Application of PCA on Breast cancer dataset and MDS on pairwise distances of 6 Italian cities.

Karan Kabbur Hanumanthappa Manjunatha

December 7, 2022

Abstract

Principal Component Analysis(PCA) is technique that aims to find a linear transformation which when applied to high dimensional data such as Wisconsin Diagnostic Breast Cancer data, gives us low dimensional representation of the original data which captures most of the variance of the data(or in other words minimizes the reconstruction error). But if the data is not provided and instead pairwise proximity/distances between objects are given, which in our case distances between 6 Italian cities, we apply Multidimensional scaling(MDS) instead and obtain 2D xy-coordinates (which is of lower dimensions). In general, the goal of MDS is to place observations in a space based on similarity scores between them.

1 Introduction

Principal Component Analysis (PCA) is a dimensionality reduction technique invented by Karl Pearson in 1901[7], which is used for identification of a smaller number of uncorrelated variables known as Principal Components from a larger set of data. It is fundamental algorithm for the dimensionalty reduction of high dimesional dataset. It has wide range of applications in neuroscience[4], quantitative finance[6] and very popular application being seen in Facial recognition[2]. The dimensionality reduction process is of reducing the number of variables in a dataset while preserving as much variation as possible. This is done by projecting each data point onto only the first few principal components. That these components have two major properties:

1. They aim to capture the most amount of variability in the original dataset.
2. They are orthogonal to (independent of) one another.

We next move to Multidimensional scaling (MDS). It is a technique which measures similarities, dissimlarities or distances between the pairs of objects in general and then visualizes it by representing the information in a spatial configuration.[5]. For the specific case when given a distance matrix between the objects, the MDS algorithm places each object into a a lower-dimensional space such that the between-object distances are preserved as much as possible and it can be visualized as 2D scatter plot.[1]

Even though the underlying concept that both PCA and MDS are used for dimensionality reduction. However, there are some notable differences between them. MDS starts with the similarities, dissimilarities, or distances, while PCA starts with a feature representation. Thus, for PCA, we already have the data in a high-dimensional space, and we map it into a lower dimensional space. However, for MDS, we do not have the data in higher dimensional space but just a measure of similarity, and we would like to induce the space from that.

Problem description: For this report we were asked to apply PCA and MDS on real world datasets. For the part of PCA, I have considered *Wisconsin Diagnostic Breast Cancer (WDBC) dataset*(see dataset description part of results) which has 30 features related to various aspects of the breast tumor. The diagnostic information that the cell nucleus can be either Malignant(M) or Benign(B) were also informed. Since the number of features are quite large, there may be few features which are not typically noteworthy and can be neglected. Thus, applying PCA using the steps presented in the *Methods* section, the data is projected to a low dimensional representation in such a way that we capture maximum amount of variance of the original dataset of 30 features. Next, I move on to apply MDS on a city distance matrix which contains the information of the distances between each pair of cities of Italy. After applying MDS, and keeping the number of dimension as 2, we obtain MDS xy coordinates. However, only the distance between the cities are preserved in the spatial representation(scatter plot) of the obtained resulting coordinates. Using rotation operator and rescaling, a further analysis and research is done to provide exact directional orientation of cities in accordance with the exact Italian map found everywhere.

2 Methods

2.1 Principal component analysis (PCA)

The algorithm of PCA is as follows[8]:

1. **Centering and standardizing the data**: Standardizing refers to subtracting the mean of the each variable from each data point and dividing by its standard deviation. This is necessary because PCA is a variance-maximizing procedure, and centering the data ensures that the first principal component explains the maximum possible variance.

$$x_s = \frac{x - \mu}{\sigma} \tag{1}$$

where x is the variable/feature columnn of the dataset X of dimension n data points $\times$ m features. And μ, σ are mean and standard deviation given by the equation:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{2}$$

