Homework 3

SM2001 : Data-Driven Methods in Engineering

Kirtan Premprakash Patel

Problem 1

We are given a 1001x2 dataset, which we first use to calculate the derivative using the difference method using the 1001x1 time data. These derivatives are found between two datapoints, hence, we find the midpoints of the given datapoints to construct a smaller 1000x2 dataset which we consider to be the datapoints where the derivatives are calculated.

We then use the 1000x2 midpoints dataset and 1000x2 derivatives calculated to identify the non0linear dynamics using the SINDy methodology.

Given that the order of dynamics is less than or equal to 3, we use it to construct library of functions i.e. Theta.

Command Window :

	x1dot	x2dot
1	0.49916	0
x	0.99973	0.20003
y	-0.99896	-0.16002
xx	0	0
xy	0	0
yy	0	0
xxx	-0.33318	0
xxy	0	0
xyy	0	0
yyy	0	0

Thus, we have the required coefficients

$$
\begin{aligned}
c_{0,0} &= 0.49916 \\
c_{1,0} &= 0.99973 \\
c_{0,1} &= -0.99896 \\
c_{3,0} &= -0.33318 \\
d_{1,0} &= 0.20003 \\
d_{0,1} &= -0.16002
\end{aligned}
$$

This agrees with the hint provided that we should be expecting to find a total of 6 non-zero coefficients.

Problem 2

Multi-Class Classification

Multiclass classification is a type of machine learning task in which the goal is to assign input data points to one of multiple predefined classes or categories. This is a common problem in various fields, including image recognition, natural language processing, and medical diagnostics. Multiclass classification and compare supervised and unsupervised methods used for solving this task, including dendrograms, K-means clustering, and neural networks.

Supervised methods for multiclass classification, like Neural Networks, involve training a model on a labeled dataset, where each data point is associated with a specific class label. The model learns to map input features to class labels based on the training examples. Unsupervised methods for multiclass classification, like K-means Clustering and Hierarchical Clustering, do not rely on labeled data but instead aim to group data points into clusters based on similarities in their features. These clusters can then be interpreted as classes.

Supervised Methods vs Unsupervised Methods

Supervised Methods posses various advantages such as

1. their ability to capture complex patterns in data (as seen in the executed example as well)
2. ability to effectively handle high-dimensional input features

They also have disadvantages such as

1. require large amount of labeled data for training
2. prone to over-fitting, especially with limited data
3. require high computational resources
4. complex architecture which might be difficult to interpret

Unsupervised Methods on the other hand have advantages like

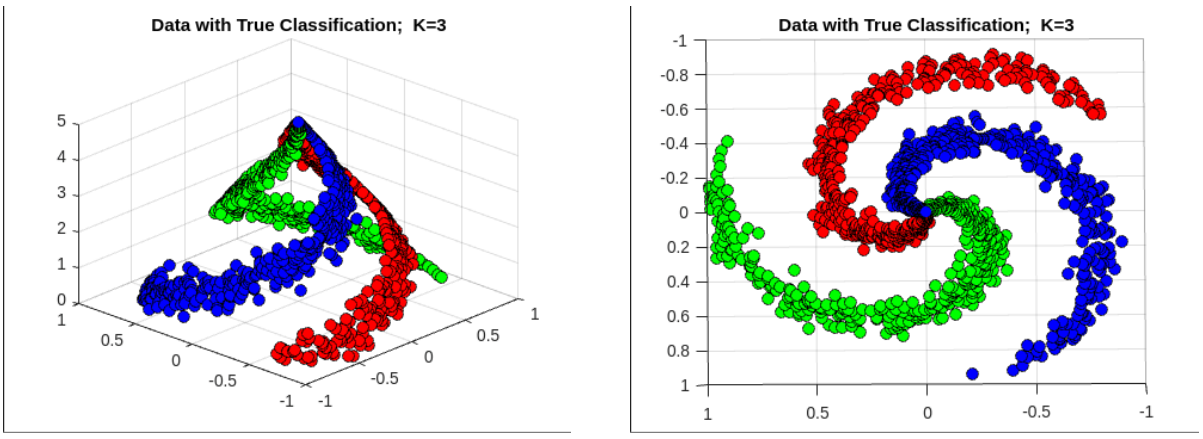
1. scalability and computational efficiency
2. no need for pre-defined labels
3. Hierarchical clustering further does not need the number of clusters to be pre-defined.

