SE 165-265 Homework #9

Assigned May 29th, 2024; due June 5th, 2024 (11:59 pm Pacific Time)

Assignment Goal:

Develop a supervised learning classifier and compare it to an unsupervised learning classifier. Also, show that classification can be improved by choosing better features.

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. We will do so both in a supervised and an unsupervised learning manner.

Task 0: Download the function called `helperCommandWindowDisplay.m` from 'Canvas/Files/Homework Assignments.' It will be used later on.

Task 1: Download the file **data3SS2009.mat** from the 'Canvas/Files/Data & Codes/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 (8192×5×850)**, 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.

- Save the data using the following commands:

```
dataset=double(dataset);
testingData=squeeze(dataset(:,5,:));
```

The matrix testingData **(8192×850)** contains data from all 850 measurements.

- Create a time vector. The analog sensor signals were discretized with 8,192 data points sampled at 3.125 ms intervals corresponding to a sampling frequency of 320 Hz. These sampling parameters yield time histories of 25.6 s in duration. You will need this information to create your vector.

We will perform all the tasks below for two AR models: AR(5) and AR(30). Keep that in mind, as you can perform the HW in a loop/function.

Task 2.a: To demonstrate that improved features help the classification process, start by fitting two different time series models to the data. We will see that the AR(30) performs better.

- Fit AR(5) and AR(30) models to all 850 time histories *x(t)* in the **testingData** using the **lpc** command (i.e., a=lpc(x,5)).This process will result in two matrices $\mathbf{AR5}_{5x850}$ and $\mathbf{AR30}_{30x850}$. Do not use **pinv**, as it wil give you different results.

The LPC command returns a vector {a}={1, $-a_1$, $-a_2$, …$-a_{30}$). The first value of one is the coefficient on the x_i term and can be eliminated by the command a(1)=[ ].

Also, the AR coefficients are in reverse order and of opposite sign to what we need for calculating an estimate of the time history using Eq. 1.

We can change the order using the **flipud** command in Matlab and if we change signs, we will have vector that corresponds to {a} in Equation 1. E.g., {a from eq1} = -flipud(a).

Note: With the AR(30) model we are predicting the time series in the following manner:

$$\begin{bmatrix} x_1 & x_2 & x_3 & \dots & x_{30} \\ x_2 & x_3 & x_4 & \dots & x_{31} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{8162} & x_{8163} & x_{8164} & \dots & x_{8191} \end{bmatrix} \begin{Bmatrix} a_{30} \\ a_{29} \\ \vdots \\ a_1 \end{Bmatrix} = \begin{Bmatrix} x_{31} \\ x_{32} \\ \vdots \\ x_{8192} \end{Bmatrix} \quad (1)$$

Task 2.b: To verify that you have done the AR model parameter fit correctly, predict the first time history **(testingData (:,1))** using both the AR(5) model and the AR(30) models.

To do so, you will have to use **(testingData (:,1))** to create an $[X]$ matrix for the AR(5) model and for the AR(30) model. Recall that

$$x_{predicted} = [X]\{a\} \quad (2)$$

This is similar to what we did in HW 7, but we are creating the AR coefficients {a} using **lpc** instead of **pinv**.

Remember, you created the time vector in Task 1 ($\mathbf{time}_{1x8192}$).

- Generate two plots (one for the AR(5) model and one for the AR(30) model) overlaying the predicted and the measured time history for points 1000 – 1100. The result for the AR(5) data are shown in the hint below.

Use the same vertical axes scale for both plots to facilitate comparison.

Notice that for the AR(5) model you will only be able to predict points 6 to 8192, and for the AR(30) model you will only be able to predict points 31 to 8192. Account for that difference when generating the plots.

Hint: Result for AR(5).