$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2} \tag{3}$$

2. **Compute the covariance matrix**: Given our standardized data matrix X_s, covariance matrix of dimension $m \times m$ is:

$$\Sigma = \frac{1}{n-1} X_s^T X_s \tag{4}$$

3. **Compute the eigenvalues and eigenvectors** of the covariance matrix, Σ.
4. **Sorting the eigenvalues and choosing the eigenvectors**: the eigenvector which maximizes the variance is the one with the largest eigenvector, and the k eigenvectors which maximize the variance are the ones with the k largest eigenvalues, so we proceed to sorting our eigenvalues and select the corresponding eigenvectors.
5. **Projecting the data**: Once we have our transformation C, we only need to multiply it with standardized data X_s: which gives us the lower dimensional representation of our data. A common way to show PCA-reduced data is by plotting the first component against the second.

$$X_p = XC \tag{5}$$

```

# 1. Standardize(which also centers) the data
import numpy as np
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X)
X_std = scaler.transform(X)

# 2. Compute the covariance matrix
cov_mat = (X_std).T.dot(X_std) / (X_std.shape[0]-1)
#cov_mat = np.cov(X_std, rowvar=False)

# 3. Compute eigenvectors and values of the cov. matrix
eig_vals, eig_vecs = np.linalg.eig(cov_mat)

# 4. Sort and choose the first 3 largest eigenvectors
srt = np.argsort(eig_vals)[::-1]
eig_vecs_srtd = eig_vecs[:, srt]
n_components = 3
C = eig_vecs_srtd[:, 0:n_components]

# 5. Project the data
Xp = np.matmul(X_std, C)
```

Listing 1. Python code depicting the steps explained for PCA given dataset of features, X (n data points x m features)

2.2 Multi-dimensional scaling, (MDS)

The procedure to implement MDS is as follows[10][1]: Given a distance matrix D which is symmetric, where D_{ab} refers to distance between city a and city b, the inverse problem is solve and obtain the position coordinates of the cities or in other words to recover the map.

1. Square the entries of the distance matrix, $D^{(2)} = [d^2]$.
2. Given n which is the number of cities, apply the double centering to squared distance matrix D^2.

$$B = -\frac{1}{2} H D^2 H^T \tag{6}$$

   with matrix $H = I - \frac{1}{n} 1.1^T$, where $1 = [1, 1, ...1]^T$ is a column vector of $1's$ whose dimension is $n \times 1$.
3. Diagonalize the matrix B and extract the corresponding eigenvalues and eigen vectors.
4. Sort the 2 largest eigen values $\lambda_1 > \lambda_2$ and the corresponding eigen vectors in $\mathbf{e}_1, \mathbf{e}_2$.
5. A 2-dimensional spatial configuration of the n cities is derived from the coordinate matrix

$$\mathbf{X} = [\mathbf{e}_1, \mathbf{e}_2] \begin{bmatrix} \sqrt{\lambda_1} & 0 \\ 0 & \sqrt{\lambda_2} \end{bmatrix} \tag{7}$$

   The xy coordinates obtained does not have any sense of direction i.e. North-South, East-West cannot be directly inferred without the rotation of the coordinates.

```
def apply_MDS(D, k=2):
    # Inputs:
    # 1. D = Distance matrix between the cities
    # 2. k=2, dimensions of the coordinates

    # dimension of the distance matrix
    n = D.shape[0]

    # Identity matrix of dim = n
    I = np.eye(n)

    # column vector of ones of dim = (n,1)
    one = np.ones(n).reshape(-1,1)

    # operator
    H = I - (1./n)*np.dot(one, one.T)

    # Double centering the squared distance matrix
    B = -0.5*np.matmul(H, np.matmul(D**2, H.T))

    # find evals, and eig vectors
    eig_vals, eig_vecs = np.linalg.eig(B)
    srt = np.argsort(eig_vals)[::-1] # indices of eigenvalues from largest to smallest
    eig_vals_srtd = eig_vals[srt] # sorted eigenvalues
    eig_vecs_srtd = eig_vecs[:, srt] # sorted eigenvectors

    # select first k(=2) largest evals and evectors
    Eval_2 = eig_vals_srtd[:k]
    Evec_2 = eig_vecs_srtd[:,:k]

    # coordinates of dimension, n x k
    X = np.matmul(Evec_2, np.diag(np.sqrt(Eval_2)))
    return X
```

