MWSUG 2017 - Paper AA11

Dimensionality Reduction using Hadamard, Discrete Cosine and Discrete Fourier Transforms in SAS®

Mohsen Asghari, CECS Department, University of Louisville, Louisville KY 40292;
Aliasghar Shahrjooihaghighi, CECS Department, University of Louisville, Louisville KY 40292;
Ahmed Desoky, CECS Department, University of Louisville, Louisville KY 40292

ABSTRACT

Dimensionality reduction studies various techniques to transform data in the most compact and efficient manner that allows modeling, analyzing, and predicting information with insignificant errors. Principle component analysis (PCA) is a method for reducing the dimensionality by decreasing the number of variables and selecting a smaller subset of uncorrelated transformed variables called principal components. PCA is data dependent and requires the computation of the correlation matrix of input data as well as performs Singular Value Decomposition (SVD) of that matrix. Hadamard, Discrete Cosine Transform (DCT), and Discrete Fourier Transform (DFT) are orthogonal transformations that are not data dependent and reduce the dimensionality by decreasing the correlation of the transform components. In this paper, we implemented Hadamard, DCT, and DFT in SAS on a standard dataset. Also, we compared the results of these transformations and PCA technique.

INTRODUCTION

Reducing large number of variables in a dataset using an efficient and fast algorithm is a challenge. One solution to face this challenge is reducing dimensionality of the data. (O'Rourke, 2013) Describes dimensionality reduction methods developed through statistics and machine learning (Cunningham, 2015). In this paper, we have implemented some of the famous ones using SAS IML programming.

Principal component analysis is a method for reducing the dimensionality by decreasing the number of variables and selecting a smaller subset of uncorrelated transformed variables called principal components. PCA is data dependent and requires the computation of correlation matrix of input data as well as the Singular Value Decomposition (SVD) matrix. Hadamard, Discrete Cosine Transform (DCT) and Discrete Fourier Transform (DFT) are orthogonal transformations that are not data dependent (SAS, 2013) (Wicklin, 2013), they reduce the dimensionality by decreasing the correlation of the transform components.

We gives a brief explanation of the 4 Algorithms in the first section, and describes the implementation of SVD, Hadamard, DCT and DFT in SAS IML programming·

DEFINITIONS

SINGULAR VALUE DECOMPOSITION (SVD)

In principal component analysis the number of components that are generated is the same as the number of variables of the original data. However, we should determine the number of the significant transform components. One of the famous classification methods of the SVD transform components is eigenvalue analysis, also known as Kaiser Criterion (Kaiser, 1991). Eigenvalues are divided into two groups, greater than 1.0 and less than 1.0. Components with eigenvalues greater than 1.0 are accounting for greater amount of variances and we can count them as significant components that have enough information. The other method is Scree Test (Cattell, 1966) which relies on sorting the eigen values and determining where they level off. Third way is to look at proportion of eigen values. Calculations of proportions exist in PROC FACTOR. Based on SAS documentation the proportion of an eigen value is defined by:

$$P = \frac{Eigenvalue\ for\ the\ component\ of\ interest}{Total\ eigenvalues\ of\ the\ correlation\ matrix} \qquad (1)$$

One of the outputs of PROC FACTOR is a column named “Proportion” which represents the proportion of each eigenvalue.

HADAMARD

Hadamard transform is one of the most well-known orthogonal transforms. It is a generalized class of Fourier transforms. A Hadamard matrix is a square matrix whose entries are either +1 or -1 and the rows and columns are mutually orthogonal (K. R. Rao, 1990). Hadamard matrix exists for every positive value of N which is a power of 2.

$$H_1 = 1, \qquad H_{2N} = \frac{1}{\sqrt{2}}\begin{pmatrix} H_N & H_N \\ H_N & -H_N \end{pmatrix} \tag{2}$$

; where N = 1, 2, 4, 8, ……

Hadamard function in SAS is used to generate the Hadamard matrix.

