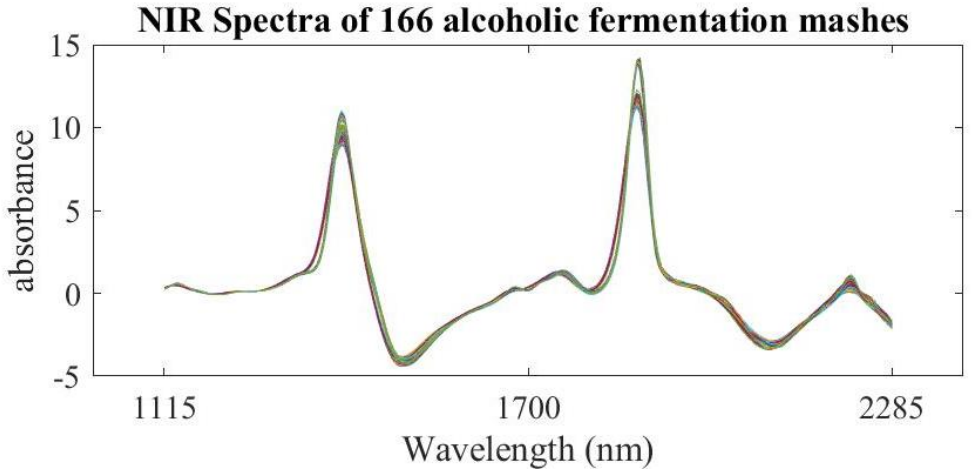
ISyE 8803 –Topics on High Dimensional Data Analytics

Exam I

- You are not allowed to discuss exam problems with anybody nor receive aid on this exam.
- You are expected to observe the Georgia Tech Honor Code throughout the exam.
- Exam is due on June 26 at 11:59pm (U.S. Eastern Time). Late submission is NOT accepted. Submit your solutions via Canvas.
- Submit your exam answers in PDF format. For problems that require programming, supply your codes in separate files.

Question 1: (34 points)

Near infrared (NIR) spectroscopy was applied for a compositional analysis of 166 alcoholic fermentation mashes of different feedstock (rye, wheat and corn). For each sample, we observe one spectrometric curve which corresponds to the absorbance measured from 1115 nm to 2285 nm with 5 nm spectral sampling. In this problem, we will predict the concentration of ethanol. "`Question1.mat`" dataset contains the near infrared spectroscopy absorbance "`NIR`", and ethanol concentration "`Ethanol`".



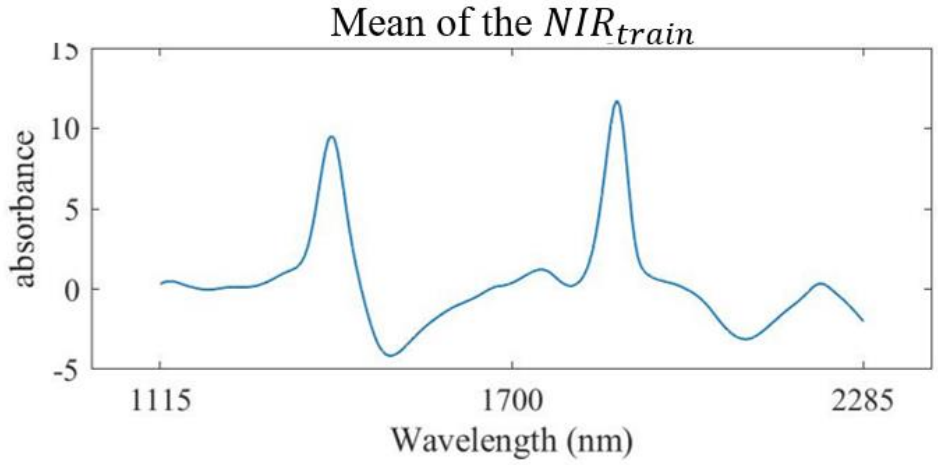
We want to build two regression models to predict ethanol content. We use the first 100 samples for training. So, the first 100 rows of "`NIR`", and the first 100 values of "`Ethanol`" are for training; and the rest (66 samples) are for the test.

Part 1: In this part, we want to use cubic B-splines for dimension reduction and feature extraction; and to build a linear regression model to predict ethanol content by performing the following steps:

a) Determine the optimal number of knots by the following step:

Estimate the mean of the NIR train (Column Mean(NIR_{train})) using cubic B-spline and vary the number of knots from 5 to 50. Choose the optimal number of knots by 5-fold cross-validation.

- Report the optimal number of knots and the cross-validation MSE.
- Plot your estimated mean of NIR_{train} using cubic B-spline along with the mean of the NIR train.



b) Use cubic B-spline with the optimal number of knots (you found in part a) to perform feature extraction on both the training and test sets. Develop a linear regression model to predict ethanol content based on the extracted features (B-spline coefficients) from the training data.
   - Report Residual Sum of Squares (RSS) for your model: $RSS_{Ethanol}$

c) Evaluate the performance of your prediction model on the test set.
   - Report $MSE_{Ethanol}$ on the test set.