Listing 2. Python code depicting the steps explained for MDS given dataset of distance matrix of cities D, dim = (n $\times n$)

Recovering the directions of the cities

1. With known longitudes and latitudes of the n cities, $V = [\mathbf{v}_{long}, \mathbf{v}_{lat}]$, the measurements mean and stdev. of latitudes and longitudes of n cities $\mu_{long}, \mu_{lat}, \sigma_{long}, \sigma_{lat}$ were found. They were standardized using *StandardScaler()* function of *sklearn.preprocessing* module of python and let's call it V_{std}.
2. The two dimensional rotation matrix which rotates points in the xy plane anti-clockwise through an angle θ about the origin is given by:

$$\mathbf{R} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \tag{8}$$

3. Find the value of θ' for which the distance between the standardized longitude-latitude and standardized MDS coordinates and averaged over all the n cities, is minimum.

$$\underset{\theta}{argmin} \frac{1}{n} \sum_{i=1}^{n} ||X_{i,std}(\theta) - V_{i,std}||_2 \tag{9}$$

where $i = 1, ..., n$ is the row vector of the corresponding matrices. $X_{std}(\theta)$ is obtained by first rotating the MDS coordinates by θ, i.e. $X(\theta) = XR(\theta)$ and finally standardizing $X(\theta)$ with mean = 0 and stdev. = 1 along each coordinates/columns.

$$\mathbf{X}_{std}(\theta) = [\frac{\mathbf{x}(\theta) - \mu_{x(\theta)}}{\sigma_{x(\theta)}}, \frac{\mathbf{y}(\theta) - \mu_{y(\theta)}}{\sigma_{y(\theta)}}] \tag{10}$$

4. The standardised MDS xy coordinates for the optimal value θ' given in Eq. 10 is

$$\mathbf{X}_{std}(\theta') = [\mathbf{x}', \mathbf{y}'] \tag{11}$$

It is then rescaled and transformed in terms of latitude and longitude as follows:

$$\mathbf{X}_{rescale} = [\mathbf{x}' * \sigma_{long} + \mu_{long}, \mathbf{y}' * \sigma_{lat} + \mu_{lat}] \tag{12}$$

5. The original latitude, longitude of n cities given by $\mathbf{V}$ along with the MDS coordinates given by Eq. 12 were visualized on a map.

3 Results

3.1 Results of PCA

PCA dataset : For applying the PCA, I have considered the *Wisconsin Diagnostic Breast Cancer (WDBC) dataset* available for common use online[3] as well as *sklearn.datasets* module of python. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The attribute information consists of ID number and the Diagnosis i.e. whether the type of cancer is malignant(M) or benign(B). There are ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness ($perimeter^2/area - 1.0$)
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. All feature values are recoded with four significant digits. There were no missing attribute values. The class distribution is: 357 benign, 212 malignant.

PCA results: Given the dataset X of dimensions n data points and m features, it was standardised and covariance of it was computed as given in Eq. 4 and is visualized as 2d heatmap in Fig. 1a.

The proportion of the variance that each eigenvector represents is calculated by dividing the eigenvalue corresponding to that eigenvector by the sum of all eigenvalues. Fig. 1b visualizes variance explained by each component. From this we can understand how many components it is better to keep based on how much information was being retained. From the same figure, we look for famous kink or elbow in the *scree plot* of the variance, and pick the number of components at which the kink occurs, the rationale being that most of the variance will be accounted for by the components before the kink. In our case, we see that just the first 3 principal components explain ~ 70% of the total variance.