DISCRETE COSINE TRANSFORM

DCT has a significant impact in digital signal processing (K. R. Rao, 1990). It is used in several standards such as jpg and mpg.

DCT of a data sequence X(m), m=0, 1,...,(N-1) is defined as:

$$G_x(0) = \frac{\sqrt{2}}{N}\sum_{m=0}^{N-1} X(m) \tag{3}$$

$$G_x(k) = \frac{2}{N}\sum_{m=0}^{N-1} X(m)\cos\frac{(2m+1)k\pi}{2N}, \qquad k = 0, \dots, (N-1) \tag{4}$$

DISCRETE FOURIER TRANSFORM

Fast Fourier Transform (FFT) algorithm is the fast system to compute the Discrete Fourier Transform (DFT) of a sequence. FFT reduces the number of computations needed to calculate DFT from O (n^2) to O ($nlogn$) (Jain, 1989).

Let $[x_0, \dots, x_{N-1}]$ be a vector of complex numbers. The DFT is defined by the formula

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi kn}{N}} \qquad k = 0, \dots, N-1 \tag{5}$$

INTRODUCTION TO SAS IML

PROC IMPORT

This procedure import external file as a SAS data set (SAS, 2013) (Wicklin, 2013).

```
proc import DATAFILE="PATH"
    DBMS= dlm | csv
    Out = "SAS Data Set Name"
    replace;
    getnames = yes;
    run;
```

READ OBSERVATION FROM SAS DATASET

READ statement (SAS, 2013) is used to create an SAS/IML matrix from an existing SAS dataset the general form of the READ statement is as follows:

READ < range > <VAR operand> < *WHERE(expression)* > < *INTO name* > ;

Range	Specifies the SAS datasets that we want to read
Operand	Specifies the variables that exists in the SAS dataset and we want to bring them to the matrix
Expression	An expression that is evaluated as being true or false. It can be used as a filter for data that we want to load in the matrix.
Name	Names a target matrix for the data.

Table 1. READ statement Parameters

SAS/IML language allows the elimination of non-numerical variables. To do this, use READ statement with reading all or part of the variables and specify numerical data by using the keyword _NUM_ in the VAR clause.

GRAPHS IN SAS/IML

SAS/IML language supports several graphs to report the data such as BAR, HISTOGRAM, SERIES etc. in this section, we want to show how you can call SAS procedure to create BAR charts and SERIES charts. For more information, you can check SAS/IML guide book (SAS, 2013).

For creating SERIES chart you need to follow statement as follow:

```
CALL SERIES (x,y
GRID={"X" <,"Y">}
LABEL={XLabel<,Ylabel>}
XVALUES=xValues
YVALUES=yValues);
```

X,Y	Specify a vectors to draw the chart based on the X vector and y vector these must be the same size
ORDER	Specifies the order in which discrete tick values are to be placed on the categorical axis.
GRID	Specifies whether to display grid lines for the X or Y axis. This option corresponds to the GRID option in the XAXIS and YAXIS statements. Valid values follow: GRID={X} displays grid lines for the X axis. GRID={Y} displays grid lines for the Y axis. GRID={X, Y} displays grid lines for both axes.
LABEL	Specifies axis labels for the X or Y axis.
XVALUES	Specifies a vector of values for ticks for the X axis.
YVALUES	Specifies a vector of values for ticks for the Y axis.

Table 2. Series Parameters

DIMENSIONALITY REDUCTION IN SAS/IML

In this section, we discuss the implementation of statistical methods for dimensionality reduction. In Figure 1, we can see the flowchart of the program that created for calculating the statistical methods.