but posses disadvantages like

1. might not always give optimal classification
2. ineffecient classification in high-dimensional non-linear dataset
3. Sensitive to the choice of distance metric, linkage method and initial placement of cluster centroids
4. Assumes spherical and equally sized clusters, which may not reflect the true data distribution

Labeled Dataset

The labelled dataset used to compare the methods is self-generated. It is an upward spiral with 3 branches.

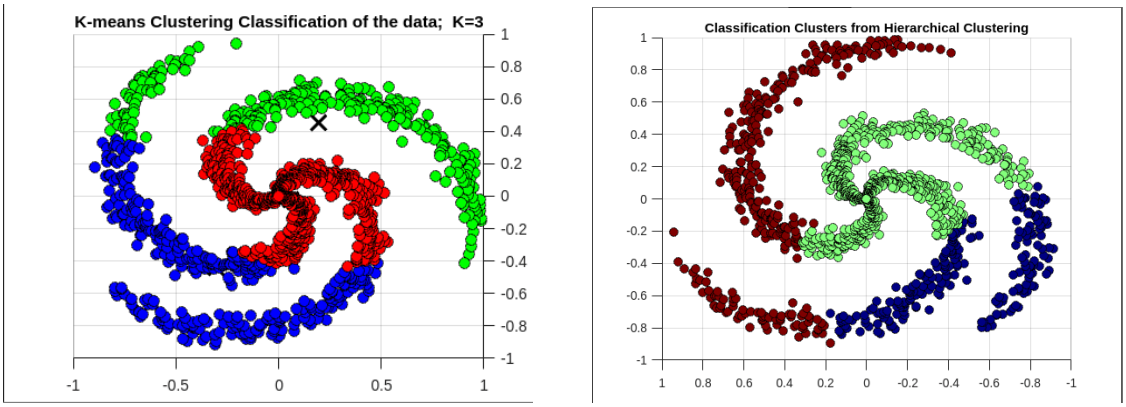


(a) (b)

Figure 1: Self-generated Labelled Dataset

The data contains the label in the first column, and the X, Y, and Z co-ordinated in the following columns. Analysis using Supervised methods has been done in the Kaggle Notebook itself, in which the data is generated. This data has then been imported to MATLAB for further analysis using Unsupervised Methods.

K-means Clustering and Hierarchical Clustering

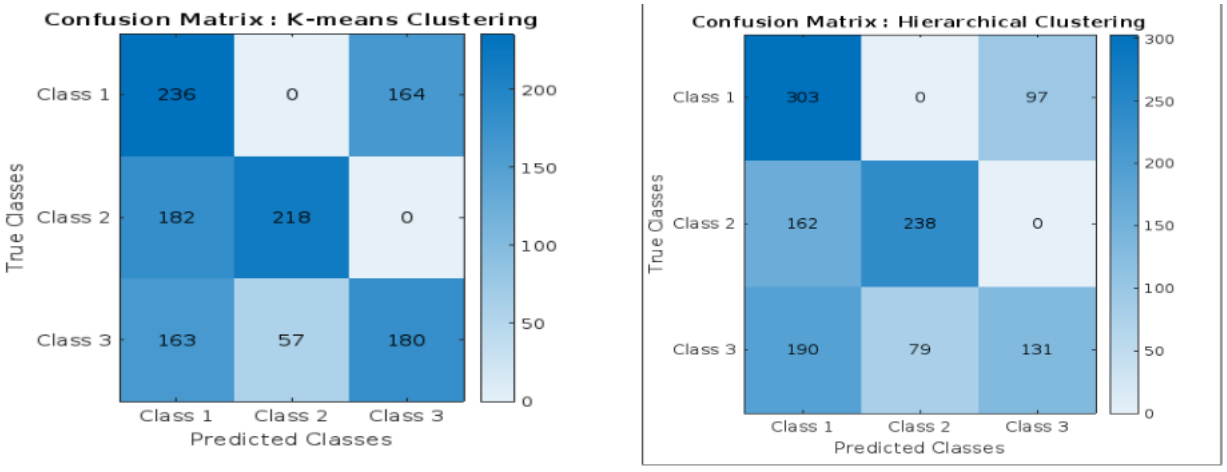


(a) (b)

Figure 2: Classification using Unsupervised Clustering

Evaluation of Clustering Classification

As seen, there is significant lack of accuracy in unsupervised clustering models when used for classification.