A common way to show PCA-reduced data is by plotting the first component against the second as shown in the Fig. 1c. We observe that the data projected in the space of the first two components shows two clusters(i.e. malignant and benign) which are quite well separated from each other and so just the from the first two principal components we can observe the differences in the 2 types of cancer cells. We investigate further on the first 3 PCA components we have found.

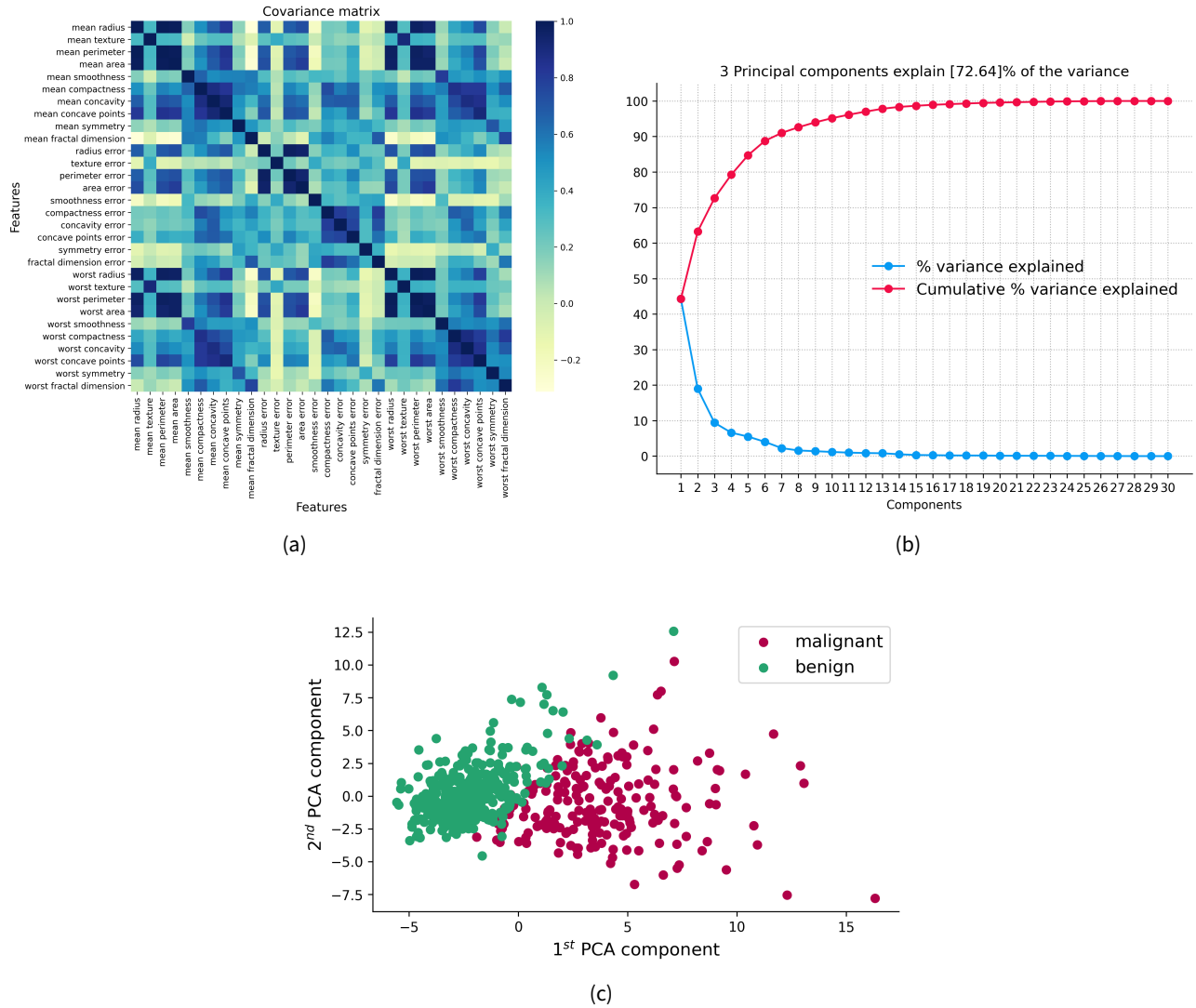


Figure 1. **(a)** Covariance plot of the breast cancer dataset where we observe correlations in some regions and less correlations in other parts of the features. **(b)** Scree plot of the variance with kink or elbow observed when number of PCA components oberved is 3 for which a total of 72% of variance is explained by the first 3 PCA components. **(c)** Plot of first two projected PCA components data.

