Dimensionality Reduction - Computer Exercise

January 16, 2018

- In this exercise you will test several dimensionality reduction algorithms. Since you will apply the same algorithms to different datasets, we suggest that the code will be as generic as possible; the code should be given a generic dataset, and return some operation on that dataset (such that you will not have to write the same code for different datasets).
- You can use the algorithms in scikit-learn. Consult the documentation for setting hyper parameters of the used algorithms.
- Submission is individual or in pairs.
- Submit a ZIP file containing all your files (including txt/csv/pkl/mat files needed to run the python code) named with 9 digit of your ID. Submission Example: '200567989.zip'.
- Submit the code as .py files, and the solution as a 'Jupyter/IPython notebook' or in a PDF format with the names of the members participated in the work. Make sure there are no runtime errors in your code.
- Use python version 3.6/2.7

1 Load Datasets

1.1 Swiss Role Dataset

1. Create and plot a Swiss Role dataset in 3D(e.g swis-sroll).
   - Create two vectors, u and v, with 3000 random elements each. Each entry in u is uniformly distributed in $[0, 10]$ and each entry in v is uniformly distributed in $[0, 10]$.
   - Embed u and v in 3 dimension according to the Swiss Role embedding

$$
\begin{aligned}
x &= u\cos(u) \\
y &= u\sin(u) \\
z &= v
\end{aligned}
$$

- Create a 'color' vector. This vector will color the data points according to the *low dimensional* manifold. We will choose the color vector to be equal to u, we suggest to normalize the vector such that its values are in $[0, 1]$. Create 'color' according to,

  ```
  color = u/u.max().
  ```

  When plotting the result, set the attribute 'c', of the 'plot' function to 'color' (use mplot3d tutorial for further guidance).

2. Plot the data points (using the 'color' vector), after the transformation, in 3D, using scatter plot. Show 3 plots from different projection directions (use view_init to do so).

3. What is the dimension of each point in the dataset? What is the intrinsic dimension of the dataset (the dimension of the dataset's manifold)?

1.2 Face Dataset

The face dataset is a well known dataset, first analyzed in the Isomap paper. It consist of head images, shown from different angles and light.

1. Load the Face dataset from the course website (this dataset is a matlab file. We use loadmat to load the data into a python environment). This dataset contains 698, 64×64 pixel images. Use the following code to load the data (set the path appropriately):

   ```
   import matplotlib.pyplot as plt
   import scipy.io as sio
   data = sio.loadmat('./face_data.mat')
   images = data['images']
   print(np.shape(images))
   ```

2. Plot 3 images from the dataset.

3. What is the dimension of each point in the dataset? What is the intrinsic dimension of the dataset (the dimension of the dataset's manifold)?

1.3 MNIST Dataset

The MNIST dataset is another well known dataset. It consist of 28×28 images of handwritten digits between 0 to 9.

1. Load the MNIST dataset from the course website. We will work with the train dataset in this exercise (unless stated otherwise). Use the following code to load that data (set the path appropriately):

   ```
   import pickle, gzip
   import numpy as np
   f = gzip.open('./mnist.pkl.gz', 'rb')
   ```

```
train_set , valid_set , test_set = pickle.load(f,encoding='latin1')
f.close()
train_set_images = train_set[0]
train_set_images = train_set_images.T
train_set_digit_number = train_set[1]
np.shape(train_set_images)
```

If using Python2, omit the encoding input to the pickle.load function.

2. Create the dataset by filtering all digits that are not in 0-5 from 'train_set_images'. Use the variable 'train_set_digit_number' to determine the digit in every image.

3. Plot 3 images from the dataset.

4. What is the dimension of each point in the dataset? Is there an intrinsic dimension to this dataset (the dimension of the dataset's manifold)?

2 Dimensionality reduction

2.1 Swiss Role Dataset

In the section 'data' refers to Swiss role dataset. Although you will first perform dimensionality reduction(DR) to $d = 2$, we suggest to keep the dimension of the output, d, as a variable which can be set as you wish. Later you will check whether the choice of $d = 2$ is justified by considering several values for d.

2.1.1 PCA Based Methods

Plot all the results of this section in a *single* figure, in different axes using subplots function.

1. Perform DR to $\mathbb{R}^2$, using the PCA algorithm. Plot the results in $\mathbb{R}^2$.

2. Perform DR to $\mathbb{R}^2$,using the KPCA algorithm with the Gaussian Kernel

$$k(x, y) = \exp(-\frac{||x - y||_2^2}{2\sigma^2}).$$

   Where $|| \cdot ||_2$ is the L_2 norm. Choose σ appropriately and explain your choice. Plot the results in $\mathbb{R}^2$.

3. *Manually* construct features and add them to each data point, such that the PCA algorithm will have better performance. (Hint: consider the parametrization of the data). Perform Dimensionality Reduction to $\mathbb{R}^2$, apply the PCA algorithm and plot the results in $\mathbb{R}^2$.

2.1.2 Manifold Based Methods

Plot all the results of this section in a *single* figure, in different axes using subplots function. In the following sections reduce the dimensionality to d=2 using the three algorithms indicated. For all cases explain how you select the algorithm's parameters.

1. LLE.
2. Isomap.

2.1.3 Euclidean distances in the low-dimensional representation

Repeat the following for all algorithms you tested (PCA, KPCA, LLE and Isomap).