(a) (b)

Figure 3: Evaluation of Clustering Classification using Confusion Matrix

Unsupervised learning tends not to detect complex non-linear patterns in the data and provides no visual clarity for the reason behind the clustering in this case. Furthermore, the accuracy can not be improved with more computation.

Unsupervised Learning

On the other hand Unsupervised Methods like Neural Network can identify the underlying complex patterns. Furthermore since it uses the provided label on the data, it can be trained to ensure that it achieves 100% accuracy by increasing the number of epochs (or the number of hidden layers). Selecting an optimal training size is crucial to ensure there is no overfitting. Table 1 illustrates both these characteristics.

Number of Epochs	Accuracy		
	Test Size = 0.2	Test Size = 0.1	Test Size = 0.05
5	0.7917	0.8750	0.750
10	0.8917	0.9167	0.8667
15	0.9458	0.9667	0.9500
25	0.9917	1.0000	0.9833
30	1.0000	1.0000	1.0000

Table 1: Effect of Number of Epochs and Size of Training Set on Accuracy

APPENDIX A

MATLAB Codes

Problem 1

```
clear all
close all
clc

%% Prepare the data
X = load('DataHw3Q1.mat').x;
t = load('DataHw3Q1.mat').t;
n = size(X,2);      % state dimension i.e. number of states

dt = t(2) - t(1);
dx_dt = diff(X) / dt;   % slope between 2 points, NOT at
x = (X(1:end-1,:) + X(2:end,:))/2; % datapoints at midpoint
% we will form SINDy using x and dx_dt


%% Build library and compute sparse regression
order = 3;
Theta = poolData(x,n,order,0); % up to third order polynomials
% Theta for order 3 gives the poolData as (x = x1, y = x2)
% [1, x, y, x*x, x*y, y*y, x*x*x, x*x*y, x*y*y, y*y*y]
% column 1 contains variables c, and column 2 contain variables d
% [c0,0 c1,0 c0,1 c2,0 c1,1 c0,2 c3,0 c2,1 c1,2 c0,3]
% [d0,0 d1,0 d0,1 d2,0 d1,1 d0,2 d3,0 d2,1 d1,2 d0,3]
lambda = 0.025;
% lambda is our sparsification knob.
Xi = sparsifyDynamics(Theta,dx_dt,lambda,n);
%disp(Xi);

rowNames={'1','x','y','xx','xy','yy','xxx','xxy','xyy','yyy'};
colNames = {'x1dot','x2dot'};
sTable = array2table(Xi,'RowNames',rowNames,'VariableNames',colNames);
disp(sTable);
```