In the Fig. 2, we visualize the feature weights of the first 3 PCA components. From the figure, let us consider Dimension 1 whose weights are a linear combination of the 30 features which identifies the direction of largest variance in feature

space. We observe that all the feature weights of Dimension 1 are positive in contrast to feature weights from Dimension 2 and Dimension 3. Note that the first 2 components themselves explain ~ 63% of the variance. In the Dimension 1, we notice that *texture error* and *smoothness error* having low feature weights compared to other features. This means that, when projecting the data on the first PCA component, PCA mostly looks at all the other features but and it almost doesn't consider texture error and smoothness error. However, the same is not true in the case of Dimension 2 for those 2 features. In the dimension 2 we observe that fractal dimension, its "mean" as well as "worst" has high contribution to the weight. In the Dimension 3, it should be noted that only the main 10 real values features(with the keywords "mean", "worst") contributes positively to the weights and has enough to the variance compared to the other features.

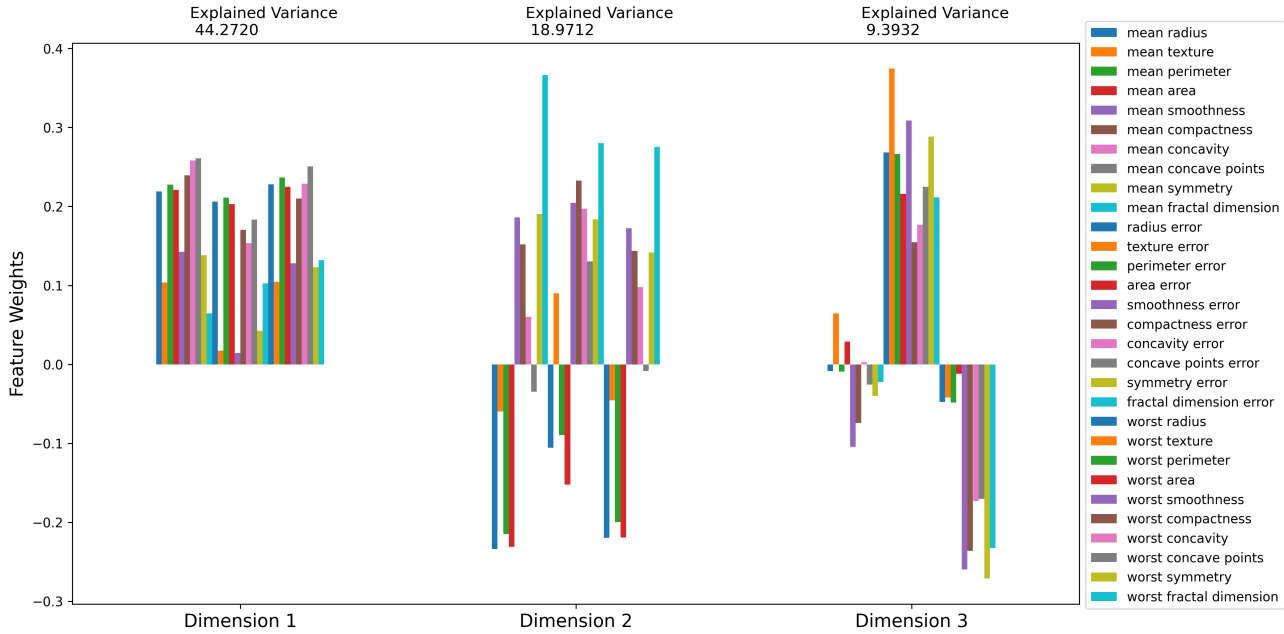


Figure 2. Visualizing feature weights as bar plots for the first 3 PCA components.