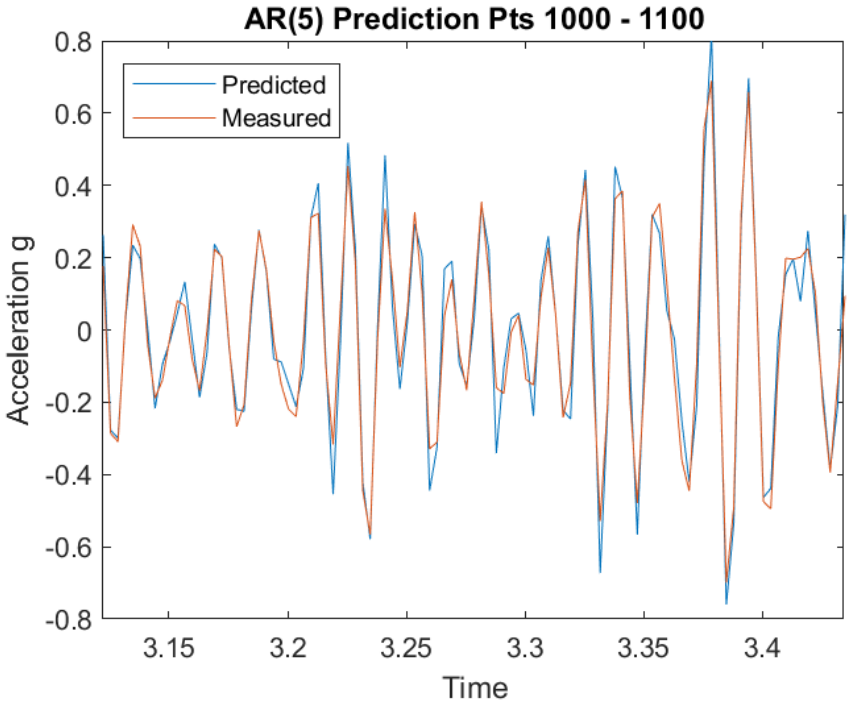

Figure 1- Overlay of the predicted time series using AR(5) and actual time series, for points 1000-1100.

Task 3.a. In this step you will examine some indicators of the AR(5) and AR(30) models' ability to accurately represent the time histories.

- Calculate the residual errors for the AR(5) and AR(30) model predictions of the first time history (**testingData (:,1)**), I'm using the names **re5**$_{8187x1}$ and **re30**$_{8162x1}$, respectively, for these residual error vectors. Recall that the residual error vector is given by

$$error = x\_measured_{31:8192} - x\_estimated \tag{3}$$

Remember, when we make this prediction of the time series, we will only be able to predict the last 8187 points ($Npoints - model\ order$) of the time series because we need the first 5 point to get an estimate of the 6th point. Consequently, we will only have 8187 estimates of the residual errors for the AR(5) model and 8162 for the AR(30).

- Plot the histograms of these residual errors. For comparison you need to plot the two histograms using the same number of bins, the same bin locations and the same bin widths. This plotting can be done fairly easily with the **histogram** command.

  First, generate a 90-bin histogram of the residual errors for the AR(5) model predictions using the function form **h=histogram(re5,90)**. This function produces an object **h** that will contain a vector **h.BinEdges.**

  Then, on the second plot, specify this vector as an argument in the histogram function for the AR(30) model. E.g., **histogram(re30, h.BinEdges)**, will produce a histogram with the same bins as those generated for the AR(5) model.

  Use the same horizontal and vertical axis for these two plots.

  You will see that the AR(5) model has a larger variance (distribution width) than the AR(30) model, indicating that the fit to the actual time series is not as good as the AR(30) model fit.

Task 3.b. Calculate and plot the power spectral density (PSD) of the residual errors for the AR(5) and AR(30) model predictions of the first time history (**testingData (:,1)**).

Use the **pwelch** function with a hanning window ('hann'), 8 averages and zero overlap. You know that the sampling frequency is 320 Hz.

Create the plots using 'plot'. We will not use 'semilogy' this time, as using a linear scale will emphasize the amplitudes.

Plot the two PSDs on the same vertical scale. If the model has fit the data well, you should see a uniform (flat) PSD. Neither of the two models produces a completely flat PSD. However, you should see that the amplitude of the PSD is much larger for the **re5** values and you should see three distinct peaks in the re5 values, which are indicators that the AR(5) model is not doing as good of a job predicting the time history compared to the AR(30) model. The result for the AR(5) model is shown in Figure 3.

