SE 165/265 Homework 6
Assigned May 8th, 2024; Due May 15th, 2024 (11:59 pm Pacific Time)

Assignment Goal:

1. Develop a data normalization approach based on the Mahalanobis Distance where the operational and environmental variability of the undamaged structure is encoded in the covariance matrix. The damage-sensitive features that will be used are the coefficients of an autoregressive model.

2. Reinforce material presented in Lecture 8 on using autoregressive model parameters as damage-sensitive features the material on data normalization presented in the lectures 11 and 12.

Background Reading: *Structural Health Monitoring: A Machine Learning Perspective,* Chapter *7.11 Time Series Models* and *Chapter 12: Data Normalization*

Introduction
We will continue to use experimental data from the 4-story structure. You can refer to Homework #1 to recall how the data were acquired. Recall from lecture 8 that autoregressive (AR) models are a type of linear time-series model that can be fit to our measured response data. The parameters (coefficients) of these models can be used as damage sensitive features. However, it is necessary to be able to distinguish changes in these features caused by damage from changes in these features when the data are acquired under varying environmental and operational (E&O) conditions.

With the 4-story structure, the first 9 states correspond to a system that is linear, but with different stiffness and mass values to simulate E&O variability one might encounter in real-world applications. We would expect the AR models that represent the data acquired from these various conditions to change as result of this variability. Damage that is introduced with the bumper mechanism in states 10-17 results in nonlinear system response somewhat analogous to crack opening and closing. Fitting AR models to data from these cases results in a model that is the best linear approximation to the data from the nonlinear system. Therefore, we would also expect the AR models to change as a result of this transition from a linear to nonlinear system, which results from the introduction of the simulated damage.

We need to develop a method that allows us to distinguish the AR coefficient obtained from data acquired on the undamaged structure from the coefficients obtained from data acquired from the damaged structure.

Task 1: Load data and create arrays (Identical to Task 1 of HW-5).

Download the file **data3SS2009.mat** from the SE 165-265 CANVAS Data File folder, 4-story structure data. **Load** this file in Matlab. This file contains measurements from all 5 sensors.

Loading this file will give you a 3-D matrix called **dataset** ($\mathbf{8192 \times 5 \times 850}$), where 8,192 corresponds to the data points, 5 corresponds to channels 1 to 5, and 850 is the number of measurements (50 measurements each for 17 states). Recall from HW 1 that states 1-9 are considered undamaged, and states 10-17 are damaged.

In this assignment, **we will work with data from all the channels: Channel 1** (shaker input), **Channels 2 - 5** (response data from the accelerometers on the floors 1-4).

Although not necessary, I first saved the data from channels 1-5 in new 8192x850 arrays called:

Channel 1 = **input**
Channel 2 = **sensor1**
Channel 3 = **sensor2**
Channel 4 = **sensor3**
Channel 5 = **sensor4**

We will not work with the input data on this assignment. <u>All the tasks below should be done for all the 4 sensors.</u>

<u>Task 2: Fit an AR model to all the response time-histories</u>.

For each sensor, fit AR(10) models to all time histories (all 850 measurements) and store them in a 850 x 10 matrix. Use the **lpc** command in Matlab (specify order 10 model, e.g. **a=lpc (x,10)**, where **x** is your sensor data).

<u>Note:</u> With the AR(10) model, the coefficients are estimate in the following manner:

$$\begin{bmatrix} x_1 & x_2 & \cdots & x \\ x_2 & x_3 & \cdots & x_{11} \\ \vdots & \vdots & \ddots & \vdots \\ x_{8182} & x_{8183} & \cdots & x_{8191} \end{bmatrix} \begin{Bmatrix} a_{10} \\ \vdots \\ a_2 \\ a_1 \end{Bmatrix} = \begin{Bmatrix} x_{11} \\ x_{12} \\ \vdots \\ x_{8192} \end{Bmatrix}$$

The **lpc** command estimates the AR coefficients by minimizing the following function:

$$(x_i - a_1 x_{i-1} - a_2 x_{i-2} - a_3 x_{i-3} - a_4 x_{i-4} - \cdots - a_{10} x_{i-10})^2$$

which is the square of the residual error.

Note that you will have to specify **x** as double precision using the **double** function in Matlab.

The LPC command returns a vector $a = [1, -a_1, -a_2, \ldots, -a_{10}]$ with 11 elements. The first value of one is the coefficient on the x_i term and should be eliminated using the command $a(1) = [\,]$.

Also, the AR coefficients are in reverse order and of opposite sign to the ones we would obtain by solving the least-squares problem shown in the matrix equation above. We won't worry about this ordering as we are only looking at the relative values of the coefficients.

Now we will generate some plots to get an idea of how the AR model coefficients vary at each sensor location. Remember, for all the tasks that requires **subplots**, when creating the figure (before the subplots), use:

```
      figure('Renderer', 'painters', 'Position', [10 10 1200 900])
```

<u>**Plot 1:**</u> For each sensor, overlay a plot of the AR model coefficients versus coefficient number for the first measurement of each <u>undamaged</u> state (rows 1, 51,101…401). These four plots should be displayed as a **2 x2 grid of subplots.** You do not need to create a legend for this figure. We just want to observe the general trend.

These plots should give an idea of how much the simulated E&O variability influences the AR models.

<u>**Plot 2:**</u> For each sensor, plot an overlay of the AR model coefficients from the first measurement of state 1 (row 1) and the first measurement of state 10 (row 451) (low damage level).