3.2 Results of MDS

MDS dataset : For the report, I have considered the longitudes and latitudes of 5 Italian cities, namely *Bari(BA), Firenze(FI), Milan(MI), Napoli(NA), Rome(RM), Torino(TO)*.

cities	Longitude	Latitude
BA	16.869820	41.12066
FI	11.255814	43.76956
MI	9.189510	45.46427
NA	14.268110	40.85216
RM	12.511330	41.89193
TO	7.686820	45.07049

Table 1. Longitudes and latitudes of 6 Italian cities

I have constructed a distance matrix(see Fig. 3a) among these 6 Italian cities using latitudes and longitudes information available online[9](see Table 1). The latitude and longitudes were transformed into xyz coordinates using the transformation

rule:

$$x = Rcos(lat_{rad})cos(long_{rad}) \tag{13}$$

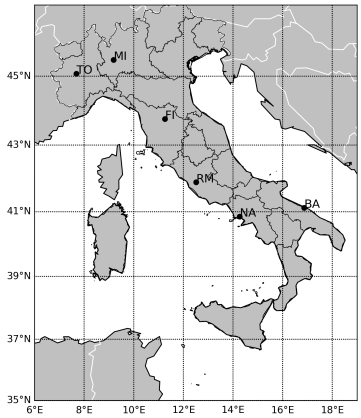
$$y = Rcos(lat_{rad})sin(long_{rad}) \tag{14}$$

$$z = Rsin(lat_{rad}) \tag{15}$$

where the latitudes and longitudes of the cities are converted to radians before applying cosine and sine functions. The radius R was considered as radius of the earth 6371km. The library sklearn of python includes a function which takes a list of xyz-coordinates combinations in radians and creates a matrix of one list by the other of the distance between these points. The distance computed here is a euclidean distance using the function *sklearn.neighbors.DistanceMetric.get_metric('euclidean')*.

cities	BA	FI	MI	NA	RM	TO
cities						
BA	0.000000	546.406592	786.086353	220.391980	372.835434	863.917119
FI	546.406592	0.000000	249.483717	408.019980	232.514616	318.149444
MI	786.086353	249.483717	0.000000	657.217264	478.485627	125.481511
NA	220.391980	408.019980	657.217264	0.000000	186.686316	711.115004
RM	372.835434	232.514616	478.485627	186.686316	0.000000	525.449558
TO	863.917119	318.149444	125.481511	711.115004	525.449558	0.000000

(a)



(b)

Figure 3. (a). Euclidean Distance matrix between 6 Italian cities computed from latitude and longitudes as seen in Table 1 **(b).** Italian map with 6 Italian cities located and annotated.

Using *mpltoolkits.basemap* module of python and the given latitudes and longitudes as shown in Table 1, a map of Italy with the locations of the 6 Italian cities was visualized as shown in Fig.3b. The geographic (GIS) data corresponding to Italy is obtained from *https://www.diva-gis.org/gdata* and is imported into Python for the visualization.

MDS results By considering the distance matrix of the 6 Italian cities as given in Fig. 3a, we applied MDS as given the code listing 2. The dimension 1(as (-x) coordinate, where values are flipped or multiplied by -1) and dimension 2(as y coordinate) are plotted in Fig. 4a. The optimal rotation angle θ' for which we recover the directional sense of the cities is given by Eq. 9 and is found to be $\theta' = 39$ as shown in the Fig. 4b. Applying this optimal rotation angle(θ') on Fig. 4a by using the Eq. 11, we obtain rotated and standardised MDS coordinates which is shown in Fig. 4c and we observe that when we compare it to the original map of Italy with annotated cities(see Fig. 3b), the directional sense is well preserved. The rescaled MDS coordinates of the cities as given by Eq. 12 is transformed into longitude and latitudes and is represented on the Italian map in red markers as shown in Fig. 4d. We see that MDS coordinates which is actually obtained from the city distance matrix, are in good agreement with the original coordinates of the cities(shown in black markers).