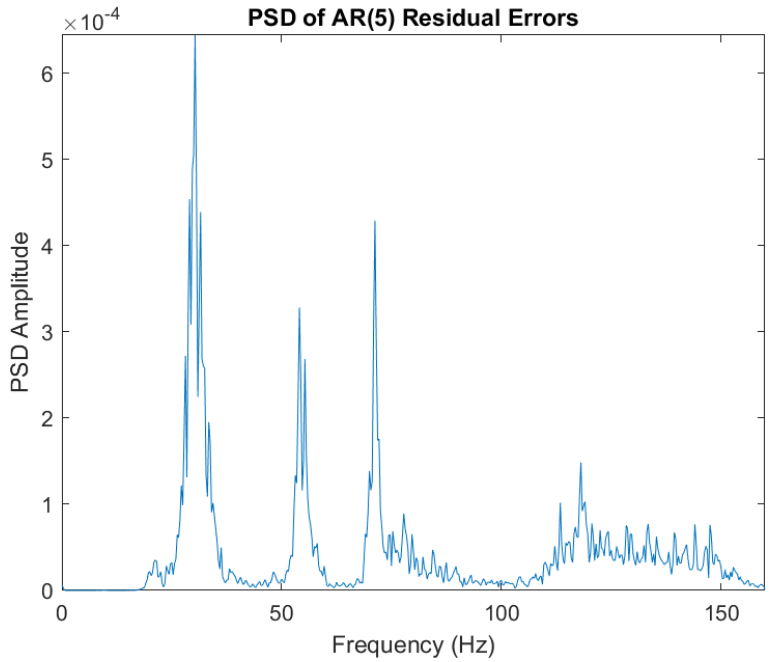

Figure 3- Power spectral density estimate of the residual errors of the AR(5) model.

Task 4a. Unsupervised learning.

Recall from HW 1 that states 1-9 are considered undamaged, and states 10-17 are damaged.

Divide the AR(5) model parameters created on Task 2 (i.e., the **AR5**$_{5x850}$ matrix) into training and testing data sets, as follows. You do not need to follow the naming convention, but we will use this for consistency.

1. **Train_u5**$_{5x225}$: A training set of the AR(5) model parameters that correspond to the odd-numbered undamaged cases (cases 1, 3, 5, …449).
2. **Test_u5**$_{5x225}$: A testing set of AR(5) model parameters corresponding to even undamaged cases (cases 2, 4, 6, …450).
3. **Test_d5**$_{5x400}$: A testing set of the AR(5) model parameters that correspond to the all damaged cases (cases 451, 452, 453, …850).
4. **Train_d5**$_{5x200}$: A training set of the AR(5) model parameters that correspond to the odd-numbered damaged cases (cases 451, 453, …849)**.**
5. **Test_svm_d5**$_{5x200}$: A testing set of the AR(5) model parameters that correspond to the even-numbered damaged cases (cases 452, 454, 456, …850).

Task 4b. Repeat Task 4a. for the AR(30) model parameters created on Task 2 (i.e., the **AR30**$_{30x850}$ matrix) using the same names, with 5 replaced by 30 in the respective names.

Task 5.a. Perform unsupervised learning outlier detection using the squared Mahalanobis distance metric.

Background: The first matrix in the arguments to the Mahal function contains the feature vectors (row vectors after you take the transpose) whose Mahalanobis distance you want to calculate. The second matrix in the argument list contains the row vectors of the features you want to use as your reference (the mean vector and covariance matrix in the squared Mahalanobis distance formulation will be based on the reference data).

Because we are simulating an unsupervised learning problem, we will assume we only have the undamaged data available for the reference, which means we will use **Train_u5** (or **Train_u30**) as the reference data for all the squared Mahalanobis distances that we will calculate below.

Now execute the following steps in Matlab for both the AR(5) model and the AR(30) model:

- Calculate the squared Mahalanobis distances for the undamaged training data **Train_u5** (or **Train_u30**) using the function **mahal**. For this case you will specify the same matrices for both arguments to the function. Note that the **mahal** command requires the feature vectors to be row vectors, so you will have to use the transpose of the respective testing and training matrices defined in the previous step.

```
MD_trainu5=mahal(Train_u5',Train_u5');
```

- Plot the histogram of the squared Mahalanobis distances corresponding to the undamaged training data (Train_u5 and Train_u30). Use **histfit** and specify 30 bins.
- Add vertical lines to these histograms corresponding to the **three standard deviations from the mean** of the Mahalanobis squared distance values.
- Use the same y-axis for both plots.

Task 5.b. Outlier detection. Perform all steps for both AR(5) and AR(30).

- Calculate the squared Mahalanobis distances for the **undamaged** testing data **Test_u5** (**or Test_u30**). For this case you will specify the **Test_u5** (**or Test_u30**) as the first argument to the mahal function, and **Train_u5** (**or trainu30**) as the second argument (reference data).
- Calculate the number of **undamaged** testing values that lie outside the upper 3-sigma confidence intervals calculated during step 5.a. and name it **FP_u5** (or **FP_u30**). These are the false-positives. Note we are only interested in the ones that lie outside of the upper 3-sigma bound.
- Now, calculate the squared Mahalanobis distances for all the **damaged** features vectors. For this case you will specify the **Test_d5** (or **Test_d30**) as the first argument to the **mahal** function and **Train_u5** (**or Train_u30**) as the second argument (reference data).