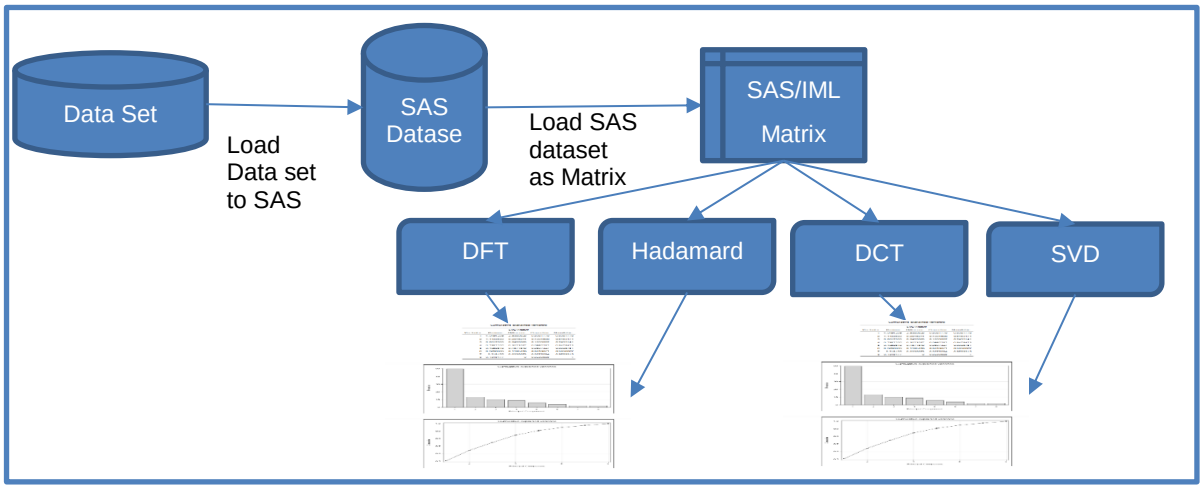

Figure 1. Flowchart of SAS/IML program

We divide this section into three parts; First part is data preparation, in second part we discuss statistical methods we used, and in the third part, we will show the results.

DATA PREPARATION

Data preparation has two parts.

Part 1

The following procedure loads the data into a SAS dataset named *mydata.* Procedure standard obtains standard data set named *STEX* with mean =0 and std = 1.

```
proc import datafile="[Path]\Housing.csv"
out=mydata dbms=csv replace;
getnames=yes;
run;

proc standard data=mydata mean=0 std=1 out=STEX;
RUN;
```

Part 2

Pocedure iml creats data matrix A using SAS data set *STEX*

```
Proc iml;
use STEX; read all var _num_ into A[colname=names];
```

STATISTICAL METHODS

DISCRETE COSINE TRANSFORMATION

Discrete cosine transform (DCT) is a widely-used method for image compression. It can also be used in dimensionality reduction in other data besides image. DCT can be performed by simple matrix computations. SAS does not have a function to calculate the DCT matrix. Function dctmtx generates an $n \times n$ matrix using equations (3) and (4)

```
start dctmtx(n);
```

```
    pi = constant("pi");
    cc = repeat(0:n-1,n,1);
    rr = cc`;
     c = sqrt(2 / n) * cos(pi * (2*cc+1) # rr / (2 * n));
    c[1,] = c[1,]/ sqrt(2);
return c;
finish;
```

DCTDR function computes the transform coefficients using matrix multiplication A*s; A is the data matrix and s is the dctmtx with size NxN.

```
start DCTDR(A);
    size = ncol(A);
    s = dctmtx(size);
    w = A*s;
    variance = var(w);
    Factor = prepareresult(variance,ncol(A));
    return Factor;
finish;
```

Hadamard Transformation

SAS supports the Hadamard matrix. We can create this matrix by calling the *"Hadamard(size)"* . The following code does Hadamard transformation:

```
start HCDR(A);
    size = ncol(A);
    h = hadamard(size);
    w = A*h;
    variance = var(w);
    Factor = prepareresult(variance,ncol(A));
    return Factor;
finish;
```

h contains the Hadamard matrix and A represents our data and w contains the transformed components . Then we calculate the variance of each column and names the vector "variance". Then prepare the result by calling "prepareresult" and at the end return the principal components.

