MEHMET TUGRUL SAVRAN
***M**assachusetts **I**nstitute of **T**echnology*
6.835 SPRING 2018
MINIPROJECT 2

GESTURE RECOGNITION WITH KINECT

Question 1:

It is not difficult to see the "natural" differences among the sequences for each type of gesture. These differences between sequences are probably useful to prevent overfitting when training models. For example, for the **zoom in** gesture, different sequences start with different initial bodily orientation (the guy in Sequence 1 is definitely more tilted forwards than the guy in Sequence 0). Different sequences along this gesture also seem to tilt the body *during* the course of action as well as the arms. The tilting of the body seems to be a current trend for most of the gestures, in fact. I also saw slight bodily rotation differences between sequences for the same gesture. I believe, all in all, these sequences represent a diverse set of viable and natural ways of executing these gestures and it feels to me they will do a good job to prevent overfitting of the models (fingers crossed!).

Nearest Neighbor, if data weren't normalized, wouldn't succeed since un-normalized data implies varying dimensions. It doesn't mathematically make sense to compute distance between 2-dimensional and 3-dimensional coordinates, for example.

Question 2:

Uniform distribution among 9 events in sample space means an overall accuracy of 1/9 (~11.1%)

NN PART

Question 3:

The NN algorithm may only be executed on fixed-length gestures, since by definition is computes the L2^2 norms. L2 norm represents a vector that shows the triangular distance (Euclidean) between 2 vectors. This, by definition requires the sizes to be the same. (It doesn't make sense to compute the distance between (x=0,y=3,z=5) with (x=0,y=1) because z is not defined on the second vector).

For the function, what I did was to take on a modular approach in tackling the problem. I wrote a helper function that handles the addition and removal of indices. When adding indices, I computed a variable showing per how many steps we will be adding/duplicating the frames. This variable, called steps in my function is computed by cur_frame_number//num_frames_to_add
I then computed the start index, from which we will execute these steps of addition. Start is defined as start =(cur_frame_number%num_frames_to_add)//2

The function returns a list of indices of frames we will add .

For the removal of indices, a similar approach was taken to find the variable called steps, which is defined the same. However, start's definition is different in that we start skipping after the first step, formally:
start = (steps+cur_frame_number%num_frames_to_remove)//2;

Next, and finally, I iterated over each sequence for each gesture and call these helper methods to modify the frames of sequences.

Question 4:

Nearest Neighbor is an awesome technique, because it is very simple and can yield extremely powerful results: *Occam's Razor*! It is simple because you don't need to hand-tune or "computer-tune" parameters. It simply calculates proximity based on a heuristic mainly the L2 norm. No actual training involved!

Nearest Neighbor is set back by its dependence on a meaningful distance function. Here, we are blessed by the 3D coordinates which we can exploit meaningfully. However, L2 norm won't always apply to every single classification problem. Another drawback would be its computational efficiency. If there are m examples, who are represented by d dimensions, then it'll calculate the distance to all other m examples (it runs in O(md) time).

Question 5:

ratio/frames	0.4	0.5	0.6	0.7	0.8
20	0.62	0.67	0.63	0.69	0.77
30	0.80	0.84	0.85	0.91	0.90
40	0.81	0.80	0.85	0.86	0.85

The accuracy seems to be increasing in both ratio and frame numbers. However, 30 seems to be achieving the highest accuracy, due in part of duplicating frames resulting in loss of accuracy.

Indeed, following the results from this table, I tested the model with 30 frames and ratio 0.9, and saw 0.96 accuracy, which seems promising!

DECISION TREE PART

Question 6 (this question was misnumbered as Question 5 on the PDF)

Before looking at the code, I find it quite normal that test accuracy changes because the software could be randomly dividing the data set into training and testing chunks which would be contributing to the accuracy being non-deterministic.

After looking at the code, I found that the non-deterministic result was indeed due to the random sampling of training/test data.

Question 7 (labelled Question 6 on the PDF)

Let's go through what happens from the very beginning of the code. Initially, raw Kinect gesture data is loaded through the load_gestures() method. The data is then normalized through the implemented normalize_frames() method.

Next, **labels** and **actual sequences** are sampled to separate arrays, Y and X respectively.

X's shape is (270, 1188).
Y's (270,), which is pretty much an array of labels.
270 is the total number of sequences and 1188 is the total number of samples

These X and Y arrays are further split into training and testing divisions, namely X_train, Y_train and X_test, Y_test.

X_train and Y_train are fed into the DecisionTreeClassifier. X_test and Y_test aren't shown to the classifier, they are rather used to evaluate the classifier (testing the classifier).

Accuracy\Ratio	0.1	0.4	0.7	0.9
Percent	30.52	69.135	72.84	75.56

The accuracies were found by evaluating each ratio 5 times and then taking the average performance.

0.9 seems to be the best ratio.

Question 8 (Labelled Question 7 in PDF)

Gini impurity measures how frequently a randomly chosen element from a set would be incorrectly labelled if it were to be randomly labeled with respect to the label distribution.

Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset.

