Report

On

"Supervised Dimensionality Reduction Techniques"

As a part of Course

Machine Learning (BITS F464)

By-

Ayush Singhal 2017A7PS0116P
Pratik Kakade 2017A7PS0086P
Yashdeep Gupta 2017A7PS0114P

Birla Institute of Technology and Science, Pilani
Sem II, 2019-20

Supervised by: Dr. Navneet Goyal

1. Objective:

To understand and implement two supervised dimensionality reduction techniques (FLD for multiple classes and Neighborhood component analysis for metric learning) and further compare the results for both supervised and unsupervised techniques on three chosen datasets.

2. Table of Contents:

3. Introduction:

Machine learning engages in data analysis to build automated models to carry out specific tasks such as classification or prediction of values based on certain attributes by drawing statistical inferences and patterns from the data. Hence, it is expected that the accuracy of models would increase with increase in the amount of available data. But as the number of dimensions (attributes) present in the data grow a problem is encountered. The number of datapoints present per unit volume becomes sparse and it becomes difficult to build a reliable model. This is known as the curse of dimensionality. One of the popular approaches to overcome the curse of dimensionality is dimensionality reduction. Dimensionality reduction is the process of reducing the number of features present in the dataset by filtering out the less significant data. It can be of two types - feature selection and feature extraction. Feature selection tries to select a subset of the features present while feature extraction creates new features by projecting data from higher dimensions to lower dimensional space.

Another way to classify dimensionality reduction approaches can be supervised and unsupervised. Formally speaking, given the data matrix $X^{N\times D}$ and label vector Y^N , where N represents total number of data points and D the number of dimensions, dimensionality reduction aims to create the best possible representation of the data matrix as $U^{N\times d}$ where $d < D$. While both approaches result in lower dimensional data, the process of supervised dimensionality reduction takes into account the label information to ensure that the dimension reduction is a direct aid to the process of model building.

3.1 Singular Value Decomposition

In linear algebra, singular value decomposition of a matrix is the factorization of the matrix that splits it into 3 components. By definition:

$$M_{m\times n} = U_{m\times m} S_{m\times n} V^{T}_{n\times n}$$

S is a diagonal matrix that contains the singular values of M which are square roots of the eigenvalues A^TA and AA^T .
U is an orthonormal, unitary matrix and is the eigenvectors of A^TA and its column vectors are known as the right singular vectors of M.
V is an orthonormal, unitary matrix and is the eigenvectors of AA^T and its column vectors are known as the left singular vectors of M.

To perform dimensionality reduction, we simply take the top d singular values of S and their corresponding eigenvectors from U and V .

$$M'_{m\times n} = U_{m\times d} S_{d\times d} V^{T}_{n\times d}$$

While it appears that the number of dimensions are still the same, what has reduced is the rank of the matrix and the resulting matrix contains less noise.

3.2 Principal Component Analysis

PCA is one of the most widely used unsupervised dimensionality reduction techniques. It operates by orthogonally projecting data from higher dimensions onto a lower dimensional linear space which is known as principal subspace. The directions along which the data is projected is such that variance of projected data is maximized.
By doing so it ensures that maximum information is preserved.

3.3 Fisher Linear Discriminant

Both Fisher's Linear Discriminant(FLD) and Principal Component Analysis(PCA) are popular dimensionality reduction algorithms that project data onto lower dimensions. However, unlike PCA which aims to preserve information by projecting data onto components in the direction of maximum variance, FLD projects data in the direction that results in maximum

class separability. Thus, FLD tends to be much more useful than PCA when the end goal is classification. This is as FLD takes into account the class labels. In real applications, both PCA and FLD are useful, and they can be combined to produce better linear dimensionality reduction results. PCA+FLD has been a very successful method in face recognition.

The idea proposed by Fisher is to maximize a function that will give a large separation between the projected class means while also giving a small variance within each class, thereby minimizing the class overlap. In other words the two objectives of Fisher's criterion are:-