Discrete Fourier Transformation

Fast Fourier Transform (FFT) algorithm is the fast system to compute the Discrete Fourier Transform (DFT) of a sequence. FFT reduces the number of computations needed to calculate DFT from O(N^2) to O ($NlogN$) (Jain, 1989). The following code does the DFT transformation of data matrix A.

```
start DFTRD2(A);
    r = nrow(A);
    do i = 1 to r;
          x = fft(A[i,]);
          amplitude = x[,1]##2 + x[,2]##2;
          T = T // amplitude`;
    end;
    w = abs( T` * A);
    variance = var(w);
    Factor = prepareresult(variance,ncol(A));
    return Factor;
finish;
```

RESULT PREPARATION

The function "prepareresults" calculates these information and creates the results table.

```
  start prepareresult(r,n);
      r = r`;
      ColVar = J(n,2);
      do i = 1 to n;ColVar[i,1] = i;ColVar[i,2] = r[i];end;
      Factor = J(n,5);
      Sumvariance = sum(ColVar[,2]);
      call sort(ColVar,2,2);
      lastproportion = 0;
      do i = 1 to nrow(ColVar);
            currentvar = ColVar[i,2];
            proportion = currentvar / Sumvariance;
            Cumulative = proportion + lastproportion;
            Factor[i,1] = i; /*ColVar[i,1];*/
            Factor[i,2] = currentvar;
            Factor[i,4] = proportion;
            Factor[i,5] = Cumulative;
            difference  = 0;
            if i ^= nrow(ColVar)then do;
                  nextcurrentvar = ColVar[i+1,2];
                  difference = currentvar  - nextcurrentvar;
            end;
            Factor[i,3] = difference ;
            lastproportion  = Cumulative;
      end;
  return Factor;
```

"PrintReport" prints the results table as well two graphs. The scree plot graphs the Normalized Variance against the component number. The second graph includes the cumulative normalized variances which is the used for the proper selection of the number of principal components.

```
start RrintReport(factor,names,reportlable);

      orderedname = J(nrow(factor),1,"                    ");

      do i = 1 to nrow(factor);

            indexname =  factor[i,1];

            orderedname[i,1] = names[1,indexname];

      end;

      Columname = {"Components" "Variance" "Difference" "Proportion"
"Cumulative"};

      print factor[colname = Columname label = reportlable];

      /*Scree plot*/

      call Series(factor[,1],factor[,4])  grid={X Y} option="markers"
label={"Principal Component","Varience"};

      /*Varience Explained*/

      g = repeat({"Cumulative","Proportion"}, 1, nrow(factor));

      g = g`;

      x = factor[,1] || factor[,1];

      y = factor[,5] || factor[,4];

      call Series(x,y) group=g  grid={X Y} option="markers" label={"Principal
Component","Proportion"};
```

```
finish;
```

EXPERIMENTAL RESULT

DATA EXPLANATION

We have chosen Housing dataset which is provided by UCLA. This data set contains 16 features and 506 samples which they represent in order: CRIM per capita crime rate by town, ZN proportion of residential land zoned for lots over 25,000 sq. INDUS proportion of non-retail business acres per town CHAS Charles River dummy variable (= 1 if tract bounds river; 0 otherwise) NOX nitric oxides concentration (parts per 10 million) RM average number of rooms per dwelling AGE proportion of owner-occupied units built prior to 1940 DIS weighted distances to five Boston employment centers RAD index of accessibility to radial highways TAX full-value property-tax rate per $10,000 PTRATIO pupil-teacher ratio by town B 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town LSTAT % lower status of the population MEDV Median value of owner-occupied homes in $1000's, entropy and STD of each row.