4 Conclusions

1. Dimensionality reduction: When the data is not given but instead proximity(similarity, dissimilarity or distances) between pairs of objects is given, we apply MDS else we apply PCA.
2. **PCA**
   (a) First two principal components captures $\sim$ 63% of the variance i.e. the maximal variance within the breast cancer data.
   (b) A good distinction between the Malign and Benign tumor is still observed even when the data is projected in the space of first two dimensions.

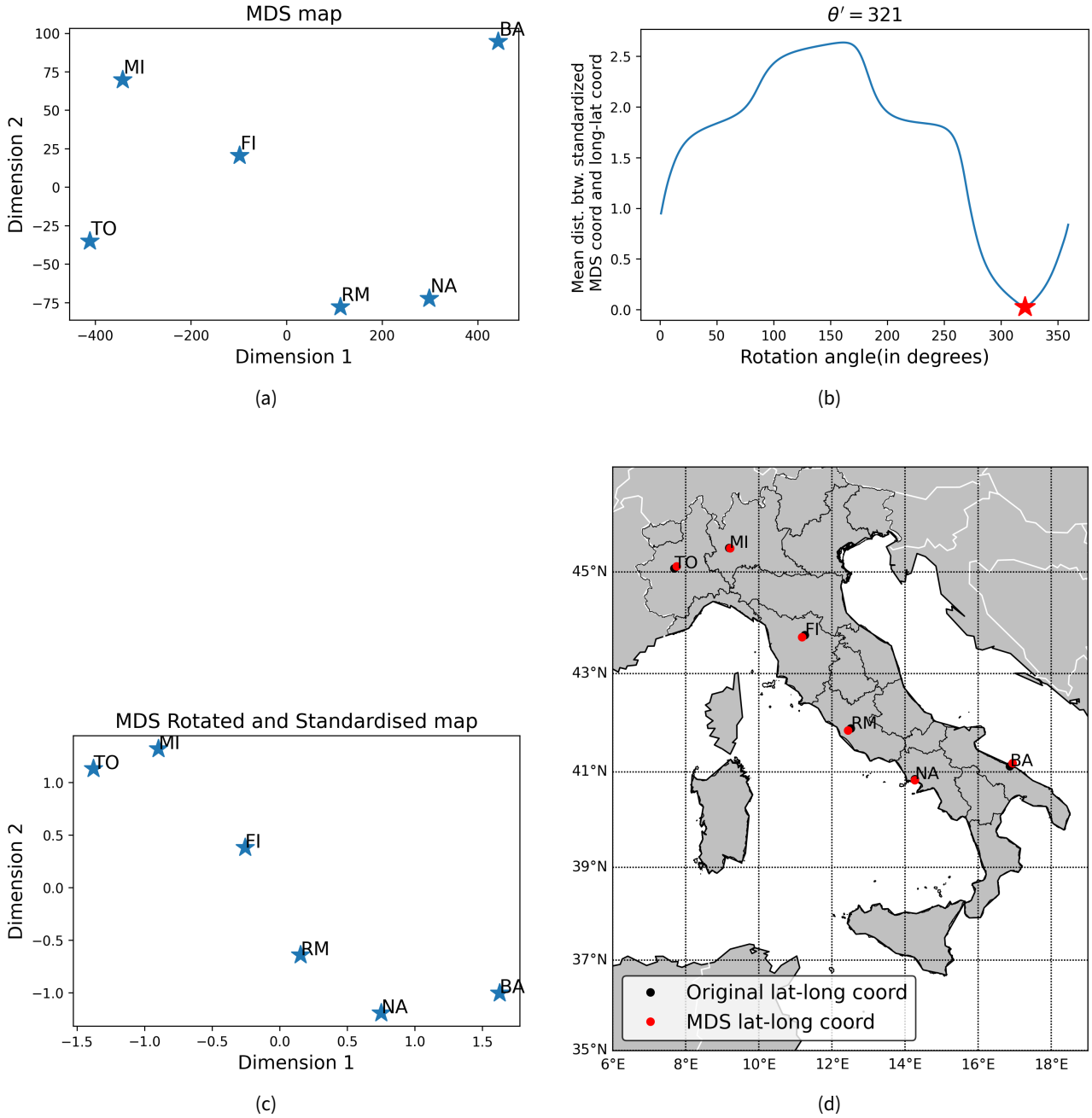


Figure 4. (a). MDS applied to distance matrix of 6 Italian cities. Dimension 1 and Dimension 2 coordinates of the cities are represented here. Note that there is no directional sense and we can guess that it has to be rotated to obtain approximate direction as in Fig. 3b. The Dimension 1 was actually flipped on the x axis by multiplying by -1 to x coordinates of cities and represented here. **(b.)** The x-axis represents the angle of rotation of the coordinates represented in Fig. 4a. The red marker is the optimal angle for which the rotated MDS coordinates almost closely coincides with the coordinates shown in the Map of Italy(Fig. 3b). **(c).** MDS coordinates are rotated by optimal angle θ' and then standardized to mean=0 and stdev.=1. **(d).** MDS coordinates of Fig.4c are rescaled into long-lat coordinates and represented on top Italian map as red markers along with the original known coordinates(in black markers).

3. **MDS**
   (a) Application of MDS on city distance matrix provides us the 2d xy-coordinates of the cities where the distance is preserved but not the sense of direction or exact location of cities with respect to map.
   (b) Rotation operator has to be applied on the MDS coordinates to recover the direction.