Problem 2 : Unsupervised Learning

```
clear all
close all
clc

%% Load the data
labelledData = load('labelledNonLinearDataset.mat').data;
% 1200x4 data matrix
% Column1 : labels
% Column2 : X coordinates
% Column3 : Y coordinates
% Column4 : Z coordinates

label = labelledData(:,1);  % this is the true classification
X = labelledData(:,2:4);

% number of true classes in the data
K = 3;

% Plot the data with tru label
colors = ['r', 'g', 'b', 'c', 'm', 'y', 'k'];
figure;
hold on;
for i = 1:K
    cluster_points = X(label == i, :);
    scatter3(cluster_points(:, 1), cluster_points(:, 2),
       cluster_points(:, 3), 50, colors(i),'filled', 'DisplayName', ['
       Cluster ',num2str(i,'%d') ], 'MarkerEdgeColor', 'k');
end
colormap('jet');  % Set the colormap as needed
title(['Data with True Classification;  K=',num2str(K,'%d')]);
%legend('Location', 'Best');
grid on;
hold off;

% Set the perspective view
view(-30, -30); % Azimuth: 30 degrees, Elevation: 30 degree

%% Normalize dataset
mean = mean(X);
stdDev = std(X);
zNormalData = normalize(X);
%X_reconstructed = zNormalData.*stdDev + mean;


%% Part (a) : K-means clustering
K = 3;
numInitializations = 1000;       % define the number of random
   initializations
choiceMultiple = 10;
% random centroids generated are numInitializations*choiceMultiple
% this is done to further increase randomness by using datasample
paddedNumInits = choiceMultiple*numInitializations;

% Variables to store the results
bestIdx = [];
bestCentroids = [];
bestWCSS = Inf;
randomCentroids = [randn(paddedNumInits, 1), randn(paddedNumInits, 1),
    randn(paddedNumInits, 1)];
```

```
56
57  for i = 1:numInitializations
58      % Randomly initialize cluster centroids
59      initialCentroids = datasample(randomCentroids, K, 'Replace', false
            );
60
61      % Perform K-means clustering with these initial centroids
62      [idx, centroids, sumd] = kmeans(zNormalData, K, 'Start',
            initialCentroids);
63
64      % Calculate the total within-cluster sum of squares (WCSS)
65      wcss = sum(sumd);
66
67      % Check if this initialization results in a lower WCSS
68      if wcss < bestWCSS
69          bestIdx = idx;
70          bestCentroids = centroids;
71          bestWCSS = wcss;
72          %disp(bestWCSS);
73      end
74  end
75
76
77  % Plot the best clustered data
78  upscaledCentroids = bestCentroids.*stdDev + mean;
79  colors = ['r', 'g', 'b', 'c', 'm', 'y', 'k'];
80  figure;
81  hold on;
82  for i = 1:K
83      cluster_points = X(bestIdx == i, :);
84
85      scatter3(cluster_points(:, 1), cluster_points(:, 2),
            cluster_points(:, 3), 50, colors(i),'filled', 'DisplayName', ['
            Cluster ',num2str(i,'%d') ], 'MarkerEdgeColor', 'k');
86  end
87  colormap('jet');  % Set the colormap as needed
88  scatter3(upscaledCentroids(:, 1), upscaledCentroids(:, 2),
        upscaledCentroids(:, 3), 200, 'k', 'Marker', 'x', 'LineWidth',2, '
        DisplayName', 'Centroid');
89  title(['K-means Clustering Classification of the data;  K=',num2str(K,
        '%d')]);
90  %legend('Location', 'Best');
91  grid on;
92  hold off;
93
94  % Set the perspective view
95  view(-30, -30); % Azimuth: -30 degrees, Elevation: -30 degrees
96
97
98  % Create Confusion Matrix to check Classification of Non-Linear
        Dataset
99  trueClassifications = label;
100 predictedClassifications = bestIdx;
101
102 % Create a confusion matrix
103 C = confusionmat(trueClassifications, predictedClassifications);
104
105 % Display the confusion matrix
106 disp('Confusion Matrix:');
107 disp(C);
108
```

```
initial_diagonal_sum = sum(diag(C));

% Generate all possible permutations of row indices and column indices
n = size(C, 1); % Assuming a square matrix
row_permutations = perms(1:n);
col_permutations = perms(1:n);

max_diagonal_sum = initial_diagonal_sum;
best_row_permutation = 1:n;
best_col_permutation = 1:n;

for i = 1:size(row_permutations, 1)
    for j = 1:size(col_permutations, 1)
        B = C(row_permutations(i, :), col_permutations(j, :));
        diagonal_sum = sum(diag(B));
        if diagonal_sum > max_diagonal_sum && B(1,1) > B(2,2) && B
            (2,2) > B(3,3)
            max_diagonal_sum = diagonal_sum;
            best_row_permutation = row_permutations(i, :);
            best_col_permutation = col_permutations(j, :);
        end
    end
end

% Reorder the best row and column permutation
C_mod = C(best_row_permutation, best_col_permutation);
disp('Matrix with Maximized Diagonal:');
disp(C_mod);

