From Principal Subspaces to Principal Components with Linear Autoencoders

Raquel Parrondo Pizarro
Universitat Pompeu Fabra

Abstract

Autoencoders are a Machine Learning unsupervised method extensively employed in deep learning. Particularly, the principal component loading vectors can be recovered from the weights of an autoencoder with a single fully-connected hidden layer, a linear activation function and a squared error cost function using the Singular Value Decomposition.

1 Background

Principal Component Analysis (PCA) is a multivariate statistical technique that applies a linear transformation of the feature vectors in a way that best captures the variance of data. The goal of PCA is to identify patterns in the data and find a way to represent the data in a lower-dimensional space, while still preserving as much of the original variation as possible. This is done by finding a set of orthogonal directions, called principal components (PCs). Through the eigendecomposition of the covariance matrix, we can capture a set of eigenvectors (corresponding to the principal components) ranked by the decreasing fraction of the total variance explained (defined by the eigenvalues) [1]. PCA is a powerful technique for reducing the dimensionality of the data when keeping only the first few PCs, among other relevant applications [5].

Auto-associative neural networks (ANNs, or autoencoders) are a deep learning method based on the sequential use of two unsupervised neural networks separated by the bottleneck layer: the encoder and the decoder; where output and input layers have the same number of nodes. The main idea behind autoencoders is learning a compressed representation of a data sample by training a neural network using backpropagation with an optimizer to reconstruct the original sample from the compressed representation minimizing the reconstruction error, which is often measured by the loss function [3]. They can be used for a variety of tasks, being dimensionality reduction and feature learning two of the most significant [6].

PCA and a linear 1-layer autoencoder with squared error cost function are closely related, given that its weights span the subspace spanned by the first loading vectors [2]. This article seeks to address the following query: is it possible to compute PCA using linear autoencoders?

2 Introduction

2.1 Principal Component Analysis (PCA)

Let Y be the original data matrix whose columns are the N observations vectors of dimension n. Considering $Y_0 \in \mathbb{R}^{n\times N}$ the centered matrix computed by subtracting the element-wise mean from each observation vector, it is possible to represent a linear transformation of a finite dimensional vector as a matrix multiplication:

$$X_0 = W^T Y_0$$

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

where $X_0 \in \mathbb{R}^{m \times N}$ is the matrix whose columns are the centered vectors of transformed observations.

When the transformation's matrix W^T represents principal component analysis, we denote $W = P$ and the *principal component loading vectors* corresponds to the columns of P.

The overall set of loading vectors can be computed as the resulting n orthogonal eigenvectors of the covariance matrix, denoted as $\Omega = Y_0 Y_0^T \in \mathbb{R}^{n \times n}$, corresponding to descending non-negative eigenvalues. As it is well known that symmetric matrices are diagonalizable, Ω is always diagonalizable given its symmetric nature and, therefore, P may be computed through the Jordan matrix decomposition of Ω:

$$\Omega = Y_0 Y_0^T = P \Lambda P^{-1} = P \Lambda P^T$$

where $\Lambda = X_0 X_0^T$ is a diagonal matrix whose diagonal entries correspond to the eigenvalues of Ω sorted in decreasing order.

2.2 Dimensionality reduction through PCA

PCA is frequently employed as a dimensionality reduction technique in order to decrease the number of variables in a model. Keeping the first m principal components ($m < n$), a truncated transformation is applied:

$$X_m = P_m^T Y$$

where $P_m \in \mathbb{R}^{n \times m}$ matrix has the first m loading vectors as its columns.

The loss of information is minimized by maximizing the variance of the selected principal components, i.e., by finding the corresponding eigenvectors to the m largest eigenvalues of the covariance matrix Ω.

2.3 Singular-Value Decomposition (SVD)

Eigendecomposition of the covariance matrix is equivalent to the Singular-Value Decomposition (SVD) of Y_0, whose factorization can be written as:

$$Y_0 = U \Sigma V^T$$

where the orthogonal columns of $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{N \times N}$ are referred to as left and right singular vectors, respectively, and the diagonal entries of the diagonal matrix $\Sigma \in \mathbb{R}^{n \times N}$ to as singular values, which are sorted in decreasing order. Thus, the eigenvalues of Ω are the squared singular values of Y_0, and the eigenvectors of Ω (i.e., the loading vectors of Y) are the left singular vectors of Y_0.

2.4 PCA on large data sets of high-dimensional data

The computation of the loading vectors through either the eigendecomposition of the covariance matrix or the SVD of the observations might be infeasible when we deal with datasets with numerous observations (large N) and in which each observation is high-dimensional (large n).

