**ECE593 Introduction to Machine Learning Homework 4**

**Learning how to use Dimensionality Reduction**

Bo Lin

Introduction

In machine learning, while we want to do some classification or regression, we hope to extract the information from the dataset we have. If the dataset we are about to study has a high dimension, it will increase the complexity of the learning algorithm. It is important to find a way to map high dimensional dataset to lower dimensional space with minimum loss of information. This is the idea of dimensionality reduction. The following includes using four different method of dimensionality reduction to map the given dataset.

1. “pca.m” Testing

The given “pca.m” MATLAB code follows the algorithm of principle component analysis.

Assume that  is the projection of the 64 dimensional dataset to two-dimensional space. We define



where  will center the dataset on the origin. Since we would like to have uncorrelated , is the matrix that has two leading eigenvectors of  the estimator of .

The result of using “pca.m” to replicate figure 6.3 in the book is shown in Fig.1 and Fig.2 where in Fig1, the classes are shown by numbers, in Fig2, the classes are shown by distinct color.



Figure 1 Exact Replicate Figure 6.3



Figure 2 Figure 6.3 with distinct color



Figure 3 Reconstruction error

After x is projected to the z-space, we can calculated the error by backproject z to the original space as  .

The reconstruction error is shown in Fig 3 with number of principal components from 1 to 63. As dimension reduces, the mean-squared reconstruction error increases dramatically.

2. Principle Component Analysis

The PCA function in DRToolbox is basically the same as “pca.m” with default dimensional reduction to two-dimension space. The result is shown in Fig.4.



Figure 4 Principle Component Analysis Using DR Toolbox

3. Linear Discriminant Analysis

The LDA function in DRToolbox will first regard the first column of the dataset as the label set since it is a supervised method. It is similar to PCA because both of them use linear transformation. However, PCA does not take into account any difference in class. LDA explicitly attempts to model the difference between the classes of data. We want the classes to be separated as far apart as possible; also we want the classes be scatted in as small a region as possible.

Fisher’s linear discriminant is ***w*** that maximize  . The result of LDA is shown in Fig. 5.



Figure 5 Linear Discriminant Analysis Using DR Toolbox

From the result we can see that the dataset is projected alone one dimension ***w***. It is actually not well separated and there are many overlaps between different classes. This is probably due to the fact that Fisher’s linear discriminant is optimal if the classes are normally distributed. Another reason is that both dimensions and classes are high for this dataset and by reducing to one dimension it is really hard to separate; otherwise a lot of information will loss. For this dataset, the PCA performs better than LDA.

4. Multi-Dimensional Scaling

Multidimensional scaling places points in a low dimensional space such that Euclidean distance between them is as close as possible to the original space. For metric MDS, through checking the toolbox, I found out that MDS function is redirected to PCA function. So for this dataset, since it is already in metric form, MDS will perform exactly the same as PCA. However, in PCA we have the assumed multivariate distributed data and trying to figure out how to reduce unnecessary dimension. In MDS, we have the matrix of distances between the objects, and we are trying to figure out what the locations of these objects in space. I think the concept is different. The result is shown in Fig. 6.



Figure 6 Multi-Dimensional Scaling Using DR Toolbox

5. Stochastic Neighbor Embedding (SNE)

Stochastic Neighbor Embedding tries to place the dataset in a low-dimensional space so as to optimally preserve neighborhood identity. For each data point i and each potential neighbor j, the algorithm will first compute the asymmetric probability that they will become neighbors.



where d is the scaled squared Euclidean distance. We also set another possibility  for low-dimensional spaces where Gaussian neighborhoods is used with fixed variance.



The aim of the mapping is to match these two distributions as well as possible. It is achieved by minimizing a cost function of sum of Kullback-Leibler divergences between two distributions for each point. The result of SNE is shown in Fig.7. The performance of SNE looks better than PCA and LDA.



Figure 7 Stochastic Neighbor Embedding

6. Mystery Dataset

For mystery dataset, it has no label so we can only recognize different classes by their positions in the two-dimensional space. By looking at the data, some of the features are decimal numbers, negative values, many features has a large variance.

The result of PCA is shown in Fig. 8. The scale of the axis is much larger than the first dataset. And the data points seem well separated. Since it is unsupervised learning, I have no clue if it classifies the data points correctly. However I checked through different features, in some features, even with the same value, points are very far away which I think maybe the eigenvalue for that vector is not in leading rank.



Figure 8 Principle Component Analysis for Mystery dataset

The result of PCA is shown in Fig. 9. The mapping looks like unified distribution. But if zoom in the origin point, it is clearly seen that is it not unified distribution. So the classification seems “does not work”.

However, after read through the code of SNE function, I found out that the mapping of SNE is initially generated randomly very close to the original point “Y = 0.0001 \* rand(n, d);”, though the iteration of finding minimum cost function, the coordinate Y will extend. It is not like LDA that project points on one direction; it will project points from the origin point to further distance area.

By the end of the 2000 iteration, the error (cost) which is the sum of error of all points is 0.1537 0. It is quite small given a dataset with 294 points. Which means the possibility of p and q are similar. So it might work just I can’t tell.

In addition, I checked how t-SNE works in Fig. 11. It works even better than PCA. The only difference between t-SNE and SNE is that s-SNE uses a heavy-tailed Student-t distribution (with one-degree of freedom) to measure similarities between low-dimensional points in order to allow dissimilar objects to be modeled far apart in the map.

In coding, t-SNE seems to first use PCA to preprocessing the data unlike SNE just generate random mapping points.

So my conclusion is that for nun Gaussian distributed data, SNE will probably perform poorly, or “work” by separating data by near origin point and far away points.



Figure 9 Stochastic Neighbor Embedding for Mystery dataset



Figure 10 Origin Area zoomed in mapping



Figure 11 t-distributed Stochastic Neighbor Embedding