% Define the minimum and maximum values for the color scale
min_val = min(C_mod(:));  % Set the minimum value
max_val = max(C_mod(:)); % Set the maximum value

% Create a figure
figure(3);
set(gcf, 'Position', [100, 100, 400, 400]);

% Create the heatmap
imagesc(C_mod, [min_val, max_val]);
colormap('sky');

% Add labels and colorbar
title('Confusion Matrix : K-means Clustering');
xlabel('Predicted Classes');
ylabel('True Classes');
set(gca, 'XTick', 1:size(C_mod, 2));
set(gca, 'YTick', 1:size(C_mod, 1));

% Manually set the color scale limits
clim([min_val, max_val]);

% Display the values in each cell
textStrings = num2str(C_mod(:),'%d');
textStrings = strtrim(cellstr(textStrings));
[x, y] = meshgrid(1:size(C_mod, 2), 1:size(C_mod, 1));
hStrings = text(x(:), y(:), textStrings(:), 'HorizontalAlignment', '
    center');
colorbar;

set(gca, 'XTickLabel', {'Class 1', 'Class 2', 'Class 3'});
set(gca, 'YTickLabel', {'Class 1', 'Class 2', 'Class 3'});
```

```
168
169
170 %% Part (b) : Hierarchical clustering
171
172 % Perform hierarchical clustering
173 Z = linkage(zNormalData, 'ward'); % You can choose different linkage
      methods
174
175 %{
176 % Visualize the hierarchical clustering using a dendrogram
177 figure;
178 dendrogram(Z);
179
180 title('Hierarchical Clustering Dendrogram');
181 xlabel('Data Points');
182 ylabel('Distance');
183 %}
184
185 % Use Hierarchical Clustering to form Clusters
186 T = cluster(Z,"maxclust",3);
187
188 % Display the clusters using different colors in a scatter plot
189 figure;
190 scatter3(X(:, 1), X(:, 2), X(:, 3), 50, T, 'filled', 'MarkerEdgeColor'
      , 'k');
191 title('Classification Clusters from Hierarchical Clustering');
192 colormap('jet');
193
194 % Set the perspective view
195 view(-30, -30); % Azimuth: 30 degrees, Elevation: 45 degrees
196
197
198 % Create Confusion Matrix to check Classification of Non-Linear
      Dataset
199 trueClassifications = label;
200 predictedClassifications = T;
201
202 % Create a confusion matrix
203 C2 = confusionmat(trueClassifications, predictedClassifications);
204
205 % Display the confusion matrix
206 disp('Confusion Matrix:');
207 disp(C2);
208
209 % But this might lead to false inference since the numbering of the
210 % newly formed clusters can be changed to ensure that the number
211 % assigned to the cluster is such that it maximizes diagonal elements.
212 % i.e. we can get better inference by row and column swapping
213
214 % Calculate the sum of diagonal elements
215 initial_diagonal_sum = sum(diag(C2));
216
217 % Generate all possible permutations of row indices and column indices
218 n = size(C2, 1); % Assuming a square matrix
219 row_permutations = perms(1:n);
220 col_permutations = perms(1:n);
221
222 max_diagonal_sum = initial_diagonal_sum;
223 best_row_permutation = 1:n;
224 best_col_permutation = 1:n;
225
```

```
% Iterate through all row and column permutations to find the best
    diagonal sum
for i = 1:size(row_permutations, 1)
    for j = 1:size(col_permutations, 1)
        B = C2(row_permutations(i, :), col_permutations(j, :));
        diagonal_sum = sum(diag(B));
        if diagonal_sum > max_diagonal_sum && B(1,1) > B(2,2) && B
            (2,2) > B(3,3)
            max_diagonal_sum = diagonal_sum;
            best_row_permutation = row_permutations(i, :);
            best_col_permutation = col_permutations(j, :);
        end
    end
end

% Reorder the best row and column permutation
C_mod = C2(best_row_permutation, best_col_permutation);
disp('Matrix with Maximized Diagonal:');
disp(C_mod);


% Define the minimum and maximum values for the color scale
min_val = min(C_mod(:));  % Set the minimum value
max_val = max(C_mod(:)); % Set the maximum value

% Create a figure
figure(5);
set(gcf, 'Position', [100, 100, 400, 400]);

% Create the heatmap
imagesc(C_mod, [min_val, max_val]);
colormap('sky');

% Add labels and colorbar
title('Confusion Matrix : Hierarchical Clustering');
xlabel('Predicted Classes');
ylabel('True Classes');
set(gca, 'XTick', 1:size(C_mod, 2));
set(gca, 'YTick', 1:size(C_mod, 1));

% Manually set the color scale limits
clim([min_val, max_val]);

% Display the values in each cell
textStrings = num2str(C_mod(:),'%d');
textStrings = strtrim(cellstr(textStrings));
[x, y] = meshgrid(1:size(C_mod, 2), 1:size(C_mod, 1));
hStrings = text(x(:), y(:), textStrings(:), 'HorizontalAlignment', '
    center');
colorbar;

% Adjust axis settings for better display
set(gca, 'XTickLabel', {'Class 1', 'Class 2', 'Class 3'});
set(gca, 'YTickLabel', {'Class 1', 'Class 2', 'Class 3'});
```

