Data Mining (CSE 572)

Spring 2020

PROJECT TOPIC TITLE: CSE 572 Data Mining Project 1

Name: Ziming Dong

Date: 2/10/2020

School of Computing, Informatics, and Decision Systems Engineering Arizona State University,
Tempe, AZ, USA, 85281
Spring 2020

Abstract:

For this project, I am responsible for extracting features from fixed data gave by datafolder. I only use CGM data cell array and CGM timestamp cell array for 5 subjects. I choose four different methods to extract features of data, they are polynomial curve fitting, fast fourier transform (FFT), skewness and interquartile range. Then I provide the feature matrix to PCA and derive the new feature matrix, I have 13 principle components and I find the 5 highest eigenvalues correspond to my feature matrix. Finally, I can figure out which feature(s) type makes more sense for my data.

Description:

To begin with: Clean the data

The CGM data cell array and CGM timestamp cell array make up the tissue glucose level every 5 mins for 2.5 hours during many lunch meals. The number of time series should be 30, but different data files have different lengths, so I choose the first 30 cells in the CGM time series data files and their glucose level data files. Then I find there are so many missing values in several data files, so I drop all of them and combine the 5 clean data csv files as a dataframe by the numpy function.

n [6]:
```
# Combine the five csv files and clean the data
path= "/Users/dongziming/Desktop/data mining/assignment 1/DataFolder"
```

n [7]:
```
all_files = glob.glob(os.path.join(path,"CGMSeriesLunch*.csv"))
all_df = []
for f in all_files:
    df = pd.read_csv(f, sep=',')
    df['file'] = f.split('/')[-1]
    all_df.append(df)
merged_df = pd.concat(all_df, ignore_index=True, sort=True)
```

n [8]:
```
#Drop the extra time series for every subject.
total=merged_df.drop(['cgmSeries_42','cgmSeries_41','cgmSeries_40','cgmSeries_39','cgmSeries_38','cgmSeries_37','cgmSer
```

n [9]:
```
#Drop the missing value array.
df1=total.dropna()
```

[10]:
```
final=df1.iloc[:,::-1]
nofrows= final.shape[0]
print (nofrows)
```

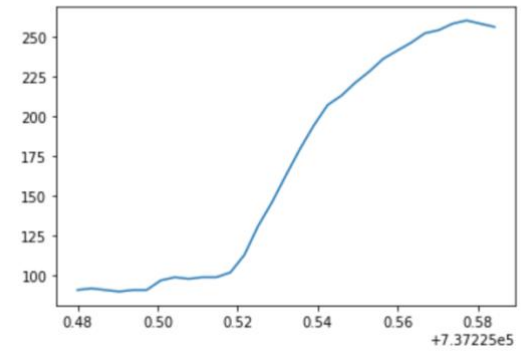
```
186
```

After cleaning the dataframe, I find there are total of 186 meals for 5 subjects in CGM data files without missing value.

Step1*:* Extract 4 different types of time series features from only the CGM data cell array and CGM timestamp cell array.

I spend a lot of time to think about which type of feature I should use for this time-series data. I simply plot the first meal of the first subject graph, to see the tendency of the data. When the time grows, the tissue glucose increases at sometimes and decreases a little at the end of time. I plot other meals and I find they are almost the same tendencies.

```
plt.figure()
plt.plot(a.iloc[0],b.iloc[0])
plt.show()
```



Thus, I choose Polynomial Curve Fitting, Fast Fourier transform, Skewness and interquartile range as feature types to do the feature engineering.

Step2: For each time series explain why you chose such features.

I think these four methods are all making sense to present the feature of data, I will explain the reasons one by one:

Polynomial Curve fitting is the process of constructing a mathematical function to best fit a series of data points. It requires the relation between dependent variables to be linear. For the CGM data, there should be a relationship that when time grows up, glucose has some changes. Thus, polynomial Curve fitting can summarize the relationship between glucose and time-series variables.