- Increase between class variance
- Decrease within class variance

3.3.1 Mathematics behind Two class FLD

Let a training example for FLD be represented by (x_i, y_i) in which $x_i \varepsilon R^D$ is the original input vector that describes the i-th example in the training set, and $y_i \varepsilon \{0, 1\}$ where y_i is the label that specifies to which class x_i belongs. Hence, a training set X with N such pairs can be written as $\{(x_i, y_i)\}_{i=1}^{N}$ which is an abbreviation for the set $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$. Our training set is considered to be split into two parts by class:-

$$X_0 = \{x_i | 1 \leq i \leq N,\ y_i = 0\}$$

where $N_0 = |X_0|$ and

$$X_1 = \{x_i | 1 \leq i \leq N,\ y_i = 1\}$$

where $N_1 = |X_1|$. The mean vectors for the two classes are defined as:-

$$m_0 = \frac{1}{N_0} \sum_{x \in X_0} x$$

$$m_1 = \frac{1}{N_1} \sum_{x \in X_1} x$$

Our covariance matrices can be represented as:-

$$C_0 = \frac{1}{N_0} \sum_{x \in X_0} (x - m_0)(x - m_0)^T$$

$$C_1 = \frac{1}{N_1} \sum_{x \in X_1} (x - m_1)(x - m_1)^T$$

Let us assume the projection direction to be w. Then the projected value for x is $x^T w$. Thus, the variance for the projected examples is

$$\frac{1}{N_i} \sum_{x \in X_i} (x^T w - m_i^T w)^2$$

$$= w^T \left(\frac{1}{N_i} \sum_{x \in X_i} (x - m_i)(x - m_i)^T\right) w$$

$$= w^T C_i w$$

and the standard deviation is

$$\sigma_i = \sqrt{w^T C_i w}$$

The ratio that is to be maximized by Fisher's criterion can be represented as

$$\frac{|m_1^T w - m_0^T w|}{\sqrt{\sigma_0^2 + \sigma_1^2}}$$

which is same as maximizing

$$\frac{(m_1^T w - m_0^T w)^2}{\sigma_0^2 + \sigma_1^2}$$

$$\frac{w^T (m_1 - m_0)(m_1 - m_0)^T w}{w^T (C_0 + C_1) w}$$

The term $C_0 + C_1$ measures how scattered the two classes are. Other than covariance matrices, the scatter matrix also measures how scattered a set of points is. A scatter matrix for a set of points $z_1, z_2, ..., z_n$ is defined as

$$S = \sum_{i=1}^{n} (z_i - \bar{z})(z_i - \bar{z})^T ,$$

where $\bar{z}$ is the mean of points.Thus, scatter matrix is related to covariance matrix by

$$S_i = Ni\ Ci$$

As scatter matrix is used in FLD, we modify the FLD objective as:-

$$\frac{w^T (m_1 - m_0)(m_1 - m_0)^T w}{w^T (S_0 + S_1) w}$$

When the number of examples in both classes is the same, we can expect the same answer from both equations.

We introduce two additional symbols

$$S_B = (m_1 - m_0)(m_1 - m_0)^T$$

$$S_W = (S_0 + S_1)$$

where S_B represents between class scatter matrix which measures the scatter caused by between class means and S_W represents a within-class scatter matrix which measures how scatter is the original dataset within each class. Both are $D \times D$ matrices.

We define the final FLD objective as

$$J = \frac{w^T S_B w}{w^T S_W w}$$

Since we want to maximize J, we differentiate with respect to w and set it to 0

$$\frac{\delta J}{\delta w} = \frac{2((w^T S_w w) S_B w - (w^T S_B w) S_V w}{(w^T S_W w)^2} = 0$$

$$S_B w = J\, S_W w$$

$$S_W w = \tfrac{1}{J}\, S_B w$$

$$= \tfrac{1}{J}\, w(m_1 - m_0)(m_1 - m_0)^T w$$

$$= \tfrac{1}{J}\, w(m_1 - m_0)^T w(m_1 - m_0)$$

$$= c(m_1 - m_0)$$

where c is defined as $\frac{1}{J}\, w(m_1 - m_0)^T w$ and c is a scalar value. We can ignore it as we will take the normalized vector anyways.Thus we define the optimal projection direction to be:

$$S_W^{-1}(m_1 - m_0)$$

Thus by calculating $x_i^T w$ for every example (x_i, y_i) we get the reduced dataset $\{(x_i^T w, y_i)\}_{i=1}^N$.

Note: Since it is not necessary that S_W^{-1} will exist, we take the Moore-Penrose pseudoinverse instead.

3.3.2 Mathematics behind Multi-class FLD

We can extend most of the mathematics to the multi class scenario. Let a training example for FLD be represented by (x_i, y_i) in which $x_i \varepsilon R^D$ is the original input vector that describes the i-th example in the training set, and $y_i \varepsilon Y$ where $Y \varepsilon \{1, 2, ..., K\}$ denotes the K classes and y_i is the label that specifies to which class x_i belongs.