DISCRETE COSINE TRANSFORMATION

Results of DCT algorithm are obtained using the following instructions:

```
%let _timer_start_DCT = %sysfunc(datetime());
DCTFactor = DCTDR(A);
dur_DCT = (datetime() - &_timer_start_DCT)*1000;
call RrintReport(DCTFactor,names,"Discrete Cosines Result");
runningtimes = runningtimes  || dur_DCT;
```

Components	Variance	Difference	Proportion	Cumulative
1	3.7240887	1.6821704	0.2327555	0.2327555
2	2.0419183	0.6916541	0.1276199	0.3603754
3	1.3502642	0.1426024	0.0843915	0.4447669
4	1.2076618	0.1175081	0.0754789	0.5202458
5	1.0901537	0.2586636	0.0681346	0.5883804
6	0.8314901	0.0234354	0.0519681	0.6403485
7	0.8080546	0.116778	0.0505034	0.690852
8	0.6912766	0.0533092	0.0432048	0.7340567
9	0.6379675	0.0205056	0.039873	0.7739297
10	0.6174618	0.0123333	0.0385914	0.8125211
11	0.6051286	0.0375029	0.0378205	0.8503416
12	0.5676257	0.0892401	0.0354766	0.8858182
13	0.4783856	0.0146019	0.0298991	0.9157173
14	0.4637837	0.0203005	0.0289865	0.9447038
15	0.4434831	0.0022272	0.0277177	0.9724215
16	0.441256	0	0.0275785	1

Table 3. Discrete Cosines Variances

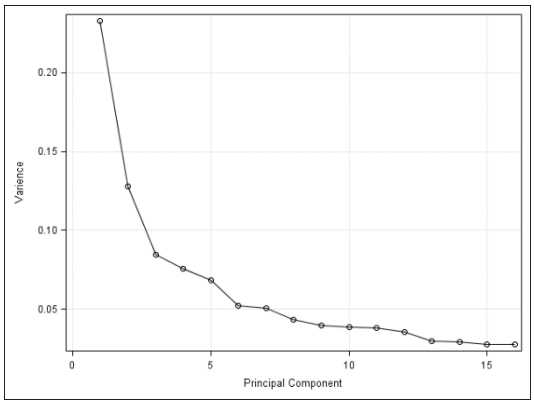

Figure 2. Scree Plot

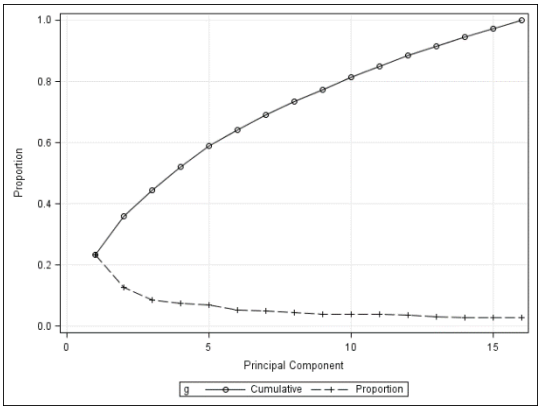

Figure 3. Proportion/Cumulative Plot

HADAMARD TRANSFORMATION

Results of Hadamard algorithm are obtained using the following instructions:

```
%let _timer_start_Hadamard = %sysfunc(datetime());
HFactor = HCDR(A);
dur_H = (datetime() - &_timer_start_Hadamard)*1000;
call RrintReport(HFactor,names,"Hadamard Matrices Result");
runningtimes = runningtimes  || dur_H;
```