Fast Fourier transform can be used to simply characterize the magnitude and phase of a signal. It is a fast implementation of the DFT, it saves a lot of time by taking advantage of the symmetry properties and reduces the number of operations to N logN. Because some CGM data cell is hard to see the feature of them in the time domain, but if they are transformed into the frequency domain, it is easy to see the features. This is why I choose the FFT method.

Skewness refers to whether the data curve shift to be the left or to the right. After I plot a lot of time series data, I found there either positively-skewed or negatively-skewed, thus I think this method should be meaningful to extract features of data.

Interquartile Range measures the variability of the data, it divides the data set into quartiles. Quartiles divide the data set into four equal parts. The value is equal to the difference between 75th and 25th percentiles. The reason why I choose this method is that I find the first 25th percentiles and last 25 percentiles always have a large variation, so it might be a good feature extraction for my data.

Step3: Show the values of each of the features and argue that your intuition in step2 is validated or disproved?

Polynomial Curve fitting:

```
In [60]: #Feature 1 Ploynomial curve fitting
         z_total=[]
         z_norm=[]
         for i in range(0,186):
             yaxis=np.asarray(final.iloc[i])
             x=np.arange(30)
             z1=np.polyfit(x,yaxis,5)
             z2=z1/z1.max(axis=0)
             z_total.append(z1)
             z_norm.append(z2)
         z_total

Out[60]: [array([-5.16314423e-05,  5.55933951e-03, -1.91396409e-01,  2.47083028e+00,
                 -5.11660689e+00,  1.15239676e+02]),
          array([ 1.09898099e-04, -7.74377897e-03,  1.58981028e-01, -5.20504231e-01,
                 -2.49007302e+00,  9.41564171e+01]),
          array([-8.18942448e-05,  1.02549526e-02, -4.17472223e-01,  6.34453522e+00,
                 -2.56510982e+01,  1.13386036e+02]),
          array([-1.39015279e-04,  1.21446522e-02, -3.76229467e-01,  4.75514701e+00,
                 -1.51371311e+01,  6.73301018e+01]),
```

I choose quantic polynomial fitting in the function. There are 6 values shown for each time series. The reason why I choose 5 as the number of polynomials is that I do not want to overfit the data curve since my data scale is not very big, 5 should

be fine. The polyfit function returns an equation with 6 values goes through all points from the time series. It will return a best fit that minimizes the squared error. After I check the value of each array, there are no particular large, small or zero values in the array, so this method should be meaningful.

Fast Fourier Transform:

```
In [71]: #Plot Feature 2 FFT
         fft_x=np.linspace(0,30,30)
         fft_y=final.iloc[0]
         fft_yy=abs(fft(fft_y))
         plt.plot(fft_x,fft_yy)
         fft_yy

Out[71]: array([5306.        ,  539.94964822,  389.05705489,  247.8904353 ,
                 190.52949194,  133.40539719,  115.47253396,  105.7442707 ,
                  86.66972069,   91.09518147,   83.14445261,   74.4358826 ,
                  70.97953157,   69.10011561,   65.25703222,   68.        ,
                  65.25703222,   69.10011561,   70.97953157,   74.4358826 ,
                  83.14445261,   91.09518147,   86.66972069,  105.7442707 ,
                 115.47253396,  133.40539719,  190.52949194,  247.8904353 ,
                 389.05705489,  539.94964822])
```

For the FFT result, the frequency values are symmetric, which accord with the definition of fourier transform. After transform from the time domain to the frequency domain, I can simply find the highest values in the frequency domain and collect them as features. I only take the five maximum frequency value in the frequency domain for each time series, the higher frequency value should represent the tendency of the data curve. I also remove the first highest value in each array because it is too big

```
In [53]: #Feature 2 FFT
         f_total=[]
         f_norm=[]
         for i in range(0,186):
             x=np.linspace(0,29,30)
             y=final.iloc[i]
             yy=abs(fft(y))
             yy1=np.delete(yy,0)
             yy2=np.unique(yy1)
             Max5=np.partition(yy2,-5)[-5:]
             normMax5=Max5/Max5.max(axis=0)
             f_total.append(Max5)
             f_norm.append(normMax5)
         f_total

Out[53]: [array([133.40539719, 190.52949194, 247.8904353 , 389.05705489,
                  539.94964822]),
          array([ 113.06192993,  221.30831431,  226.3123164 ,  324.81529978,
                 1216.36433319]),
          array([ 24.51530134,  41.88675583,  94.60844093, 188.6807845 ,
                  874.02004871]),
          array([126.99085798, 132.50283016, 257.42130828, 417.1095914 ,
                  690.248913571),
```