Let X_k be the subset of dataset X containing all elements belonging to k-th class.

$$X_k = \{x_i | 1 \leq i \leq N,\ y_i = k\}$$

Then N_k, m_k, C_k and S_k be the size, mean, covariance matrix and scatter matrix, respectively, of X_k

$$N_k = |X_k|$$

$$m_k = \frac{1}{N_k} \sum_{x \in X_k} x$$

$$C_k = \frac{1}{N_k} \sum_{x \in X_k} (x - m_k)(x - m_k)^T$$

$$S_k = N_k C_k$$

The within class scatter matrix will be the sum of all scatter matrices of individual classes

$$S_W = \sum_{k=1}^{K} S_k$$

Directly extending the definition for between class matrix results in:

$$S_B = \sum_{i=1}^{K} \sum_{j=1}^{K} (m_i - m_j)(m_i - m_j)^T$$

However it is computationally inefficient. Through some algebraic manipulation and filtering out of constant values a better definition is:

$$S_B = \sum_{k=1}^{K} (m_k - \overline{m})(m_k - \overline{m})^T$$

where $\overline{m} = \frac{1}{K} \sum_{k=1}^{K} m_k$ is the mean of means. This new equation is the scatter of K mean vectors and is less computation intensive.

However this definition is still lacking as it does not take class imbalance of the dataset into account.

The scatter matrix for the entire training set is denoted by S_T and is called the total scatter matrix. It is defined as:

$$S_T = \sum (x_i - m)(x_i - m)^T$$

where $m = \frac{1}{N} \sum_{i=1}^{N} x_i$ is the global mean of the training set.

$$S_T - S_W = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T$$

We redefine this to be the between-class scatter matrix

$$S_B = \sum_{k=1}^{K} N_k (m_k - m)(m_k - m)^T$$

It differs from the previous definition if all class have equal sizes only by a factor of $\frac{K}{N}$, which does not affect the final answer. Also it is effective in class imbalanced datasets as it gives more weightage to classes having more examples. Therefore:

$$S_B = S_T - S_W$$

The final FLD objective J has the same form as of that in the 2 class problem and derivating it with respect to w results in the same equation

$$S_B w = J S_W w$$

$$S_w^{-1} S_B w = J w$$

Thus, J would be the eigenvalue and w the eigenvector for the matrix $S_w^{-1} S_B$. For reducing the dataset to $d < D$ dimensions, we compute the weight vector w by taking the eigenvectors corresponding to the top d eigenvalues.

3.4 Neighbourhood Component Analysis

Neighbourhood Component Analysis(NCA) is a distance metric learning based supervised dimensionality reduction algorithm that extracts features to aid in classification. The base principle underlying this algorithm is to find a projection of the data from a higher dimensional space to a lower dimensional space where the kNN algorithm can be applied effectively. The distance metric that it makes use of is the Mahalanobis distance which is an effective metric in high dimensional spaces and the test error is defined via leave-one-out cross validation.

3.4.1 Mathematics behind NCA

Let our labeled dataset be consist of n real-valued input vectors $x_1, x_2, ..., x_n$ in R^D and their labels be denoted by $y_1, y_2, ..., y_n$ where $y_i \varepsilon \{C_1, C_2, ..., C_k\}$ representing the k classes.

The algorithm intends to find a linear transformation on the input space such that in the transformed space KNN will perform well. We select the Mahalanobis (quadratic) distance metric as the similarity measure to be used by the KNN algorithm after transformation. Then, we are effectively learning the metric $Q = w^T w$ (where w is the transformation matrix) and distance between two points in the transformed space will be defined as:

$$d(x, y) = (x - y)^T Q(x - y) = (x - y)^T w^T w(x - y) = (wx - wy)^T (wx - wy)$$

The leave one out classification error of KNN is a discontinuous function as a slight change in w could result in sudden jump or fall due to change in neighbours. Instead, a smooth and well behaved measure of nearest neighbour performance is used. We denote the probability that a point i selects another point j as its nearest neighbour and inherits its class label as p_{ij} defined as:

$$p_{ij} = \frac{exp(-\|wx_i - wx_j\|)}{\sum_{k \neq i} exp(-\|wx_i - wx_j\|)}, \; p_{ii} = 0$$