Part 2: In this part, we want to use functional PCA for dimension reduction and feature extraction, and to build a linear regression model to predict ethanol content by performing the following steps:

a) First, smooth ("`smooth.basis`") the data using cubic B-spline basis with the same number of knots you found in Part 1a.
   - Plot the smoothed data (all the 166 samples).

b) Perform functional PCA on the dataset.
   - How many FPC-scores are required to explain 99% of variations?

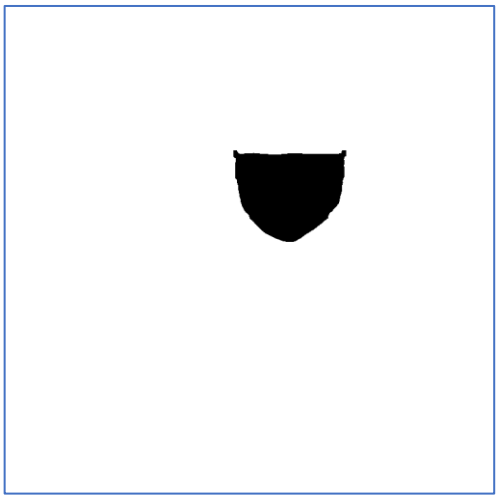
c) Develop a linear regression model to predict the ethanol content based on the scores on the principal components (or harmonics) of the training data.
   - Report Residual Sum of Squares (RSS) for your model: $\text{RSS}_{\text{Ethanol}}$

d) Evaluate the performance of your prediction model on the test set.
   - Report $\text{MSE}_{\text{Ethanol}}$ on the test set.

Question 2: (33 points)

COVID-19 pandemic poses a severe threat to human health. Wearing a face mask is one of the feasible ways to prevent the disease. Many places are required to wear a mask. Face mask detection techniques are needed to determine whether a person wears a mask or not in busy places. In this problem, we won't create a machine learning model to detect face masks automatically. Instead, we will play several image processing techniques, such as image transformation, image segmentation and edge detection, to highlight the face mask in an image. An image named "facemask.jpg" is provided.

1. Read the image, report the size of the image.
2. Cut the image into two pieces equally and keep the left half (with a person wearing a mask). The following analyses are based on the left half of the image.
3. Convert the image to grayscale. Show the grayscale image and its histogram.
4. Extract the face mask from the image by selecting the columns and rows of the image that contain the face mask. Use thresholding to convert your extracted image into black and white. Show the extracted face mask. See the example of an extracted face mask.
5. Use k-means algorithm to segment the image into k clusters where $k = 3{:}\,9$. Try to get the face mask cluster as accurate as possible. Highlight the face mask cluster and report the optimal k. Note: you may pre-process the image to make the face mask stand out before doing K-means clustering.
6. Use your Canny function for edge detection from Homework 2. Show the detected edges. The edge of the face mask should be detected clearly. See the example of edge detection.



Example of an extracted face mask

Example of edge detection

Question 3: (33 points)

Consider a set of multidimensional tensors $\underline{\boldsymbol{X}}^{(k)} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ for $k = 1, 2, \ldots, K$, representing training data coming from C classes. Each training sample $\underline{\boldsymbol{X}}^{(k)}$is given a label c_k indicating the category (class) to which it belongs, and a set of test data $\breve{\underline{\boldsymbol{X}}}^{(t)} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ for $t = 1, 2, \ldots, P$. The challenge is to find appropriate labels for the test data. The classification paradigm can be generally performed in the following steps:

1- Find the set of basis matrices and corresponding features for the training data $\underline{\boldsymbol{X}}$.
2- Perform feature extraction for test samples $\breve{\underline{\boldsymbol{X}}}$ using the basis factors found for the training data.
3- Perform classification by comparing the test features with the training features.

In general, a sample (object) is explained by N factors (basis matrices) $\boldsymbol{U}^{(n)} \in \mathbb{R}^{I_n \times J_n}$ for $n = 1,2, \ldots, N$ giving features represented by core tensors. We can assume that each basis factor $\boldsymbol{U}^{(n)}$ contains J_n components. The relation of a sample $\underline{\boldsymbol{X}}^{(k)}$ and N basis factors $\boldsymbol{U}^{(n)}$ can be expressed as:

$$\underline{\boldsymbol{X}}^{(k)} = \underline{\boldsymbol{G}}^{(k)} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \ldots \times_N \boldsymbol{U}^{(N)}$$

Where the compressed core tensor $\underline{\boldsymbol{G}}^{(k)} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_N}$ representing features is of a much lower dimension than the raw tensor $\underline{\boldsymbol{X}}^{(k)}$. In this question, consider $J_1 = \cdots = J_N = 10$. The pseudocode of the algorithm you are going to implement to find the set of basis matrices and the corresponding training and test features is given in Algorithm 1.