Problem 2 : Supervised Learning

In [1]:
```
import pandas
import numpy as np
from tqdm import tqdm
import matplotlib.pyplot as plt
from scipy.io import savemat, loadmat
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import to_categorical
from sklearn.model_selection import train_test_split
```

In [2]:
```
def generate_spirals():
    N = 400 # number of points per class
    D = 3 # dimensionality
    K = 3 # number of classes
    Z_max = 5 # max height
    data = np.zeros((N*K,D)) # data matrix (each row = single example)
    label = np.zeros(N*K, dtype='uint8') # class labels
    for j in range(K):
        ix = range(N*j,N*(j+1))
        r = np.linspace(0.0,1,N) # radius
        t = np.linspace(j*4,(j+1)*4,N) + np.random.randn(N)*0.2 # theta
        z = np.linspace(Z_max,0.0,N) # radius
        data[ix] = np.c_[r*np.sin(t), r*np.cos(t), z]
        label[ix] = j      # so that classification is from 1 onwards.
    # Create a 3D scatter plot
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
    ax.scatter(data[:, 0], data[:, 1], data[:, 2], c=label, s=40)

    ax.legend()

    plt.show()
    return [data, label]
```

In [3]:
```
class MultiClassNN:
    def __init__(self, input_dim, num_classes, hidden_units=(64, 32),
    activation='relu', optimizer='adam'):
        self.input_dim = input_dim
        self.num_classes = num_classes
        self.hidden_units = hidden_units
        self.activation = activation
        self.optimizer = optimizer

        self.model = self._build_model()

    def _build_model(self):
        model = Sequential()
        model.add(Dense(self.hidden_units[0], input_dim=self.input_dim,
    activation=self.activation))
        for units in self.hidden_units[1:]:
            model.add(Dense(units, activation=self.activation))
        model.add(Dense(self.num_classes, activation='softmax'))
        model.compile(loss='categorical_crossentropy',
    optimizer=self.optimizer, metrics=['accuracy'])
        return model

    def predict(self, X):
        return np.argmax(self.model.predict(X), axis=1)

    def evaluate(self, X_test, y_test):
        y_pred = self.predict(X_test)
        accuracy = np.mean(y_pred == y_test)
        return accuracy
```

In [4]:

```
X = dataset
y = labels
input_dim = X.shape[1]
num_classes = len(np.unique(y))
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.05,
    random_state=42)

# Initialize the neural network classifier
Classifier = MultiClassNN(input_dim, num_classes)
```

In [5]:

```
epochs = 30

# Train the Classification NN Model
print("Training the MultiClass Classification NN Model")
progress = tqdm(range(epochs))
accuracyRecords = np.zeros(epochs)

y_categorical = to_categorical(y_train, num_classes=Classifier.num_classes)

for epoch in progress:
    Classifier.model.fit(X_train, y_categorical, epochs=1, verbose = 0)

# Evaluate the classifier
print("\nEvaluting Classification Model with Test Data")
accuracy = Classifier.evaluate(X_test, y_test)
print("Result after training is completed")
print(f"Accuracy: {accuracy}")
```