Components	Variance	Difference	Proportion	Cumulative
1	47.392746	23.810112	0.1851279	0.1851279
2	23.582634	0.193269	0.0921197	0.2772476
3	23.389365	3.1560017	0.0913647	0.3686123
4	20.233364	2.8008854	0.0790366	0.4476489
5	17.432478	1.4075239	0.0680956	0.5157445
6	16.024954	0.5025595	0.0625975	0.578342
7	15.522395	1.3735879	0.0606344	0.6389763
8	14.148807	2.4965995	0.0552688	0.6942451
9	11.652208	0.6224841	0.0455164	0.7397615
10	11.029723	0.0525698	0.0430849	0.7828464
11	10.977154	0.3972271	0.0428795	0.8257259
12	10.579927	1.6097697	0.0413278	0.8670537
13	8.9701569	0.3074983	0.0350397	0.9020934
14	8.6626586	0.4151443	0.0338385	0.9359319
15	8.2475143	0.0935995	0.0322169	0.9681488
16	8.1539148	0	0.0318512	1

Table 4. Hadamard Transformation Variance Explanation

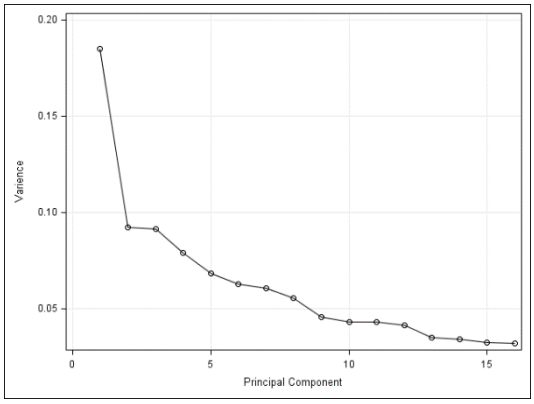

Figure 4. Scree Plot

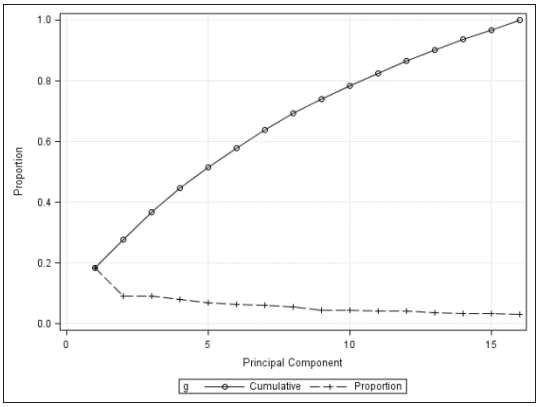

Figure 5. Proportion/Cumulative Plot

DISCRETE FOURIER TRANSFORMATION

Results of DFT algorithm are obtained using the following instructions:

```
%let _timer_start_DFT = %sysfunc(datetime());
DFTFactor = DFTDR(A);
dur_DFT = (datetime() - &_timer_start_DFT)*1000;
call RrintReport(DFTFactor,names,"Discrete Furier Transformation Result");
runningtimes = runningtimes  || dur_DFT;
```

Components	Variance	Difference	Proportion	Cumulative
1	37578632	21774714	0.3916618	0.3916618
2	15803917	7199368.8	0.1647157	0.5563775
3	8604548.6	652082.92	0.0896806	0.6460581
4	7952465.7	1646834.7	0.0828843	0.7289424
5	6305631	3291208.5	0.0657202	0.7946626
6	3014422.5	50766.232	0.0314177	0.8260803
7	2963656.3	193433.83	0.0308886	0.8569689
8	2770222.4	288973.16	0.0288725	0.8858414
9	2481249.3	518311.95	0.0258607	0.9117021
10	1962937.3	223842.24	0.0204586	0.9321608
11	1739095.1	220869.19	0.0181257	0.9502864
12	1518225.9	500150.61	0.0158237	0.9661101
13	1018075.3	154719.81	0.0106109	0.9767209
14	863355.48	58984.237	0.0089983	0.9857192
15	804371.24	238549.53	0.0083835	0.9941027
16	565821.71	0	0.0058973	1

