MA2823: Introduction to Machine Learning
CentraleSupélec
Lab 1: Dimensionality Reduction

Instructor: Fragkiskos Malliaros

September 28, 2018

1 Description

The goal of this lab is to study several *dimensionality reduction* techniques and to examine how they can be applied on real data. Initially, we give a brief introduction of the basic dimensionality reduction techniques that will be used in this lab, and then we describe the tasks that need to be performed.

2 Brief Introduction to Dimensionality Reduction Techniques

The goal of a dimensionality reduction technique is to transform the data from a higher dimensional space into a lower dimensional one, such that uninformative variance in the data is discarded, or such that a subspace in which the data lives is detected. Let $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_m}),\ \mathbf{x_i} \in \mathbb{R}^d$ be our dataset. The goal of a dimensionality reduction technique is to find a credible mapping of the m vectors to $\mathbb{R}^k, k \ll d$, maintaining as much as possible the variation/distances of the data. There are many reasons why dimensionality reduction is a useful tool in data mining and machine learning. For example, in many cases, not all the measured variables are important for understanding the underlying phenomena of interest.

As we will see in next lab, dimensionality reduction techniques differ from the so-called *feature selection* techniques, which retain a subset of the initial features of the dataset. On the other hand, dimensionality reduction techniques map the data into a new representation space, creating an alternative, smaller set of variables – defined as functions over all features – to represent the data. Additionally, in contrast to feature selection techniques, dimensionality reduction methods do not take into account the class labels associated with the instances of the data (e.g., labels that can be used in the classification task).

In this lab, we will focus on the following three dimensionality reduction methods:

- SVD (Singular Value Decomposition)
- PCA (Principal Component Analysis)
- MDS (Multidimensional Scaling).

Next, we will briefly present each one of these methods.

2.1 SVD (Singular Value Decomposition)

Singular Value Decomposition (SVD) is a matrix decomposition method that leads to a low-dimensional representation of a high-dimensional matrix. Let $\mathbf{A}$ be a $m \times n$ matrix. Then the Singular Value Decomposition of matrix $\mathbf{A}$ is defined as

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$$

where

- $\mathbf{U}$: $m \times m$ matrix has as columns the eigenvectors of $\mathbf{A}\mathbf{A}^\top$
- Σ : $m \times n$ is a diagonal matrix with the singular values of $\mathbf{A}$ in the diagonal (= square roots of $\mathbf{A}\mathbf{A}^\top$ eigenvalues)
- $\mathbf{V}$: $n \times n$ matrix has as columns the eigenvectors of $\mathbf{A}^\top\mathbf{A}$.

SVD allows an exact representation of any matrix, and also makes it easy to eliminate the less important parts of that representation in order to produce an approximate representation with any desired number of dimensions – leading to the concept of *low rank approximation* of a matrix. Let $\mathbf{A}$ be a $m \times n$ matrix, where $\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^\top$. Then, a r rank approximation of $\mathbf{A}$ is given by

$$\mathbf{Y} = \mathbf{U}_{m\times r}\text{diag}(\sigma_1, \ldots, \sigma_r){\mathbf{V}^\top}_{r\times n}.$$

How to perform dimensionality reduction with SVD?

The goal of dimensionality reduction is to find an approximation $\mathbf{Y}$ of $\mathbf{A}$ using r dimensions instead of the original n, $r < n$. This can be done using the properties of SVD: $\mathbf{Y}_{m\times n} = \mathbf{U}_{m\times r}\mathbf{\Sigma}_{r\times r}{\mathbf{V}^\top}_{r\times n}$. In practice, however, the purpose is not to actually reconstruct the original matrix, but to use the reduced dimensionality representation $\mathbf{U}_{m\times r}\mathbf{\Sigma}_{r\times r}{\mathbf{V}^\top}_{r\times n}$ in order to analyze the data.

The quality of low rank approximation $\mathbf{Y}$ of a matrix $\mathbf{A}$ can be evaluated using the *Frobenious norm* $\| \mathbf{A} - \mathbf{Y} \|_\mathbf{F}$, where

$$\| \mathbf{X} \|_\mathbf{F} = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}|x_{ij}|^2} = \sqrt{\sum_{i=1}^{\min(m,n)}\sigma_i^2}.$$

The best low rank approximation is the one that minimizes the Frobenius norm $\| \mathbf{A} - \mathbf{Y} \|_\mathbf{F}$.

Practical issues: How many singular values should we retain?

A useful rule of thumb is to retain enough singular values to make up 90% of the energy in $\mathbf{\Sigma}$. That is, the sum of the squares of the retained singular values should be at least 90% of the sum of the squares of all singular values.

2.1.1 Tasks to be Done in the Lab

Here, we will implement the SVD method in Python, and then we will apply it on real datasets. Fill in the Python script `my_svd.py` under the directory `Code/SVD`, in order to implement and apply the SVD decomposition on a matrix $\mathbf{X}$ that corresponds to an image. Then, you will need to reconstruct the original matrix (image) $\mathbf{X}$ using the top (largest) $k = \{10, 20, 50, 100, 200\}$ singular values. Compare (visually) these approximations to the original image. What is the error of each approximation? What do you observe? Hint: examine the distribution of the singular values of the original matrix $\mathbf{X}$.

2.2 Principal Component Analysis (PCA)

Principal components analysis (PCA) is a very popular technique for dimensionality reduction. Given a set of data on n dimensions, PCA aims to find a linear subspace of dimension d lower than n such that the data points lie mainly on this linear subspace. Such a reduced subspace attempts to maintain most of the variance of the data. The linear subspace can be specified by d orthogonal vectors that form a new coordinate system, called the *principal components*. The principal components are linear transformations of the original data points, so there can be no more than n of them. However, the hope is that only $d < n$ principal components are needed to approximate the space spanned by the n original axes. Thus, for a given set of data vectors $x_i, i \in 1 \dots t$, the d principal axes are those orthonormal axes onto which the variance retained under projection is maximal. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to (i.e., uncorrelated with) the preceding components.