These plots should be displayed in **a 2 x 2 grid** of subplots. Add a **legend** to identify the AR coefficients corresponding to the two states. These plots should give an idea of how much the low level of damage influences the AR models. When compared to Plot 1, these plots give an idea of how much the influence of the low damage level has on the AR models relative to the influence of the environmental and operation variability.

<u>**Plot 3:**</u> For each sensor plot an overlay of the AR model coefficients from the first sample of state 1 (row 1) and the first sample of state 14 (row 651) (high damage level).

These plots should be displayed in a **2 x 2 grid**. Add a **legend** to identify the AR coefficients corresponding to the two states. These plots should give an idea of how much the high level of damage influences the AR models. When compared to Plot 1, these plots give an idea of how much the influence of the high damage level has on the AR models relative to the influence of the environmental and operation variability.

<u>Task 3: Develop training and testing AR coefficient data sets.</u>

For all 4 sensors, we will split our AR coefficients into training and test data sets. We will use data from our **undamaged** states to from a <u>training matrix</u>, and data from both our **undamaged and damaged** states to form a <u>testing matrix</u>, as follows:

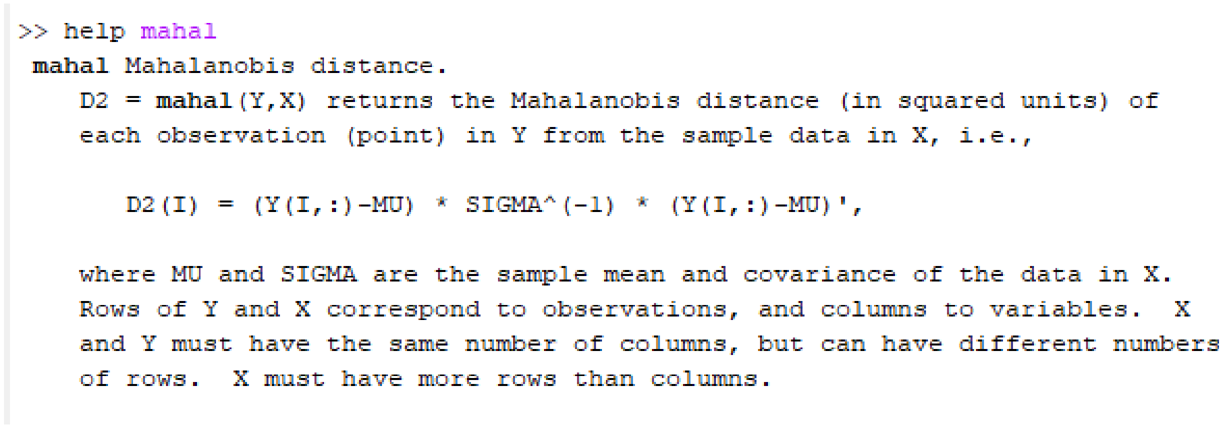
- Form a [225 x 10] training matrix [ARtrain] where each row is the AR coefficients obtained from only the **first 25 measurements** of each **undamaged** state.

- Form a [625 x 10] testing matrix [ARtest] where the first 225 rows are the AR coefficients obtained from the **second 25 measurements** of each **undamaged** state, and the last 400 rows are the AR coefficients obtained from **all the measurements** for each of the **damaged** states.

Task 4. Calculate the Mahalanobis distances (MD) for all the testing data based on the mean and covariance of the training data.

- For each sensor, use the Matlab function **mahal**(ARtest, ARtrain)

Note that with this process, the E&O variability has been encoded in the mean vector and covariance matrix of our training data (the **mahal** command does that automatically, as you can see below. You are not supposed to calculate the mean and covariance separately).



```
>> help mahal
 mahal Mahalanobis distance.
    D2 = mahal(Y,X) returns the Mahalanobis distance (in squared units) of
    each observation (point) in Y from the sample data in X, i.e.,

       D2(I) = (Y(I,:)-MU) * SIGMA^(-1) * (Y(I,:)-MU)',

    where MU and SIGMA are the sample mean and covariance of the data in X.
    Rows of Y and X correspond to observations, and columns to variables.  X
    and Y must have the same number of columns, but can have different numbers
    of rows.  X must have more rows than columns.
```

Plot 4: In a **2x2 grid,** create four bar plots (one for each sensor) using the MD values that you just calculated. Therefore, these plots should have 650 vertical bars.

When using the **bar** command to plot your data, specify the bar width to 1, as in **bar(data, 1)**, so that there is no gap between the bars being plotted.

In addition, on each plot, add a vertical red line that separates the undamaged cases (1-225) and damaged cases (226-625). Also, add a horizontal red line corresponding to the lowest value of all the damaged cases for that sensor (thus the horizontal line will vary for each plot, while the vertical one is fixed at 225). These lines will give an idea of how well separated the damage cases are from the undamaged cases.

Task 5. Calculate a Receiver-Operating Characteristic (ROC) Curve for the testing data.

Use the **perfcurve** function in Matlab. To use this function, you will have to create a vector of labels that identify which Mahalanobis distance values correspond to the undamaged condition (assigned a value of 0) and which values correspond to the damaged condition (assigned a value of 1). In a **2x2 grid,** create a plot of the ROC curve for each sensor.

For the ROC curves,

```
xlabel('False positive rate')
ylabel('True positive rate')
```

Don’t forget to add a title to your plots that indicate the content of the plot and the sensor number.