We can thus define the probability that point i is correctly classified as:

$$p_i = \sum_{j \in C_i} p_{ij}$$

Since we want to maximize number of correctly classified points, the objective function $f(w)$ that we wish to maximize is:

$$f(w) = \sum_i p_i$$

Differentiating with respect to the transformation matrix w results in a gradient rule that can be used for learning(we denote $x_{ij} = x_i - x_j$):

$$\frac{\delta f}{\delta w} = -2w \sum_i \sum_{j \in C_i} p_{ij} (x_{ij} x_{ij} x_{ik} - \sum_k p_{ik} x_{ik} x_{ik}^T)$$

On reordering,

$$\frac{\delta f}{\delta w} = 2w \sum_i (p_i \sum_k p_{ik} x_{ik} x_{ik}^T - \sum_{j \varepsilon C_i} p_{ij} x_{ij} x_{ij}^T)$$

Thus, the above function needs to be maximized using a gradient based optimizer to find the transformation matrix w.

4. Datasets Used and Applications:

4.1 Wine Dataset[1]-

This dataset is the result of chemical analysis of wines grown in the same region in Italy but from three different cultivators. The analysis determined the quantities of 13 constituents found in each of the three types of wines. All the attributes are continuous, with a total of 178 total instances available.

The model generated can be used to classify the given wine sample by feeding the chemical analysis of the wine as input.

4.2 Ionosphere Dataset[2]:-

This radar data was collected by a system in Goose Bay, Labrador. This system consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. The targets were free electrons in the ionosphere. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere. Received signals were processed using an autocorrelation function whose arguments are the time of a pulse and the pulse number. There were 17 pulse numbers for the Goose Bay system. There are 351 instances and 34 attributes.

The model generated can be used to determine the presence of structure in the ionosphere.

4.3 Connectionist Bench (Sonar, Mines vs. Rocks) Dataset[3]-

This dataset consists of two classes, mines and rocks. Dataset about mines contains 111 attributes and the one with rocks contains 97 attributes. Mine dataset was made by bouncing sonar signals off a metal cylinder from various angles and various conditions. Similar procedure was done for Rock's dataset but by using rocks instead of a cylinder. Each pattern is a set of 60 numbers in the range 0.0 to 1.0. Each number represents the energy within a particular frequency band, integrated over a certain period of time. The final dataset consists of 60 attributes with 208 instances.

The model generated can be used for differentiating between solar mines and rocks by analysing the bounced signals.

5. Implementation details:

The implementation for this assignment was done in R. The code applies 4 dimensionality reduction techniques, namely Principal Component Analysis, Singular Value Decomposition, Multi-class Fisher Linear Discriminant and Neighbourhood Component Analysis, the first two unsupervised and the latter two supervised. The performance all 4 is compared by applying a SVM-based as well as a KNN- based classifier. Accuracy was taken as a performance evaluation metric while ensuring that the class were balanced so that it would be reliable.

To run the code, open the main-datasetname.r files. There are 3, each for a particular dataset.

Three libraries of R were utilised:-

- caTools: Used function sample.split() for creating a split in dataset for train and test
- e1071:Used function svm() for SVM classifier
- class: Used function knn() for KNN classifier

6. Results:

Both supervised and unsupervised dimensionality reduction techniques were applied on the mentioned three datasets. In supervised reduction technique, we have implemented FLD and Neighbour Component Analysis. For unsupervised reduction techniques, we have used SVD and PCA. Further, we have checked the accuracy for these techniques on two classifiers, SVM and KNN. Here are the results for different datasets-

Sonar (reduced to 20 dimensions)

Classifier\Reduction Technique	Original	FLD	SVD	PCA	NCA
SVM	0.9193548	1	0.919354	0.969230	0.9538462
KNN	0.6052632	0.979591	0.605263	1	0.7346939

Confusion matrix for original dataset

SVM

Actual\Predicted	1	2	3
1	22	0	0
2	1	20	4
3	0	0	15

KNN

Actual\Predicted	1	2	3
1	13	0	0
2	2	4	10
3	1	2	6

Confusion matrix for FLD reduced dataset

SVM

Actual\Predicted	1	2	3
1	22	0	0
2	1	26	0
3	0	0	17

KNN

Actual\Predicted	1	2	3
1	17	0	0
2	0	18	0
3	0	1	13

Confusion matrix for SVD reduced dataset

SVM

Actual\Predicted	1	2	3
1	21	1	0
2	1	21	3
3	0	0	15

KNN

Actual\Predicted	1	2	3
1	13	0	0
2	2	4	10
3	1	2	6

Confusion matrix for PCA reduced dataset

SVM

Actual\Predicted	1	2	3
1	22	0	0
2	1	24	0
3	0	0	17

KNN

Actual\Predicted	1	2	3
1	17	0	0
2	0	18	0
3	0	0	14

Confusion matrix for NCA reduced dataset

SVM

Actual\Predicted	1	2	3
1	21	0	1
2	1	25	0
3	0	1	16

KNN

Actual\Predicted	1	2	3
1	15	2	0
2	0	15	3
3	1	7	6

Wine (reduced to 8 dimensions)