   (c) To recover the exact long-lat position, we need to know the proper parameters used for scaling like μ and σ so as to rescale the MDS coordinates after rotation.

References

[1] Ingwer Borg and Patrick J F Groenen. *Modern multidimensional scaling*. en. Springer Series in Statistics. New York, NY: Springer, May 2000.

[2] Jiachen Chen and W. Kenneth Jenkins. "Facial recognition with PCA and machine learning methods". In: *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*. 2017, pp. 973–976. DOI: 10.1109/MWSCAS.2017.8053088.

[3] Olvi L. Mangasarian Dr. William H. Wolberg W. Nick Street. *UCI Machine Learning Repository, Breast Cancer Wisconsin (Diagnostic) Data Set*. 1992. URL: https://archive.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+(diagnostic).

[4] V. K. Jirsa et al. "A theoretical model of phase transitions in the human brain". In: *Biological Cybernetics* 71.1 (May 1994), pp. 27–35. DOI: 10.1007/bf00198909. URL: https://doi.org/10.1007/bf00198909.

[5] A. Mead. "Review of the Development of Multidimensional Scaling Methods". In: *The Statistician* 41.1 (1992), p. 27. DOI: 10.2307/2348634. URL: https://doi.org/10.2307/2348634.

[6] G. Pasini. "PRINCIPAL COMPONENT ANALYSIS FOR STOCK PORTFOLIO MANAGEMENT". In: *International Journal of Pure and Apllied Mathematics* 115.1 (June 2017). DOI: 10.12732/ijpam.v115i1.12. URL: https://doi.org/10.12732/ijpam.v115i1.12.

[7] Karl Pearson. "LIII. iOn lines and planes of closest fit to systems of points in space/i". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (Nov. 1901), pp. 559–572. DOI: 10.1080/14786440109462720. URL: https://doi.org/10.1080/14786440109462720.

[8] Jonathon Shlens. "A Tutorial on Principal Component Analysis". In: *Educational* 51 (Apr. 2014).

[9] *simplemaps, Interactive maps and data, Italy cities database*. URL: https://simplemaps.com/data/it-cities.

[10] Florian Wickelmaier. "An introduction to MDS". In: (Apr. 2003).