$$

Regarding the previous expresion, if our prediction stands well for the original data, the stress value should lower, relating zero stress to the perfect performance of the MDS algorithm.

1.3 TSNE: T-Stochastic Neighbour Embedding

T-Stochastic Neighbour Embedding is our non-linear example of dimensionality reduction. It relies on Stochastic Neighbour Embedding (SNE) which will be explained right below. The main characteristic why this algorithm is used is because it is capable of visualizating both the local and the global structure of the original data. As said, this section will be divided in two: the basis SNE and the T-SNE upgrades.

1.3.1 SNE

SNE approaches the dimensionality reduction by converting the Euclidean distances into conditional probabilities as a way of expressing similarities between points. That means we measure the similarity of two points x_i, x_j as the probability $p_{i|j}$ of considering the second one as a neighbour of the first. The probability in the original and in the low dimensional space is computated as seen:

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

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

As the point of this metric is to compute similarities as probabilities, we can calculate the mismatch between $p_{i|j}$ and $q_{i|j}$ with all the datapoints and consequently obtain the algorithm's behaviour by analysing how many neighbours have been mantained in the low dimensional map. In terms of conditional probabilities, Kullback-Leibler divergence could suit this need. Summing up all the previous ideas we get to the point of minimizing the cost function described as:

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

The limitations of this algorithm are the non symmetric general expresion of the Kullback-Leibler divergence, the difficulty to optimize the cost function, the fact that we need to choose diferent values of the variance depending on the point and the "crowding problem". T-SNE tries to solve this limitations as in the next section is described.

1.3.2 T-SNE

Chapter 2

Framework

Bibliography

Code