Skewness:

```
In [88]: #feature 3 skewness
         s_total=[]
         for i in range(0,186):
             s1=np.array(skew(final.iloc[i]))
             s_total.append(s1)
         s_total
```

```
Out[88]: [array(-0.24476876),
          array(-0.01466725),
          array(0.31559477),
          array(-0.57472905),
          array(-1.51687427),
          array(-0.20470875),
          array(-0.09651655),
```

All of skewness values are make sense because they are either positive value or negative value, the data series which get positive value are right-skewness, the data series which get negative value are left-skewness. Then we can take a look for the average of values to see the tendency of the data curve. Thus, this feature extraction method makes sense.

Interquartile Range:

```
In [89]: #feature 4 interquartile range
         q_total=[]
         for i in range(0,186):
             q=final.iloc[i]
             q1_x=np.quantile(q,0.25,interpolation='midpoint')
             q3_x=np.quantile(q,0.75,interpolation='midpoint')
             q4=np.array(q3_x-q1_x)
             q_total.append(q4)
         q_total
```

```
Out[89]: [array(53.5),
          array(134.5),
          array(75.),
          array(57.5),
          array(26.),
          array(82.5),
          array(91.),
          array(43.),
          array(146.5),
          array(31.5),
          array(65.),
```

The value of each time series represents the density of the data curve, if the result is too small, it means that most of the data are distributed in the middle part. If the result is very big, it means the data lunch meal influence a lot for people`s glucose.

Step4: Create a feature matrix where each row is a collection of feature time series.

```
In [20]: z_norm=np.array(z_norm)
         f_norm=np.array(f_norm)

In [21]: matrix=np.append(z_norm,f_norm,axis=1)

In [22]: matrix1=np.vstack((s_norm,q_norm))

In [23]: matrix1=matrix1.T

In [24]: feature_matrix=np.concatenate((matrix,matrix1),axis=1)

In [90]: feature_matrix

Out[90]: array([[-4.48035297e-07,  4.82415408e-05, -1.66085515e-03, ...,
                  1.00000000e+00,  4.12479818e-01,  2.44845361e-01],
                [ 1.16718650e-06, -8.22437728e-05,  1.68847788e-03, ...,
                  1.00000000e+00,  4.87090162e-01,  6.62371134e-01],
                [-7.22260411e-07,  9.04428178e-05, -3.68186629e-03, ...,
                  1.00000000e+00,  5.94177519e-01,  3.55670103e-01],
                ...,
                [ 1.86957844e-06, -1.31247265e-04,  3.11585802e-03, ...,
                  1.00000000e+00,  5.87272534e-01,  1.72680412e-01],
                [ 9.17497106e-07, -5.03586572e-05,  6.24017530e-04, ...,
                  1.00000000e+00,  6.73370048e-01,  1.03092784e-01],
                [ 4.31446830e-07, -2.98464377e-05,  5.70003573e-04, ...,
                  1.00000000e+00,  4.70220457e-01,  1.57216495e-01]])

In [91]: feature_matrix.shape

Out[91]: (186, 13)
```

Because I will use this feature matrix to PCA, so I normalize all of the value to 0-1 range. Then I create the feature matrix with 186x13 shape. 186 is the number of meals for 5 subjects. The 13 is the feature`s length.