2.5 Minimum total squared reconstruction error

One of the characteristics of PCA is that the reconstructions of the observations from the m leading principal components have the minimum sum of total squared reconstruction error out of all conceivable linear transformations. In other words, P_m is a feasible solution (but not unique) to:

$$\min_{W \in \mathbb{R}^{n \times m}} ||Y_0 - W W^T Y_0||^2 \tag{1}$$

2.6 Linear autoencoders

The simplest case of linear autoencoder consists in only one hidden layer with fewer nodes and no non-linear activation function. In this instance, a vector $y_i \in \mathbb{R}^{n \times 1}$ which passes through the bottleneck outputs $x_i \in \mathbb{R}^{m \times 1}$ according to:

$$x_i = W_1 y_i + b_1$$

where $W_1 \in \mathbb{R}^{m \times n}$ and $b_1 \in \mathbb{R}^{m \times 1}$ referred to, respectively, as the weight matrix and the bias vector of the first layer. Equivalently, the second layer maps x_i to $\hat{y}_i \in \mathbb{R}^{n \times 1}$ according to:

$$\hat{y}_i = W_2 x_i + b_2 = W_2(W_1 y_i + b_1) + b_2$$

where $W_2 \in \mathbb{R}^{n \times m}$ and $b_1 \in \mathbb{R}^{n \times 1}$ are, respectively, the weight matrix and the bias vector of the second layer.

The loss function measuring the total squared difference between reconstructed output $\hat{y}_i$ and original input y_i [4] is minimized iteratively, through backpropagation and using an optimizer, along the training step of the autoencoder in order to find the parameters W_1, b_1, W_2, and b_2:

$$\min_{W_1, b_1, W_2, b_2} ||Y - W_2(W_1 Y + b_1) + b_2||^2$$

Setting the partial derivative with respect to b_2 to zero and then the gradients also to zero, W_1 is the left Moore-Penrose pseudoinverse of W_2 ($W_1 = W_2^+ = (W_2^T W_2)^{-1} W_2^T$) and, accordingly, the minimization expression remains with respect to a single matrix:

$$\min_{W_2 \in \mathbb{R}^{n \times m}} ||Y_0 - W_2 W_2^+ Y_0||^2 \tag{2}$$

The Moore-Penrose pseudoinverse generalizes the concept of the inverse of a matrix. Given a matrix $A \in \mathbb{R}^{n \times m}$, not necessarily square, the Moore-Penrose pseudoinverse is defined as the unique matrix $A^+ \in \mathbb{R}^{m \times n}$ computed as $A^+ = (A^T A)^{-1} A^T$. We can easily see that this problem is very similar to (1), which states that the m leading loading vectors are an orthonormal basis that spans the m dimensional subspace onto which the projections of the centered observations have the minimum squared differences from the original centered observations. In the light of this, W_2 is a minimizer of (2) if and only if its column space is spanned by the first m loading vectors of Y. However, the loading vectors cannot be recovered in this way.

Linear autoencoders are said to apply PCA to input data since it is projected onto the low dimensional principal space. Nevertheless, the coordinates of the output of the hidden layer are neither uncorrelated nor sorted in decreasing order of variance.

Despite the fact that various methods for neural networks that compute the exact loading vectors having been developed [7] [8], no approach has already been proposed for retrieving the loading vectors from a simple linear autoencoder.

3 Methods

The proposed method consists of recovering the first m loading vectors of Y to simply applying SVD to the weight matrix W_2 that minimizes the loss function and retrieving the first m left singular vectors (regardless the sign). The loading vectors can also be recovered from the weights of the hidden layer, since the first m loading vectors of Y are equal to the first m left singular vectors of $W_1^T \in \mathbb{R}^{n \times m}$. This work's prior attempt to prove this statements was found to be erroneous.

A potential caveat is that, occasionally, the loading vectors computed by this approximation can be obtained as a result of an underfitting procedure. To fix this problem, we must ensure that the selected sample is sufficiently representative of the entire dataset.

4 Conclusions

In the final analysis, the usage of autoencoders allows to compute PCA, besides finding the principal subspace. Training a linear autoencoder on the (non-centered) observations and then applying SVD to the weight matrix of one of the two layers, we can recover the loading vectors amounts.

Retrieving the loading vectors in this way: (i) the solution is unique, (ii) the coordinates of the transformed data are uncorrelated, and (iii) the coordinates are sorted in decreasing order of variance; properties not held for a general principal subspace.

The novelty of this approach lies in the range of benefits of performing PCA in this way which includes the ability to:

- Process high-dimensional data.
- Process datasets containing plenty of observations.
- Avoid the requirement to center the data.
- Be implemented easily using popular machine learning frameworks and algorithms in very few lines of code.

Regarding the first two points, the training of autoencoders can be accomplished using a several stochastic optimization techniques which can handle large amounts of high-dimensional training data. However, the optimization method employed to train the neural network has no bearing on the solution.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[3] Dor Bank, Noam Koenigstein, and Raja Giryes. Autoencoders. *arXiv preprint arXiv:2003.05991*, 2020.

[4] Jason Brownlee. Loss and loss functions for training deep learning neural networks, Oct 2019.

[5] Basna Mohammed Salih Hasan and Adnan Mohsin Abdulazeez. A review of principal component analysis algorithm for dimensionality reduction. *Journal of Soft Computing and Data Mining*, 2(1):20–30, 2021.

[6] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[7] Erkki Oja. Neural networks, principal components, and subspaces. *International journal of neural systems*, 1(01):61–68, 1989.

[8] Jeanne Rubner and Paul Tavan. A self-organizing network for principal-component analysis. *EPL (Europhysics Letters)*, 10(7):693, 1989.