1. Choose two nearest neighbor data points *in the low dimensional space*, and compute their distance in the high dimensional space.
2. Choose two data points which are not in the set of the 40 nearest neighbors(NN) of one another, and compute their distance in the high dimensional space.
3. For each data point, compute the $K = 12$ NN set in the high and low dimensional spaces (K NN in the L_2 Euclidean norm). For each data point compute $C(x_i)$, which is defined to be the set of points which are in the $K = 12$ NN set in the low dimension, but not in the high dimension. Show the mean value of $C(x_i)$, $\frac{1}{n}\sum_{i=1}^{n}|C(x_i)|$, where $|\cdot|$ is the size of the set.
4. For the points which belong to the set $C(x_i)$, calculate their mean MDS measure of distances,

$$\mathrm{MDS}(C(x_i)) = \frac{1}{|C(x_i)|}\sum_{j\in C(x_i)}(||y_i - y_j|| - ||x_i - x_j||)^2,$$

   where y_i is the low dimensional representation of x_i. Show the mean value of $\mathrm{MDS}(C(x_i))$, $\frac{1}{n}\sum_{i=1}^{n}\mathrm{MDS}(C(x_i))$.

2.1.4 Verification of the intrinsic dimension

Implement the estimator of the intrinsic dimension, $\hat{m}$, and the estimators m_k, as discussed in the pdf in the courses website. Plot the values m_k as a function of k and discuss the value of $\hat{m}$. Does it fit your expectation from this type of dataset?

2.2 Face Dataset

2.2.1 PCA Based Methods

Repeat section 2.1.1, except for question 2.1.1.3, for the Face dataset.

2.2.2 Manifold Based Methods

Repeat section 2.1.2, for the Face dataset.

2.2.3 Euclidean distances in the low-dimensional representation

Repeat section 2.1.3, for the Face dataset. When asked to plot the value in the high dimensional representation draw the image.

2.2.4 Verification of the intrinsic dimension

Repeat section 2.1.4, for the Face dataset.

2.3 MNIST Dataset

2.3.1 PCA Based Methods

Repeat section 2.1.1, except for question 2.1.1.3, for the MNIST dataset.

2.3.2 Manifold Based Methods

Repeat section 2.1.2, for the MNIST dataset.

2.3.3 Euclidean distances in the low-dimensional representation

Repeat section 2.1.3, for the Face dataset. When asked to plot the value in the high dimensional representation draw the image.

2.3.4 Verification of the intrinsic dimension

Repeat section 2.1.4, for the Face dataset.

2.4 Questions

Answer in the following questions for all datasets (except when asked otherwise).

1. For the Swiss Role dataset, does the PCA with the manually added features perform better than KPCA? Why?

2. Explain the results. Which algorithm(s) performs the best on the tasks? Can you explain why (by discussing the properties of the algorithms and the type of datasets)?

3. What would you expect to be the value of $\frac{1}{n}\sum_{i=1}^{n}|C(x_i)|$ when using a 'good' DR algorithm?

4. Is the choice of $d = 2$ justified? i.e, is it justified to perform DR to $\mathbb{R}^2$ for the datasets relying on the estimator $\hat{m}$?

3 PCA Reconstruction

In this section you will compress *and* reconstruct images using the PCA algorithm. We will work with the dataset fetch_lfw_people from the sklearn library(1288 62×47 images of people).

Let $I, \hat{I} \in \mathbb{R}^{D_I}$, with $D_I = 62 \times 47 = 2914$, be the original image and the reconstructed image expressed as a vector (hint: in python, you can use the reshape function in the Numpy library switch between the matrix-vector representations). We assume that each pixel value, I_i, is in $[0, 1]$ (if not, normalize it accordingly). The reconstruction error of an image, $\epsilon_R(I)$, is defined as the absolute sum,

$$\epsilon_R(I) \triangleq \frac{1}{D_I} \sum_{i=1}^{D_I} |I_i - \hat{I}_i|,$$

Let $y(I) \in \mathbb{R}^{d_I}$ be the low dimensional representation of image I. Define the Compression Ratio(CR) of the image as:

$$CR \triangleq \frac{d_I}{D_I}$$

Verify that $\epsilon_R, CR \in [0, 1]$.

We wish to minimize both the reconstruction error *and* the CR. However, minimize the one will enlarge the second (e.g, setting $d_I = 0$ will cause ϵ_R to be big). We define the following notion of quality, which balances between the two criteria,

$$\text{quality} \triangleq E_{I \sim P(I)}[\epsilon_R(I) + CR] = E_{I \sim P(I)}[\epsilon_R(I)] + CR, \tag{1}$$

where we simply used the fact that the CR is a constant.

1. Create a dataset made from 1500 data points, where each data point is a 6×6 patch from a randomly drawn image (i.e, each data point is in $\mathbb{R}^{36}$). Specifically, repeat the following actions, for 1500 times, to create the data set:
   - Draw a random image from the data set.
   - Extract a 6×6 patch from a random place within the image.
   - Add the patch to the dataset.
2. Perform PCA on the training data to dimension $d \leq D$. Choose d as you wish.
3. Using the PCA, write a dimension reduction procedure to an image from the data set, I.
4. Calculate the average quality under the procedure you devised by drawing random samples from the data set, and calculate the average quality1.

5. Suggest a procedure to empirically justify your choice for d. Apply this procedure and show the value of d of your choice.

6. Reduce the dimension of the data as per the above procedure for 3 values of d, and then reconstruct one of input images and display it. How does it compare to the original image?