Step 5: Provide this feature matrix to PCA and derive the new feature matrix. Chose the top 5 features and plot them for each time series.

```
In [26]: pca=PCA(n_components=13)

In [27]: principalComponents=pca.fit_transform(feature_matrix)

In [28]: principalDf = pd.DataFrame(data = principalComponents
                     , columns = ['principal component 1', 'principal component 2','principal component 3',
                                  'principal component 4','principal component 5','principal component 6','principal component
                                 'principal component 8','principal component 9','principal component 10','principal component
                                ,'principal component 12','principal component 13'])

In [29]: principalDf
```

Out[29]:

	principal component 1	principal component 2	principal component 3	principal component 4	principal component 5	principal component 6	principal component 7	principal component 8	principal component 9	principal component 10	principal component 11	principal component 12	principal component 13
0	0.332782	0.119730	0.142293	0.016678	-0.046730	-0.019347	0.023734	0.005133	1.570745e-04	-2.613530e-06	3.093516e-08	2.836884e-19	-1.392422e-25
1	-0.364532	0.019488	0.264388	0.017724	0.000345	-0.009975	0.039988	-0.014109	9.805495e-06	3.225864e-06	4.899808e-09	-1.779396e-20	-4.775312e-25
2	-0.325202	-0.176680	-0.047350	0.209279	-0.043259	0.045728	0.010089	0.003173	2.958064e-04	-4.781219e-06	-2.006447e-08	-3.865761e-19	-1.737101e-25
3	0.151563	0.166797	0.050973	0.226509	-0.070241	0.065742	-0.025539	0.009710	3.505402e-04	2.837966e-06	3.686299e-08	1.926998e-20	9.916275e-25

Because the feature-length is 13, so I set up 13 principal components in my principal dataframe.

Then I try to see the percentages for each principal component, it shows the first principal component counts 53% of the total, and the second principal component counts 17% of the total. Thus, I would like to select the top five features from these two components. I try to see values in principal components then I know which feature`s importance is higher than the other.

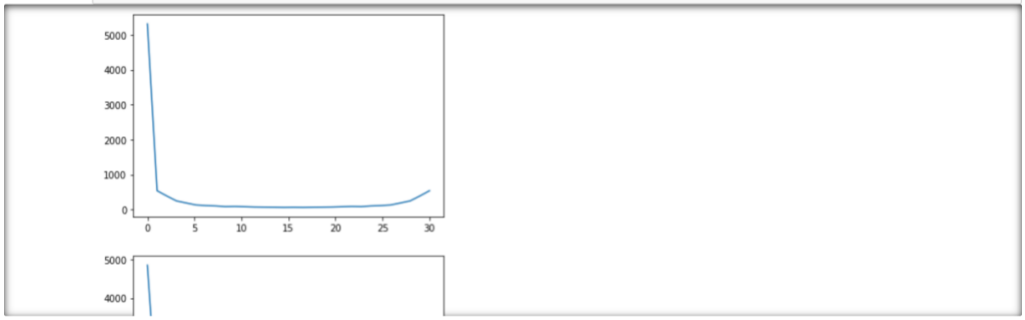
```
In [30]: print(pca.explained_variance_ratio_)

[5.38206980e-01 1.68025301e-01 1.30017118e-01 8.82504126e-02
 5.03979857e-02 1.96053888e-02 4.40636066e-03 1.08924262e-03
 1.21094623e-06 1.59315440e-10 1.17445720e-15 4.90368462e-37
 2.75251189e-48]
```

```
In [31]: values=abs(pca.components_)

In [32]: values

Out[32]: array([[9.76099145e-07, 6.80369037e-05, 1.46908951e-03, 9.52509634e-03,
                 4.14621526e-02, 0.00000000e+00, 1.98937592e-01, 2.55406210e-01,
                 4.23484743e-01, 7.08818019e-01, 0.00000000e+00, 6.78195287e-02,
                 4.54991980e-01],
                [8.14238276e-07, 5.07108192e-05, 9.09902055e-04, 3.83896244e-04,
                 1.52983591e-01, 2.77555756e-17, 4.10245567e-02, 6.59269485e-02,
                 4.06401700e-02, 1.27436956e-01, 0.00000000e+00, 9.66914500e-01,
                 1.33229931e-01],
```

After selecting the four highest value from first principal components and 1 highest value in the second principal components, I find the 7th, 8th, 9th, 10th, 12th feature in my feature matrix have the higher value than others. I find the 7th, 8th, 9th and 10th feature are the components of FFT feature, they corresponds the four highest frequency value in the frequency domain. And the 12th feature correspond to the skewness feature. Thus, I just plot these two features for 180 time series.