- Calculate the number of **damaged** testing values that lie inside the 3-sigma confidence intervals and name it **FN_d5** (or **FN_d30**). These are false-negatives. Note we are only interested in the ones that are below the upper 3-sigma bound.
- Knowing that the actual undamaged cases are 225, and that the actual quantity of damaged cases is 400, calculate the number of true positives and true negatives.
- Display the confusion matrices in the command window for the AR(5) and AR(30) unsupervised learning classifiers using the provided function called `helperCommandWindowDisplay.m`

To do so, just use the following command with the appropriate inputs to it:

```
helperCommandWindowDisplay(MLtype,nAR,TrueNegative,FalseNegative,TrueP
ositive,FalsePositive)
```

Where the inputs are:

MLtype is 0 for unsupervised learning, and 1 for supervised (next task)
nAR is 5 or 30, depending on the model
TrueNegative is **TN_5**
FalseNegative is **FN_d5**
TruePositive is **TP_5**
FalsePositive is **FP_u5**

The function will display the confusion chart, as well as the following:

True Positive Rate: the percent true positives relative to all 400 damage testing cases

True Negative Rate: the percent true negatives relative to all 225 undamaged testing cases

False Positive Rate: the percent of false negative relative to all 225 undamaged testing cases

False Negative Rate: the percent of false negative relative to all 400 damaged testing cases

Overall Classification Accuracy: the percent of the sum of true positive plus true negatives relative to the sum of all 625 damaged and undamaged testing cases.

After performing Tasks 5 for the AR(5) model, your command window should display the following results:

Unsupervised Learning Confusion Matrix AR(5) Model

	Actual Undamaged	Actual Damaged	Total
Predicted undamaged	222	145	367
Predicted damaged	3	255	258
Total	225	400	NaN

True Positive (damage) Rate: 63.8%
True Negative (undamaged) Rate: 98.7%
False Positive Rate: 1.3%
False Negative Rate: 36.2%
Overall Classification Accuracy: 76.3%

Don't forget to calculate and display the results for the AR(30) model as well.

Task 6. Support Vector Machine.

Perform supervised learning outlier detection with a support vector machine (SVM) classifier using the AR model parameters as the damage-sensitive feature.

Do this outlier detection for both the AR(5) and AR(30) features. Implement the support vector machine classifier with a linear kernel function using the following steps:

1. Combine undamaged training data **Train_u5** (or **Train_u30**) and damaged training data **Train_d5** (or **Train_d30**) into one matrix, **svm_train5**$_{5x425}$=[**Train_u5**, **Train_d5**] (or **svm_train30**).

2. Combine undamaged testing data **Test_u5** (or **Test_u30**) and damaged testing data **Test_svm_d5** (or **Test_svm_d30**) into one matrix, **svm_test5**$_{5x425}$=[**Test_u5**, **Test_svm_d5**] (or **svmtest30**).

Pay attention to the naming conventions and make sure to follow what we created in Task 4. **Test_svm_d5** only contain 200 cases, as the other 200 were used for training.

3. Create a binary classification label row vector (**Class1**$_{1x425}$) with the first 225 columns having a value of zero (0 = undamaged) and the next 200 columns having a value of one, (1=damaged). This is how we will specify the class labels to the SVM classifier.

4. Train the svm classifier using the **fitclinear** function with the transpose of **svmtrain5** (or **svmtrain30**) and the **Class1** specified as the two arguments for this function.

5. Classify the testing data using the **predict** function, with the output of the **fitclinear** function followed by the transpose of **svm_test5** or (**svm_test30**) as the arguments for this function.

The output of predict should be a vector of 0s and 1s, which are the class labels (0 = undamaged, 1=damaged) for all the vectors in the testing data.

6. Calculate the true positives, false positives, true negatives and false negatives and prediction accuracies by comparing the actual classification labels (Class1) with the predicted classification labels.

   Notice that this time we have 225 undamaged cases and 200 damaged cases.

7. Display these quantities using the same function provided:

```
helperCommandWindowDisplay(MLtype,nAR,TrueNegative,FalseNegative,
TruePositive,FalsePositive)
```

   Now, MLtype should be set to 1, as it is a supervised learning method.

Don't forget to calculate and display the results for both AR(5) and AR(30) models.

You should see two general trends in these analyses:

1. In the general, overall classification rate improves when the AR(30) features are used compared to the AR(5) features.
2. The supervised learning method provided better overall classification than the unsupervised method.