15. PRINCIPAL COMPONENT ANALYSIS (PCA)

In unsupervised machine learning the data instances are given with their attributes and there is no target variable. The main goal is discover interesting patterns such as if there is an informative way to visualize the data or if we can discover subgroups among the attributes or among the data observations. The main unsupervised learning methods are:

- *Principal Component Analysis* – dimensionality reduction
- *Clustering* – grouping data so that data observations within each group are similar to each other and data observations from different groups are different from each other
- *Association Rule Mining* – discovering the rules that determine how or why certain variables are connected

PCA (Pearson, 1901) is a data driven technique that produces a new coordinate system that reveals interesting structure/pattern in the data. The new coordinates are uncorrelated and the dimension of the new coordinate system is reduced appropriately. In other words, PCA identifies the subspace where data approximately lies. It is used for:

- *exploratory data analysis* – for visualizing high-dimensional data sets
- *pre-processing before running a supervised ML model* – for speeding up training and reducing computational cost, removing noise and unnecessary and redundant details in the data, and reducing the complexity of the hypothesis class considered

The main idea is to project data points onto only the first few principal components to obtain lower-dimensional data while preserving as much of the data's variation as possible. In other words, using PCA we remove the redundant and highly-correlated data and we keep only the most significant information for further analysis.

The first principal component is defined as a direction that maximizes variance of the projected data, the second principal component is a direction orthogonal to the first principal component that is the next one to maximize the variance, etc. It can be proved that the principal components are the eigenvectors of the covariance matrix and are computed either by Eigendecomposition (ED) of the covariance matrix or by the Singular Value Decomposition (SVD) of the data matrix (called "design matrix"). See APPENDIX.

Assume we have data consisting of d features (such as house price, age, size, number of bedrooms, etc.) and n data observations (or data points, or samples). We form the $d \times n$ "feature – observation" matrix where variables are listed in the rows and observations in the columns. Some authors and Python prefer "observation – feature" matrix instead of "features – observations" matrix.

The goal of PCA is to reduce the data dimensionality by extracting only 2 or 3 important linear combinations of the features. Hence, we obtain a lower-dimensional coordinate system where we visualize our projected data observations. The mathematics theory behind and steps in PCA are:

1. *Standardize (center and scale) the data*

   To center the data, i.e., to move all data points so that the center is the origin, we find the average of each row (feature) and replace each value x by

$$x - \text{mean}.$$

Since in reality different feature have different scale (for example, house price vs. house age) and to ensure that PCA is not picking up wrong directions in describing data variation, we also divide by standard deviation. In other words, we scale each feature by finding the z-scores

$$z = \frac{x - \text{mean}}{\text{standard deviation}}$$

Finally, we form the $d \times n$ matrix A consisting of scaled values.

2. *Compute the covariance or correlation matrix*

   $$S = \frac{1}{n-1} AA^T$$

   If we are working with only centered data, the above matrix is the covariance matrix, and if we are working with scaled data, then S is the correlation matrix. The entries on the diagonal are the variances (or correlations) for each variable and the off-diagonal entries are the covariances (or correlations) between two variables: positive covariance indicates that the variables are directly related (when one increases, the other increases as well), negative covariance indicates inverse relationship (when one increases, the other decreases). This matrix is symmetric of size $d \times d$, so its columns are of the same size as the columns of A.

3. *Find the eigenvalues and the orthonormal eigenvectors of S*

   Recall that the eigendecomposition of AA^T is equivalent to the singular value decomposition of A. Therefore, the left-singular vectors of A are eigenvectors of S. Also, if σ_i is a singular value of A, then $\sigma_i^2/(n-1)$ is the eigenvalue of S.

4. *Find the principal components*

   We arrange the eigenvalues found in the previous step in the decreasing order. The first principal component PC_1 is in the direction of the 1st eigenvector, the second principal component PC_2 is in the direction of the 2nd eigenvector, etc. The entries of each PC_i are called *loadings* and they tell us how the PC_i is a linear combination of the original features.