Table 5. Discrete Fourier Transformation Variances

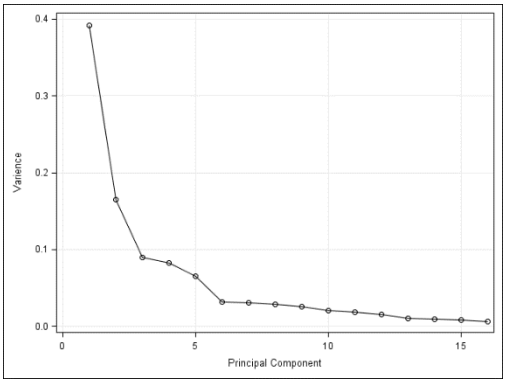

Figure 6. Scree Plot

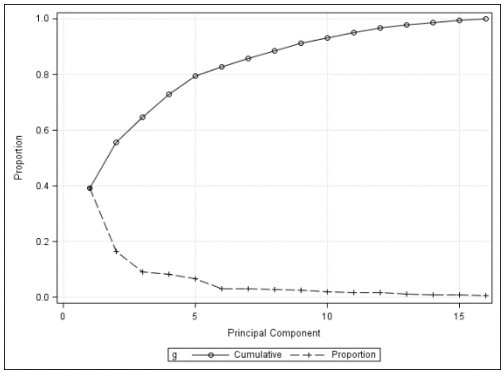

Figure 7. Proportion/Cumulative Plot

SVD TRANFORMATION

Results of SVD algorithm are obtained using the following instructions:

```
Proc princomp data = mydata out=compdata;
run;
proc print data=compdata;
run;
```

Components	Variance	Difference	Proportion	Cumulative
1	7.22016824	5.29890704	0.4513	0.4513
2	1.92126120	0.40455643	0.1201	0.5713
3	1.51670477	0.48687365	0.0948	0.6661
4	1.02983112	0.01440416	0.0644	0.7305
5	1.01542696	0.25375416	0.0635	0.7940
6	0.76167280	0.14112541	0.0476	0.8416
7	0.62054739	0.06944963	0.0388	0.8804
8	0.55109776	0.25061100	0.0344	0.9148
9	0.30048676	0.03557002	0.0188	0.9336
10	0.26491675	0.03472680	0.0166	0.9501
11	0.23018995	0.04029659	0.0144	0.9645
12	0.18989336	0.00650873	0.0119	0.9764
13	0.18338464	0.06013110	0.0115	0.9878
14	0.12325354	0.06706640	0.0077	0.9956
15	0.05618714	0.04120952	0.0035	0.9991
16	0.01497762	0	0.0009	1.0000