Algorithm 1: Algorithm for Feature extraction:

Input: $\underline{\boldsymbol{X}}$ concatenated tensor of K training samples $\in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times K}$
$\breve{\underline{\boldsymbol{X}}}$ concatenated tensor of P test samples$\in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N \times P}$

Output: $\boldsymbol{F}$: concatenated training feature matrix $\in \mathbb{R}^{(J_1 \times J_2 \times \ldots J_N) \times K}$
$\breve{\boldsymbol{F}}$: concatenated test feature matrix $\in \mathbb{R}^{(J_1 \times J_2 \times \ldots J_N) \times P}$

1: begin
2: Randomly initialize $\boldsymbol{U}^{(n)} \in \mathbb{R}_+^{I_n \times J_n}$
3: **for** iter $= 1{:}\,50$
4: **for** $n = 1{:}\,N$ **do**
5: $\underline{\boldsymbol{W}}^{(-n)} = \underline{\boldsymbol{X}} \times_{-(n,N+1)} \{\boldsymbol{U}^T\}$
6: $\boldsymbol{S}_w^{-n} = \langle \underline{\boldsymbol{W}}^{(-n)}, \underline{\boldsymbol{W}}^{(-n)} \rangle_{-n}$
7: $\left[\boldsymbol{U}^{(n)}, \sim\right] = \text{eigs}(\boldsymbol{S}_w^{-n}, J_n)$ % J_n leading eigenvectors
8: **end**
9: **end**
10: $\underline{\boldsymbol{G}} = \underline{\boldsymbol{X}} \times_{-(N+1)} \{\boldsymbol{U}\}^T$
11: $\boldsymbol{F} = \left[\text{vec}\left(\underline{\boldsymbol{G}}^{(k)}\right)\right]_{k=1}^{K}$
12: $\breve{\underline{\boldsymbol{G}}} = \breve{\underline{\boldsymbol{X}}} \times_{-(N+1)} \{\boldsymbol{U}\}^T$
13: $\breve{\boldsymbol{F}} = \left[\text{vec}\left(\breve{\underline{\boldsymbol{G}}}^{(k)}\right)\right]_{k=1}^{P}$

14: end

Part 1: Question3.A dataset consists of 1,440 grayscale images of 20 objects (72 images per object) with a variety of geometric and reflectance characteristics. Use the first 42 images of each object to construct a 3-D tensor of size $128 \times 128 \times 840$ for training, and the rest to construct a test tensor $128 \times 128 \times 600$. Apply Algorithm 1 to your training tensor to find the training features and the basis matrices. Build a multinomial SVM using your concatenated training feature matrix. Report the accuracy of your classifier on your concatenated test feature matrix. Present the confusion matrix.

Part 2: Question3.B dataset consists of 400 face images of 40 distinct subjects. Use the first 7 face images of each subject to construct a 3-D tensor of size $112 \times 92 \times 280$ for training, and the rest to construct a test tensor $112 \times 92 \times 120$. Apply Algorithm 1 to your training tensor to find the training features and the basis matrices. Build a multinomial SVM using your concatenated training feature matrix. Report the accuracy of your classifier on your concatenated test feature matrix. Present the confusion matrix.

Some basic tensor operations and notations

$\boldsymbol{X}_{(n)}$ mode-n matricization of $\underline{\boldsymbol{X}}$

$\mathbb{R}_{+}=\{x \in \mathbb{R} | x \geq 0\}$

$\underline{\boldsymbol{G}} \times \{\boldsymbol{U}\} = \underline{\boldsymbol{G}} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \ldots \times_N \boldsymbol{U}^{(N)}$

$\underline{\boldsymbol{G}} \times_{-n} \{\boldsymbol{U}\} = \underline{\boldsymbol{G}} \times_1 \boldsymbol{U}^{(1)} \ldots \times_{n-1} \boldsymbol{U}^{(n-1)} \times_{n+1} \boldsymbol{U}^{(n+1)} \ldots \times_N \boldsymbol{U}^{(N)}$

$\underline{\boldsymbol{G}} \times_{-(n,m)} \{\boldsymbol{U}\} = \underline{\boldsymbol{G}} \times_1 \boldsymbol{U}^{(1)} \ldots \times_{n-1} \boldsymbol{U}^{(n-1)} \times_{n+1} \boldsymbol{U}^{(n+1)} \ldots \times_{m-1} \boldsymbol{U}^{(m-1)} \times_{m+1} \boldsymbol{U}^{(m+1)} \ldots \times_N \boldsymbol{U}^{(N)}$

$\langle \underline{\boldsymbol{A}}, \underline{\boldsymbol{B}} \rangle_{-n} = \boldsymbol{A}_{(n)} \boldsymbol{B}_{(n)}^{T}$

$\left[\text{vec}\left(\underline{\boldsymbol{D}}^{(k)}\right)\right]_{k=1}^{Z} = \left[\text{vec}\left(\underline{\boldsymbol{D}}^{(1)}\right), \text{vec}\left(\underline{\boldsymbol{D}}^{(2)}\right), \ldots, \text{vec}\left(\underline{\boldsymbol{D}}^{(Z)}\right)\right]$