5. *Reduce the dimension of the data*

   We project data points (i.e., columns of A) onto the selected principal components (i.e., several eigenvectors of S). By the Eckart-Young theorem we know that the line closest to the data points is in the direction of PC_1, etc ("closest" is in the sense of perpendicular least squares).

   In addition, the total variance, which is the trace of S, is

   $$T = \text{trace}(S) = \frac{\sigma_1^2 + \dots \sigma_d^2}{n-1}$$

   The ith principal component PC_i explains

   $$\frac{\sigma_i^2/(n-1)}{T} = \frac{\sigma_i^2}{\sigma_1^2 + \dots \sigma_d^2}$$

   of the total variation. We use a *scree plot* to graph the percentages of variation that each PC_i accounts for. Also, the sum of squared distances from the points projected to PC_i to the origin is the eigenvalue for PC_i or the squared singular value σ_i^2.

   To determine how many principal components we should use we can look at the scree plot and apply the elbow method or, in case when we will continue with supervised ML, we can use cross validation while running the ML algorithm.

Remark:

- Another way of finding the principal components (i.e., the directions in which the variance of the projected data observations is maximal) is by considering an optimization problem.

  Consider the data set $\{x^{(1)}, \dots, x^{(n)}\}$, which is centered or scaled. To find the first principal component, we need to find a unit vector u that maximizes variance of the projected data observations projected to u. Recall that the projection of $x^{(i)}$ to vector u is the dot product $\left(x^{(i)}\right)^T u$. Therefore, the variance of projected data observations onto direction u is

$$\begin{aligned}\frac{1}{n-1}\sum_{i=1}^{n}\left(\left(x^{(i)}\right)^T u\right)^2 &= \frac{1}{n-1}\sum_{i=1}^{n} u^T x^{(i)} \left(x^{(i)}\right)^T u \\ &= u^T\left(\frac{1}{n-1}\sum_{i=1}^{n} x^{(i)}\left(x^{(i)}\right)^T\right) u \\ &= u^T S u \\ &= \lambda\end{aligned}$$

  where λ is the eigenvalue of the covariance (or correlation) matrix S corresponding to the eigenvector u. The last line follows from the fact that u is unit, and from $Su = \lambda u$, we have $u^T S u = \lambda$.

  Hence, the first principal component is given by u that maximizes the above expression. That direction is the eigenvector of S corresponding to the highest eigenvalue.

- Let $u_1, \dots, u_k$ be the first k principal components. To view the data observations $x^{(i)}$ in the lower dimensional space, we compute their projections

$$y^{(i)} = \begin{bmatrix} u_1^T \cdot x^{(i)} \\ \dots \\ \dots \\ u_k^T \cdot x^{(i)} \end{bmatrix}$$

APPENDIX

Eigenvalues and Eigenvectors

Let A be an $n \times n$ matrix. Then $\vec{v} \in \mathbb{R}^n$ is an *eigenvector* of A corresponding to an *eigenvalue* λ if

$$A\vec{v} = \lambda\vec{v}$$

and $\vec{v} \neq \vec{0}$. Note that eigenvectors are vectors that are just scaled by the linear transformation $\vec{x} \mapsto A\vec{x}$ and the amount of that scaling is the eigenvalue. The above equation implies

$$(A - \lambda I)\vec{v} = \vec{0},$$

and since $\vec{v} \neq 0$, this means that the matrix $A - \lambda I$ is not invertible, i.e., its determinant is 0. The equation

$$|A - \lambda I| = 0$$

is called the *characteristic equation* and the polynomial $p(\lambda) = |A - \lambda I|$ is called the *characteristic polynomial* of A. Since the polynomial $p(\lambda)$ is of degree n, the equation $p(\lambda) = 0$ has k distinct solutions ($k \leq n$). The set of those solutions $\lambda_1, \ldots, \lambda_k$ is called the *spectrum* of A. Therefore, we can write

$$p(\lambda) = (\lambda - \lambda_1)^{n_1} \ldots (\lambda - \lambda_k)^{n_k}, \qquad n_1 + \ldots + n_k = n.$$

Here, n_i is called the *algebraic multiplicity* of λ_i.