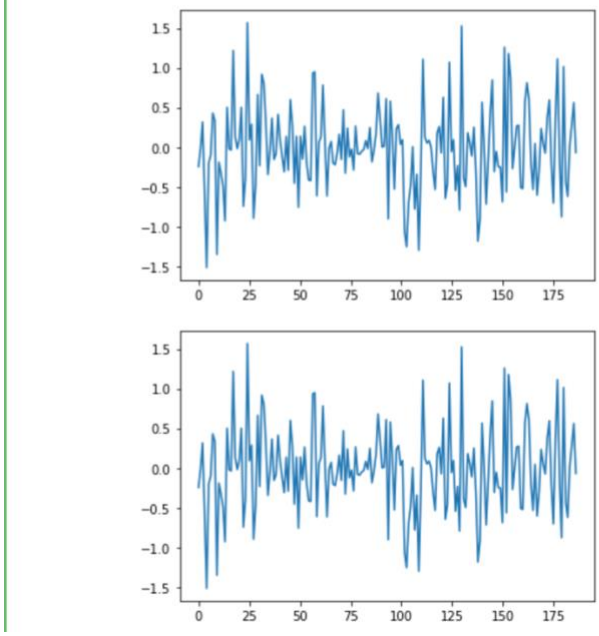
FFT:

```
In [93]: #Plot Feature 2 FFT
for i in range(0,186):
    fft_x=np.linspace(0,30,30)
    fft_y=final.iloc[i]
    fft_yy=abs(fft(fft_y))
    plt.figure()
    plt.plot(fft_x,fft_yy)
    plt.show()
```



Skewness:

```
In [95]: #Plot skewness
for i in range(0,186):
    sk_x=np.linspace(0,186,186)
    sk_y=s_total
    plt.figure()
    plt.plot(sk_x,sk_y)
    plt.show()
```



Step6: For each feature in the top 5 argue why it is chosen as a top five feature in PCA?

Since I have explained the the first principal components and the second principal components counts 70% of total. So I would like to look at values for these two principal components to select the best 5 features out of 13.

```
In [32]: values

Out[32]: array([[9.76099145e-07, 6.80369037e-05, 1.46908951e-03, 9.52509634e-03,
                 4.14621526e-02, 0.00000000e+00, 1.98937592e-01, 2.55406210e-01,
                 4.23484743e-01, 7.08818019e-01, 0.00000000e+00, 6.78195287e-02,
                 4.54991980e-01],
                [8.14238276e-07, 5.07108192e-05, 9.09902055e-04, 3.83896244e-04,
                 1.52983591e-01, 2.77555756e-17, 4.10245567e-02, 6.59269485e-02,
                 4.06401700e-02, 1.27436956e-01, 0.00000000e+00, 9.66914500e-01,
                 1.33229931e-01],
```

For values in the first principle components, the four highest values are 0.1989, 0.2554, 0.4234, 0.70881 and the highest value in the second principal components is 0.9669. The correspond index of these 5 values in the feature matrix is 7th, 8th, 9th, 10th and 12th. The correspond feature type is Fast Fourier transform and skewness. Thus, they are top 2 feature type in my feature engineering.

Conclusion:

The top 2 important feature extraction method is Fast Fourier transform and Skewness in my feature engineering. PCA helps me to reduce the dimension of the data set and lower the noise of the data. The larger variation of the data set can be seen as representing the information we would like to keep, for example, I would like to keep the first principal component and second principal component in my data set. Actually, I will not say my choices of feature type are all good, there are a lot of other different feature extraction methods for the time series data set. I would like to explore more of them to get the best solution to feature engineering.