Recall that the variance of a random variable is given by the following formula: $V(X) = \sigma^2 = E[(X - \mu)^2]$. PCA can be computed using the eigenvalue decomposition of the covariance matrix as follows:

1. Suppose that the data is organized into an $m \times n$ matrix $\mathbf{A}$, initially subtract mean values from columns: $\mathbf{C} = \mathbf{A} - \mathbf{M}$
2. Calculate the covariance matrix $\mathbf{W} = \mathbf{C}^\top \mathbf{C}$
3. Find eigenvalues and eigenvectors of the covariance matrix $\mathbf{W}$
4. *Principal Components*: the k eigenvectors $\mathbf{U}_{[1\dots k]}$ that correspond to the k largest eigenvalues
5. Project the data to the new space: $\mathbf{C}\mathbf{U}_{[1\dots k]}$

Can we compute the principal components using the SVD decomposition? Recall that the square root of the eigenvalues of the covariance matrix $\mathbf{C}^\top \mathbf{C}$ corresponds to the singular values of $\mathbf{C}$. That way, in step 2, we can compute the SVD decomposition of $\mathbf{C}$, and the principal components will correspond to the singular vectors that correspond to the k largest singular values.

As we have already discussed, PCA transforms the data into a lower dimensional space preserving the variance. The fraction of variance that is preserved in the transformed data can be captured by the following formula:

$$var = \frac{\sum_{i=0}^{k} \lambda_i}{\sum_{j=0}^{n} \lambda_j},$$

where n is the number of dimensions in the original dataset, k is the number of dimensions in the new representation space, and λ are the eigenvalues of the covariance matrix sorted in descending order.

2.2.1 Tasks to be Done in the Lab

Fill in the Python script `my_pca.py` under the directory `Code/PCA`, in order to implement the PCA technique. Then, PCA will be applied to a dataset in order to project the data into a k-dimensional space defined by the first k principal components. Here, we will analyze the *Wine* dataset, that corresponds to the chemical analysis of wines grown in the same region in Italy but derived from three different cultivars (three different types of wine; thus, three classes). This dataset is composed by 178

observations, describing the quantities of 13 constituents found in each of it. Thus, the number of dimensions of the dataset is 13. Visualize different combinations of features for the dataset. Then, project the data to the first 2 and first 3 principal components. What do you observe?

2.3 Multidimensional Scaling (MDS)

Multidimensional Scaling (MDS) is a dimensionality reduction technique used in information visualization, and in particular as a way to display the information contained in a distance matrix. The goal of the method is to place each object (point) in the N-dimensional space such that the between-object distances are preserved as well as possible.

Next we give the steps of the MDS algorithm:

1. Let $\mathbf{A}$ be the distance matrix. Construct the $N \times N$ squared distance matrix $\mathbf{D} = \mathbf{A}^2$
2. Compute the matrix $\mathbf{J} = \mathbf{I}_N - N^{-1}\mathbf{1}\mathbf{1}^T$ (where $\mathbf{I}_N$ the $N \times N$ identity matrix (one's in diagonal ans zeros elsewhere) and $\mathbf{1}\mathbf{1}^\top$ is the $N \times N$ matrix with one's at each cell)
3. Compute matrix $\mathbf{B} = -\frac{1}{2}\mathbf{J}\mathbf{D}\mathbf{J}$
4. Compute the SVD decomposition of matrix $\mathbf{B}$: $\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, and extract the k largest singular values $\mathbf{\Sigma}_k = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k)$ and the corresponding singular vectors $\mathbf{U}_k$
5. A k-dimensional spatial configuration of the N objects is derived by: $\mathbf{A}_k = \mathbf{U}_k\mathbf{\Sigma}_k^{1/2}$, where $\mathbf{\Sigma}_k$ the k largest singular values

The basic idea behind MDS is that the data points are moved in the space in such a way that the *stress criterion* is minimized. Stress is given by the following formula:

$$Stress = \sqrt{\frac{\sum_{\forall i,j}(d_{ij} - \delta_{ij})^2}{\sum_{\forall i,j}\delta_{ij}^2}}, \tag{1}$$

where d_{ij} the distance between points i and j after the trasformation and δ_{ij} their distance in the original space.

2.3.1 Tasks to be Done in the Lab

Fill in the Python script `my_mds.py` under the directory `Code/MDS`, in order to implement the MDS technique. Then, MDA will be applied to compute the position of 10 US cities in the 2-dimensional space, using the 10×10 distance matrix $\mathbf{D}$. What do you observe? Does the relative position of the cities has been preserved?

References

[1] Smith, Lindsay I. "A Tutorial on Principal Components Analysis." Cornell University, USA 51, 2002.

[2] Shlens, Jonathon. "A Tutorial on Principal Component Analysis." arXiv preprint arXiv:1404.1100, 2014.

[3] Bishop, Christopher M. "Pattern Recognition and Machine Learning." Vol. 1. New York: Springer, 2006.

[4] Jure Leskovec, Anand Rajaraman, Jeff Ullman. ”Mining of Massive Datasets”, Cambridge University Press, 2014.

[5] Wall, Michael E., Andreas Rechtsteiner, and Luis M. Rocha. ”Singular Value Decomposition and Principal Component Analysis: A Practical Approach to Microarray Data Analysis.” Springer US, 2003.