For each eigenvalue λ_i, we have the specific eigenvalue equation

$$(A - \lambda_i)\vec{v} = \vec{0}.$$

There are m_i linearly independent solutions to this equation ($m_i \leq n_i$) and m_i is called the *geometric multiplicity* of λ_i.

Example: Find the eigenvalues and eigenvectors of

$$A = \begin{bmatrix} 8 & 3 \\ 2 & 7 \end{bmatrix}$$

Solution:

$$|A - \lambda I| = \begin{vmatrix} 8-\lambda & 3 \\ 2 & 7-\lambda \end{vmatrix} = (\lambda - 10)(\lambda - 5)$$

implying that the eigenvalues are $\lambda_1 = 10$ and $\lambda_2 = 5$. Next we find an eigenvector for each of those two eigenvalues.

- By plugging eigenvalue $\lambda_1 = 10$ in $(A - \lambda I)\vec{v} = \vec{0}$, we get

  $$\begin{bmatrix} -2 & 3 \\ 2 & -3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

  We have only one equation

  $$-2v_1 + 3v_2 = 0.$$

  Therefore there is one free variable (let's say it is v_2) and

  $$v_1 = \frac{3}{2}v_2.$$

Therefore the eigenspace corresponding to $\lambda_1 = 10$ is given by

$$\vec{v} = \begin{bmatrix} v_1 \\ v_1 \end{bmatrix} = \begin{bmatrix} \frac{3}{2}v_2 \\ v_2 \end{bmatrix} = v_2 \begin{bmatrix} \frac{3}{2} \\ 1 \end{bmatrix}$$

for any nonzero $v_2 \in R$. By choosing, for example $v_2 = 2$, we get the eigenvector

$$\vec{v} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

- Similarly, for $\lambda_2 = 5$, we get

$$\begin{bmatrix} 3 & 3 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

  which gives only one equation

$$w_1 + w_2 = 0$$

  We can assume that w_1 is a free variable and then $w_2 = -w_1$. Hence the eigenspace is given by

$$\vec{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_1 \\ -w_1 \end{bmatrix} = w_1 \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

  where $w_1 \in R \setminus \{0\}$. If we take $w_1 = 1$, we get the eigenvector

$$\vec{w} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

□

Note that in the previous example, the eigenvectors $\vec{v}$ and $\vec{w}$ are linearly independent. If fact, one can prove if the eigenvectors correspond to different eigenvalues, then those eigenvectors are linearly independent. When an λ eigenvalue is of multiplicity $1 < k \leq n$, sometimes it might be possible to find k linearly independent vectors corresponding to λ and sometimes it is not possible.

Eigendecomposition

Let A be an $n \times n$ matrix with n linearly independent eigenvectors $\vec{v}_1, \ldots, \vec{v}_n$ corresponding to eigenvalues $\lambda_1, \ldots, \lambda_n$. Note that eigenvalues do not need to be distinct. Then if Q is a matrix whose ith column is the eigenvector $\vec{v}_i$ and Λ is a diagonal matrix whose diagonal elements are λ_i, from $A\vec{v} = \lambda\vec{v}$, we have

$$AQ = Q\Lambda.$$

Since the columns of Q are linearly independent, Q is invertible and we get

$$\boxed{A = Q\Lambda Q^{-1}}$$

This factorization of A is called the *eigendecomposition* of A.

Next we state some important results and remarks related to eigenvalues and eigenvectors.

