Project 3 part 2

Gordon Dina

I. Introduction

Given a data set containing 16 human activities, preformed by 10 human subjects, activity done twice: this data set contains both color depth and skeleton data. For this project the skeleton data will be used. The skeleton data is in rows containing 5 values: frame id, joint id, joint position x, joint position y, and joint position z. This data will be used to construct a human representations based on the Relative Angles and Distances (RAD) of star skeleton. RAD consists of the joint's distance from the center of the skeleton as well the angles between each joint. For this part of the project the histogram data that was obtained from previous part will be used with Support Vector Machines (SVM) to enable robot learning, producing accuracy and precision of the data as well compute a Confusion Matrix to evaluate the accuracy of a classification.

II. featurization methods and their respective histograms

Data was collected came from two different sources: train and test. Each folder contained text files of 6/16 of the human activities that were collected (activities 8, 10, 12, 13, 15, and 16). Each file contained rows that 5 values: frame id, joint id, joint position x, joint position y, and joint position z. Joint id 1 (center), 8 (right hand), 4 (head), 12 (left hand), 16 (right foot), and 20 (left foot) were collected for the normal RAD and joints 5 (right shoulder), 3 (center shoulder), 9 (left shoulder), 14 (right knee), and 18 (left knee) were collected from each frame of the data file. Then Each joint's distance from joint 1 (center) of the current frame was calculated along with the angles between them: 8-4, 4-12, 12-20, 20-16, and 16-8 for regular RAD and 5-3, 3-9, 9-18, 18-14, and 14-5 for custom RAD. Those calculations were then appended to their respected lists to be used for the histogram later on.

After calculating the distances and angles for each joint, it was then checked for any NAN values it remove any errors, then was plugged into numpy.histogram() in order to get the histogram of that distance/angle; default number of bins (10) were used for consistency. The histogram was spilt up into two parts: array of heights for the bins and array of bin edges, the bin heights were just needed for the assignment. Next step was to normalize the histogram, take the array of heights for the bins and divide it by the number of frames.

III. SVM

A. Overview

Support Vector Machines (SVM) is a supervised machine learning algorithm that is commonly used for classification and regression tasks. SVM is particularly useful when dealing with complex and high-dimensional datasets.

The basic idea behind SVM is to find the best possible line or hyperplane that separates the classes in a dataset. The line or hyperplane is chosen such that it maximizes the margin between the classes, i.e., the distance between the line/hyperplane and the nearest data points of each class. This margin is called the "maximum margin," and it represents the largest possible gap that can exist between the two classes.

SVM can handle both linearly separable and non-linearly separable datasets. In the case of non-linearly separable datasets, SVM uses a technique called "kernel trick," which maps the original dataset into a higher-dimensional space where it becomes linearly separable. This allows SVM to find a hyperplane that separates the classes in the transformed space.

One of the strengths of SVM is that it is a "black-box" algorithm, meaning that it does not require any assumptions about the underlying distribution of the data. This makes it particularly useful when dealing with complex datasets where other algorithms may fail. SVM also has a strong theoretical foundation, and its performance can be analyzed using well-established mathematical principles.

B. Hyperparameter values and Kernels

Different kernels and hyperparameter values are used to customize the algorithm's behavior. The choice of kernel and hyperparameter values can have a significant impact on the performance of SVM. Kernels are used to transform the original dataset into a higher-dimensional space where it becomes linearly separable. There are several types of kernels that can be used with SVM, including: linear kernel (this simply computes the dot product between two vectors), polynomial kernel (this raises the dot product between two vectors to a specified power), and radial basis function (RBF) kernel (this computes a similarity measure between two vectors based on their Euclidean distance). Hyperparameters are parameters that are not learned from the data but are instead set manually by the user. The hyperparameter used in SVM are C (this controls the trade-off between achieving a large margin and minimizing the classification error) and gamma (this hyperparameter controls the shape of the decision boundary in the case of non-linearly separable datasets).

IV. Results of SVM

The kernal used in the first part is linear kernal with a C of 0.000023455, in the first part is linear kernal with a C of 0.000023455, in the second part is polynomial kernal with a C of 0.00000004348354333, in the third part is adial basis function kernal with a C of 0.00000000000000000000000000000000000000004348.

A. RAD train

Accuracy: 0.7638888888888888
Precision: 0.8212811440365622

6	0	1	4	0	1
0	6	2	0	0	4
0	0	11	1	0	0
0	0	0	12	0	0
0	0	0	0	9	3
1	0	0	0	0	11

Accuracy: 0.6388888888888888
Precision: 0.7965005446623094

11	0	1	0	0	0
0	0	3	0	0	3
0	0	12	0	0	0
0	0	9	3	0	0
5	0	2	0	3	2
1	0	0	0	0	11

Accuracy: 0.9583333333333334
Precision: 0.9633699633699634

12	0	0	0	0	0
1	10	0	1	0	0
0	0	12	0	0	0
0	0	0	12	0	0
0	0	0	0	12	0
1	0	0	0	0	11

B. RAD test

Accuracy: 0.8958333333333334
Precision: 0.9151936026936026

8	0	0	0	0	0
1	6	0	1	0	0
0	0	8	0	0	0
0	0	0	8	0	0
1	0	0	0	7	0
1	0	0	0	1	6

Accuracy: 0.7916666666666666
Precision: 0.8879551820728291

8	0	0	0	0	0
3	5	0	0	0	0
0	0	8	0	0	0
0	0	0	8	0	0
2	0	0	0	6	0
4	0	0	0	1	3

Accuracy: 0.9375
Precision: 0.9545454545454546

8	0	0	0	0	0
1	7	0	0	0	0
0	0	8	0	0	0
0	0	0	8	0	0
2	0	0	0	6	0
0	0	0	0	0	8

C. Custom train

Accuracy: 0.8194444444444444
Precision: 0.8368131868131868

10	0	1	1	0	0
0	9	0	3	0	0
1	1	10	0	0	0
0	0	3	9	0	0
0	0	0	0	12	0
3	0	0	0	0	9

Accuracy: 0.7083333333333334
Precision: 0.8123015873015872

11	0	1	0	0	0
1	11	0	0	0	0
0	0	12	0	0	0
3	0	6	3	0	0
4	0	0	0	5	3
2	0	1	0	0	9

Accuracy: 0.9583333333333334
Precision: 0.9666666666666667

12	0	0	0	0	0
0	9	0	0	0	0
0	0	12	0	0	0
0	0	0	12	0	0
0	0	0	0	12	0
0	0	0	0	0	12

D. Custom test

Accuracy: 0.8541666666666666
Precision: 0.8795454545454545

8	0	0	0	0	0
1	6	0	1	0	0
0	0	6	1	0	1
1	0	0	6	0	1
1	0	0	0	7	0
0	0	0	0	0	8

Accuracy: 0.7708333333333334
Precision: 0.8694366635543106

8	0	0	0	0	0
4	3	1	0	0	0
0	0	8	0	0	0
2	0	0	6	0	0
2	0	0	0	6	0
1	0	0	0	1	6

Accuracy: 0.9166666666666666
Precision: 0.9333333333333332

8	0	0	0	0	0
1	6	0	1	0	0
1	0	6	1	0	0
0	0	0	8	0	0
0	0	0	0	8	0
0	0	0	0	0	8