Table 6 SVD Variances

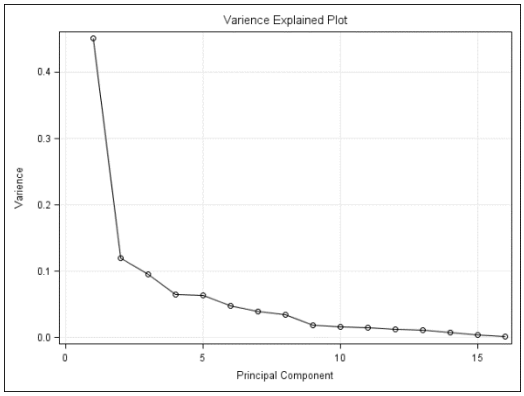

Figure 8. Scree Plot

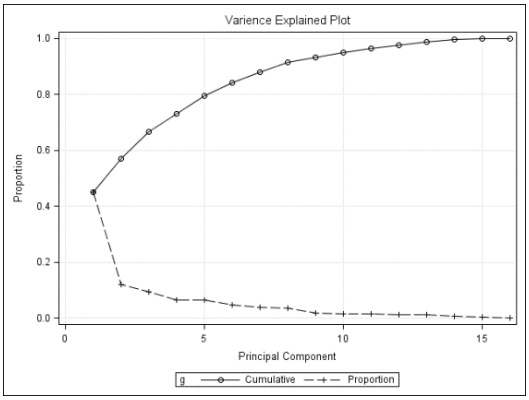

Figure 9. Proportion/Cumulative Plot

CONCLUSION

To compare these techniques, we plotted all the scree plots and cumulative plots together. Based on the Cumulative plot in Figure 11, we can see that DFT and SAS principal component (SVD) act very similar, also DCT and Hadamard are close. However, based on the Scree plot in the Figure 10, we can see that the first two components of DFT and SVD contains approximately 60% of total energy while in other two algorithms, it is around 30%. Based on that, we can conclude that first two components of DFT and SVD can represent the data in a more efficient way. Therefore, reducing dimensionality of data using DFT and Principal Component (SVD) is more efficient than DCT and Hadamard.

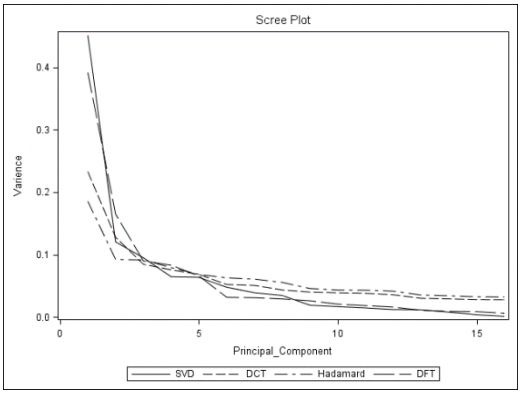

Figure 10. Scree Plot of all the methods

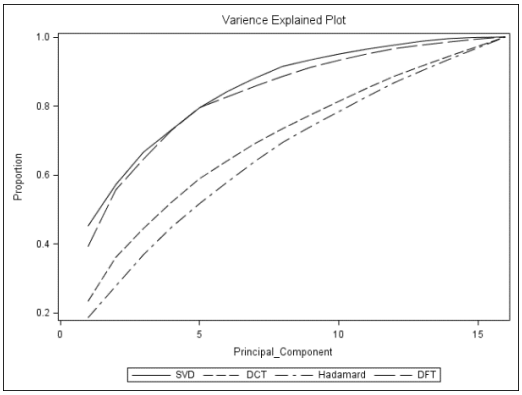

Figure 11. Cumulative Plot of all the methods

"SYSFUNC" macro calculates the running time of the algorithm. Table 7 lists the running time of each algorithm in milliseconds. Hadamard algorithm is the fastest, second position belongs to DCT, third position belongs to DFT and at the end SVD is the slowest.

SVD	DCT	Had	DFT
3.9999485	1.0001659	0.9999275	2.9997826

Table 7 Running Times

REFERENCES

Cattell, R. B. (1966). The Scree Test For The Number Of Factors. *Multivariate Behavioral Research , 1*(2), 245-276.

Cunningham, J. P. (2015). Linear dimensionality reduction: survey, insights, and generalizations. *Journal of Machine Learning Research, 16*(1), 2859--2900.

Jain, A. K. (1989). *Fundamentals of digital image processing.* Prentice-Hall.

K. R. Rao, P. Y. (1990). *Discrete cosine transform: algorithms, advantages, applications.* San Diego, CA: Academic Press Professional.

Kaiser, H. F. (1991). Coefficient Alpha for a Principal Component and the Kaiser-Guttman Rule . *Psychological Reports, 68*(3), 855-858.

O'Rourke, N. a. (2013). A step-by-step approach to using SAS for factor analysis and structural equation modeling.

SAS. (2013). *SAS/IML 13.1 User's Guid.* Carolina: SAS institute.

Wicklin, R. (2013). Getting Started with the SAS/IML® Language. *SAS Global Forum* (p. 21). SAS Institute Inc.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Mohsen Asghari
mohsen.asghari@louisville.edu

Aliasghar Shahrjooihaghighi
a0shah07@louisville.edu

Ahmed Desoky
ahmed.desoky@louisville.edu