- The sum of the eigenvalues of A is equal to the sum trace diagonal entries of A (which is called the *trace*):

$$\text{trace}(A) = a_{11} + a_{22} + \ldots + a_{nn} = \lambda_1 + \lambda_2 + \ldots + \lambda_n$$

- If A is a real symmetric matrix, then it can be proved that its eigenvalues are real and the eigenvectors can be chosen to be orthogonal and unit. Such vectors are called *orthonormal* vectors. Thus, Q is an *orthonormal matrix* and satisfies

  $$Q^TQ = QQ^T = I$$

  which implies that $Q^{-1} = Q^T$. Hence, A can be decomposed as

  $$\boxed{A = Q\Lambda Q^T}$$

  This result is known as *the Spectral Theorem.*

Singular Value Decomposition

In the previous section we studied that if $A \in \mathbb{R}^{n\times n}$ then we have

$$AQ = Q\Lambda$$

where Λ is a diagonal matrix with eigenvalues of A on the main diagonal and Q is a matrix whose columns are the corresponding eigenvectors of A. If A has n linearly independent eigenvectors, then Q will be invertible and, therefore, A will have the eigendecompositon

$$A = Q\Lambda Q^{-1}.$$

If A is also symmetric, then we can choose its eigenvectors to be orthonormal so $Q^{-1} = Q^T$ and we have

$$A = Q\Lambda Q^T.$$

Since in real life we often work with matrices that are not square, in this section we generalize the above decomposition to rectangular matrices. We will show that we can decompose an arbitrary matrix $A \in \mathbb{R}^{m\times n}$ as

$$\boxed{A = U\Sigma V^T} \tag{1}$$

where the $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are orthonormal matrices and $\Sigma \in \mathbb{R}^{m\times n}$ has the only nonzero values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$ on the diagonal (here, $r \leq \min\{m, n\}$).

The columns of U are called left singular vectors of A and the columns of V are called the right singular vectors of A. The entries σ_i of matrix Σ are called singialr values of A.

Idea:

- Given an arbitrary rectangular matrix A, the main idea is to note that A^TA is a square matrix which is also symmetric. Hence, it has real eignevalues and the eigenvectors can be chosen to be orthonormal. Therefore, by the spectral theorem, we have

  $$A^TA = V\Lambda V^T, \tag{2}$$

  where V is a matrix whose columns are orthonormal eigenvectors of A^TA and Λ is a diagonal matrix whose diagonal elements are the eigenvalues of A^TA. In addition, there is a theorem that shows that the eigenvalues of A^TA are non-negative. On the other hand, consider AA^T which is also a square matrix with the same nonzero eigenvalues and we can decompose it as

$$AA^T = U\Lambda U^T,$$

where U is a matrix whose columns are orthonormal eigenvectors of AA^T.

- To show that the decomposition (1) is possible, we want to find orthonormal vectors $\vec{v}_i$, orthonormal vectors $\vec{u}_i$, and nonnegative scalars σ_i such that

  $$A\vec{v}_1 = \sigma_1\vec{u}_1 \quad A\vec{v}_2 = \sigma_2\vec{u}_2 \quad \dots \quad A\vec{v}_r = \sigma_r\vec{u}_r \quad A\vec{v}_{r+1} = \vec{0} \quad \dots \quad A\vec{v}_n = \vec{0},$$

  or, in other words,

  $$AV = U\Sigma \quad \text{or} \quad A = U\Sigma V^T.$$

  (Note that the dimension of the null-space for A is $n - r$ and that r is the rank of A.)

  How do we find U, V and Σ? Consider

  $$A^TA = V\Sigma^TU^TU\Sigma V^T = V(\Sigma^T\Sigma)V^T$$

  since U consists of orthonormal vectors. We note from (2) that V is the matrix of eigenvectors for A^TA and the entries of $\Sigma^T\Sigma$ (denoted by σ^2) are the eigenvalues of A^TA. Similarly,

  $$AA^T = U\Sigma V^TV\Sigma^TU^T = U(\Sigma\Sigma^T)U^T$$

  implies that U consists of eigenvectors of AA^T.