Classifier\Reduction Technique	Original	FLD	SVD	PCA	NCA
SVM	0.9193548	1	0.919354	0.95	0.7166667
KNN	0.6052632	1	0.605263	1	0.75

Confusion matrix for original dataset

SVM

Actual\Predicted	1	2	3
1	22	0	0
2	1	20	4
3	0	0	15

KNN

Actual\Predicted	1	2	3
1	13	0	0
2	2	4	10
3	1	2	6

Confusion matrix for FLD reduced dataset

SVM

Actual\Predicted	1	2	3
1	20	0	0
2	0	24	0
3	0	0	16

KNN

Actual\Predicted	1	2	3
1	14	0	0
2	0	16	0
3	0	0	10

Confusion matrix for SVD reduced dataset

SVM

Actual\Predicted	1	2	3
1	21	1	0
2	1	21	3
3	0	0	15

KNN

Actual\Predicted	1	2	3
1	13	0	0
2	2	4	10
3	1	2	6

Confusion matrix for PCA reduced dataset

SVM

Actual\Predicted	1	2	3
1	20	0	0
2	1	21	2
3	0	0	16

KNN

Actual\Predicted	1	2	3
1	14	0	0
2	0	16	0
3	0	0	10

Confusion matrix for NCA reduced dataset

SVM

Actual\Predicted	1	2	3
1	19	0	1
2	1	18	5
3	1	9	6

KNN

Actual\Predicted	1	2	3
1	12	1	1
2	0	13	3
3	0	5	5

Ionosphere (reduced to 15 dimensions)

Classifier\Reduction Technique	Original	FLD	SVD	PCA	NCA
SVM	0.9043478	0.927272	0.869565	0.91818	0.9181818
KNN	0.7638889	0.920454	0.847222	0.90909	0.9318182

Confusion matrix for original dataset

SVM

Actual/Predicted	1	2
1	36	11
2	0	68

KNN

Actual/Predicted	1	2
1	14	14
2	3	41

Confusion matrix for FLD reduced dataset

SVM

Actual/Predicted	1	2
1	35	6
2	2	67

KNN

Actual/Predicted	1	2
1	25	6
2	1	56

Confusion matrix for SVD reduced dataset

SVM

Actual/Predicted	1	2
1	37	10
2	5	63

KNN

Actual/Predicted	1	2
1	19	9
2	2	42

Confusion matrix for PCA reduced dataset

SVM

Actual/Predicted	1	2
1	35	6
2	3	66

KNN

Actual/Predicted	1	2
1	23	8
2	0	57

Confusion matrix for NCA reduced dataset

SVM

Actual/Predicted	1	2
1	35	6
2	3	66

KNN

Actual/Predicted	1	2
1	25	6
2	0	57

7. Observations And Result:

It is noted that in majority cases dimensionality reduction is an effective tool in filtering out noise and overcoming majority. Also, supervised dimensionality reduction tends to outperform or do as well as unsupervised dimensionality reduction techniques. Also, NCA tends to work better with a KNN classifier than a SVM classifier which is to be expected since it is optimized to perform well with the KNN algorithm.

Supervised dimensionality reduction has proven to be a powerful tool when its purpose aligns with the models goal. While it may certainly not preserve as much information as unsupervised techniques, it operates such that the output is an aid to the process of model building in the larger picture. In most high dimensional datasets, however researchers tend to apply unsupervised dimension reduction first, usually PCA, and then supervised on top of that. This mostly done as supervised dimension reduction has a higher computational workload and the model is able to take advantage of both the techniques. It has been observed that the resulting dataset always outperforms either of the techniques on their own.

8. References:

1. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/wine]. Irvine, CA: University of California, School of Information and Computer Science
2. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/Ionosphere]. Irvine, CA: University of California, School of Information and Computer Science..
3. Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml/datasets/connectionist+bench+(sonar,+mines+vs.+rocks)] . Irvine, CA: University of California, School of Information and Computer Science.
4. https://cs.nju.edu.cn/wujx/teaching/06_FLD.pdf
5. https://www.cs.toronto.edu/~hinton/absps/nca.pdf