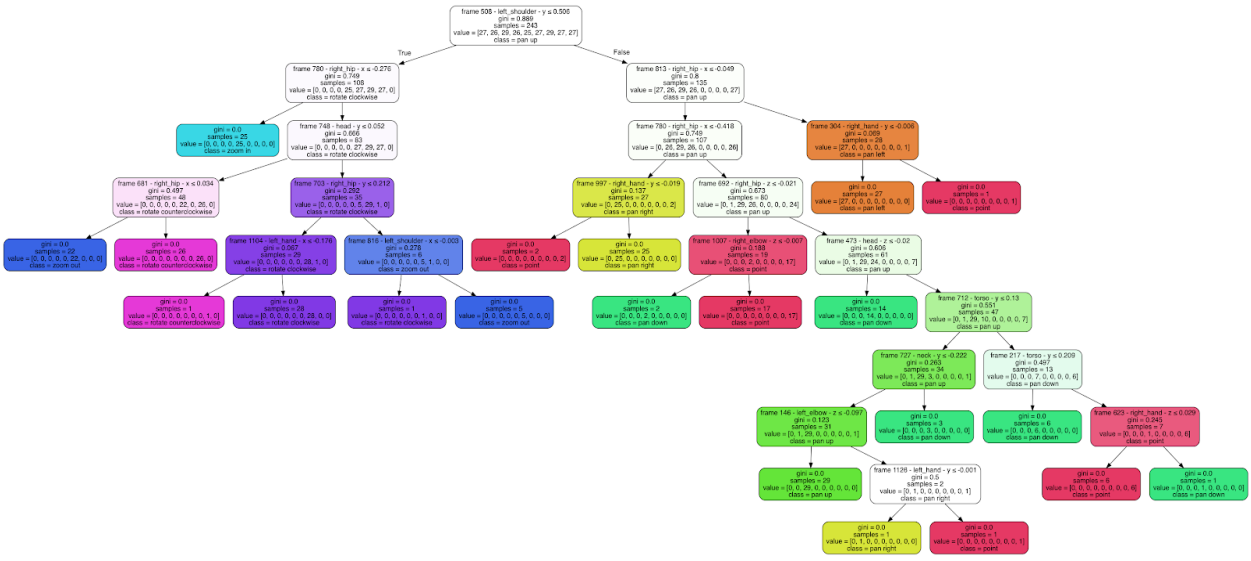
$$\text{Gini\_Impurity} = 1 - \sum_{i=1}^{N} p^2{}_i$$

Where N is the size of the finite set of classes (sequence labels in our case)

I would thus expect to see Impurity scores of 0 at the leaf nodes.

Question 9 (labelled Question 8)
After running the code with ratio 0.9, I saved the tree model whose testing accuracy was 78.52%.

All leaves have an impurity score of 0 because at the leaf node there is no remaining samples that can lead to an alternative labeling - we pruned out all other possibilities. This connects nicely with the "value" array, which describes number of samples for each class. Hence at the leaf of the tree, the value array will only have one non-zero entry, which is the number of samples for that label. The index for the non-zero entry is the label.
The tree's png image is reproduced below, and is also included within the folder.



RNN PART

Question 10 (labelled Question 9)

I tested out the rnn model on the following triplets of parameters. Each triplet is of ordered form (batch_size, epochs, latent_dim).

(12,100,16) → acc: 0.8889 - val_acc: 0.8148

(12,400,16) → acc: 1.0000 - val_acc: 0.8765

(8,40,8) → acc: 0.7090 - val_acc: 0.6667

(16,40,16)--> acc: 0.3915 - val_acc: 0.3580

(16,200,16) → acc: 1.0000 - val_acc: 0.8519

(12,400,16) → acc: 1.0000 - val_acc: 0.9383

(16,400,16) → acc: 1.0000 - val_acc: 0.9259

(24,400,24) → acc: 1.0000 - val_acc: 0.8642

The highest validation accuracy value I obtained was through a batch size of 12, 400 epochs and 16 latent_dims.

Model didn't seem to overfit with these parameters, since validation accuracy was just as high.

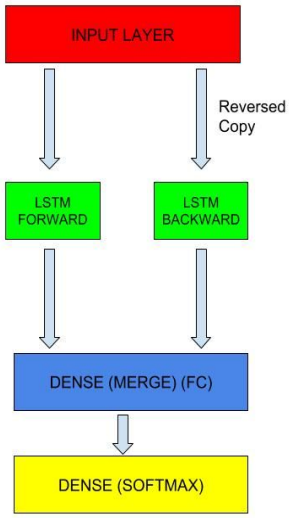
Question 11 (Labelled Question 10)

I tested the bi-directional LSTM with the 2 parameter triplets that let the unidirectional (previous) LSTM model achieve the highest validation accuracies.
(16,400,16) → acc: 1.0000 - val_acc: 0.8642
(12,400,16) → acc: 1.0000 - val_acc: 0.9226

I then explored parameter triplets with smaller epoch numbers.
(12,100,16) → acc: 0.9363 - val_acc: 0.8448
(16,200,16) → acc: 1.0000 - val_acc: 0.9012

The bidirectional LSTM seems to gather more information in fewer number of epochs - which is what we expected initially. However, the performances of both LSTM models seem to converge at higher epochs.

The schematic of the LSTM diagram follows below:



Question 12 (Labelled Question 11)

I really liked this project! It was much easier to follow through compared to miniproject 1. In miniproject 1, one had to implement functionalities that depended on previous steps, and unfortunately due to the nature of the assignment it was hard to "check yourself" after each step. (I still don't know how I didn't blow up the entire assignment- miracle!)

However, this miniproject helped me stay confident throughout - I also loved the clear explanations. It also served as a really nice "survey" of cutting edge techniques to classify gestures. I hope to see more projects like this in 6.835!