Strategy: Given a matrix A, to compute V, Σ and U, we

1. find eigenvectors and eigenvalues λ_i of A^TA
2. define $\sigma_i = \sqrt{\lambda_i}$ and form matrix Σ so that its diagonal entries σ_i are placed in the decreasing order
3. choose orthogonal eigenvectors from the first step and normalize them to get $\vec{v}_i$'s; create V as a matrix whose columns are $\vec{v}_i$'s in the same order as Σ was created
4. define $\vec{u}_i = \frac{1}{\sigma_i}A\vec{v}_i$ and normalize them if needed; create U as a matrix whose columns are $\vec{u}_i$'s

Remark:

- Note that

  $$A\vec{x} = U\Sigma V^T\vec{x}$$

  geometrically implies that $\vec{x}$ is rotated to $V^T\vec{x}$, stretched to $\Sigma(V^T\vec{x})$, and then rotated to $U(\Sigma V^T\vec{x})$, so any linear transformation $\vec{x} \mapsto A\vec{x}$ can be decomposed as a rotation, stretching, and another rotation.

- Note that $\underline{\text{eigendecomposition of } AA^T \text{ is equivalent to singular value decomposition of } A}$. First, AA^T is a square symmetric matrix and assuming $A = U\Sigma V^T$ we have

  $$AA^T = (U\Sigma V^T)(U\Sigma V^T)^T = U\Sigma V^TV\Sigma^TU^T = U\Sigma\Sigma^TU^T$$

  In fact, the eigenvalues of AA^T are the singular values of A.

- The SVD has the "full" and the "reduced" form. Using only the nonzero singular values and keeping the columns and rows of U and V^T, respectively, that matter in the multiplication, we can write SVD in the reduced form

$$A = U_r\,\Sigma_r\,V_r^T = \sigma_1 \vec{u}_1 \vec{v}_1^T + \ldots + \sigma_r \vec{u}_r \vec{v}_r^T$$

where $U_r \in \mathbb{R}^{m\times r}, \Sigma_r \in \mathbb{R}^{r\times r}$ and $V_r \in \mathbb{R}^{n\times r}$. In this notation Σ_r is square, while U_r and V_r are rectangular matrices satisfying $U_r^T U_r = I$ and $V_r^T V_r = I$, however we do not necessarily have $U_r U_r^T = I$ and $V_r V_r^T = I$.

- The rank of a matrix is the number of its nonzero singular values. Note that the SVD separates the matrix A of rank r into rank 1 matrices and these rank 1 matrices come in order of importance. In fact, the matrix A_k defined by

$$A_k = \sigma_1 \vec{u}_1 \vec{v}_1^T + \ldots + \sigma_k \vec{u}_k \vec{v}_k^T$$

is the best rank k approximation to A. More precisely, the Eckhart-Young Theorem claims that if B is any matrix of rank k, then

$$\|A - A_k\| \leq \|A - B\|,$$

for any of the following norms:

1. spectral or l^2 norm
$$\|M\|_2 = \max \frac{\|Mx\|}{\|x\|} = \sigma_1$$
2. Frobenius norm
$$\|M\|_F = \sqrt{\sigma_1^2 + \ldots + \sigma_r^2}$$
3. nuclear norm
$$\|M\|_N = \sigma_1 + \ldots + \sigma_r$$

Python codes:

Lecture_15_PCA_Iris.ipynb
Lecture_15_PCA_BreastCancerData.ipynb

References and Reading Material:

[1] Youtube lecture by Professor Strang (MIT) on SVD
https://www.youtube.com/watch?v=rYz83XPxiZo

[2] *An Introduction to Statistical Learning*, James et al., Sections 10.1 and 10.2

[3] Andrew Ng's (Stanford) notes on PCA

[4] *Hands – On Machine Learning with Scikit-Learn, Keras & TensorFlow*, by Geron, pp. 213-215 and pp. 219-229