CSE 535 Mobile Computing Assignment 2

Amey Athale*, Arpit Arora†, Aditya Chavan‡, Jayesh Kudase§
ASU School of Computing, Informatics, and Decision Systems Engineering
Arizona State University
Tempe, AZ, United States
*aathale@asu.edu
†aarora58@asu.edu
‡apchavan@asu.edu
§jkudase@asu.edu

I. Introduction

We developed an online application that has the ability to classify American Sign Language. The developed application accepts Human Pose Skeletal key points of a video that have a sign from American Sign Language. We then use machine learning models and these key points from the video to classify the ASL signs and return the label as a JSON response.

For our application,we all used the same features and have used Decision Trees, K Neighbours, Random Forest and Extra Trees algorithms to develop our machine learning models. All of these would be working on the key points independently and returning the predicted class. Now we will discuss these briefly.

II. Pre-processing and Features Extraction

A. Universal Normalization

In our project, all four members have used the same set of features which ensured maximum accuracy during testing and validation step. We have used a scaling technique called as Universal Normalization which helps generalize the input rather than keeping it dependent on the camera. As explained by Prof. Ayan Banerjee in his lecture videos, we normalized the positions of wrist with respect to eyes, nose and shoulder. This helps us keep the relative measurement and keeps it independent of the camera angle and distance from the person performing the gesture. The formula used is as follows:

$$X_{LW} = \frac{X_{LW} - X_N}{X - LE - X_{RE}}$$
$$Y_{LW} = \frac{Y_{LW} - Y_N}{Y - N - Y_{RS}}$$
$$X_{RW} = \frac{X_{RW} - X_N}{X - LE - X_{RE}}$$
$$Y_{RW} = \frac{Y_{RW} - Y_N}{Y - N - Y_{RS}}$$

where *X* and *Y* denote the X and Y coordinates of the body part and *LW, RW, LS, RS, LE, RE, N* denote Left Wrist, Right Wrist, Left Shoulder, Right Shoulder, Left Eye, Right Eye and Nose respectively.

B. Dynamic Time Warping Distance

Dynamic Time Warping(DTW) is a method for calculating distance or similarities between two temporal sequences. It is mainly used for measuring relative distances, especially when the data is in a time series format. This is one of the features used by us to determine how the same gesture done by different people varies from one another which helps us generalise our model and account for all the variations that could possibly occur in the testing data.

C. Fast Fourier Transform

Fast Fourier transform, apart from its other application, plays a vital role in image processing. The algorithm basically converts signals to their representation in the frequency domain. It is a common feature while working with time series data. The property of convolution makes it an important feature. Using FFT the data gets divided into frequency components and hence the distribution of data is better, also all the properties of data are preserved even after the transformation.

D. Discrete Wavelet Transform

Discrete Wavelet Transform converts signals to their discrete wavelet representation. DWT plays a key role in image processing. It is the basis of the JPEG2000 image compression standard. With the increase in the image resolution, the storage space for that image increases. Using DWT, the size of the image is reduced without having any effect on the quality of the image hence resolution can increase. Discrete wavelet transform decomposes signals into various subbands such that the lower subbands have finer frequency resolution.

III. Inference Models

We have a total of four different classification models which have been used for training and testing that according to us should be able to provide the maximum accuracy. The models used are as follows:

A. Extra Trees Classifier

Random Forest Classifier builds on the fact that more uncorrelated trees end up giving better precision. Extra Trees extends this application and provides great results. In Extra Trees classifier what we actually do is we introduce more deviation in ensemble learning and hence more deviation in building trees. Here also we pounce on the advantages on ensemble learning and create multiple trees but these trees are given randomized features. As different trees have different and random features they end up creating a diverse set of boundaries to make classifications. One more good thing about Extra Trees is that it is computationally less expensive as it selects random features at the splits.

The following hyper-parameters were used for training this model: Parameters:

- n_estimators = 100 : It represents the no. of trees in the forest.
- criterion = 'gini' : This parameter is used to measure the split quality using Gini Impurity.
- min_impurity_decrease: 0.05 : If the split has an impurity decrease greater than or equal to the value specified, the node will split.
- min_samples_split = 2 : It represents the minimum samples required to split.
- bootstrap = True : It determines whether bootstrap samples are used for building trees.
- max_leaf_nodes = 0 : This means the tree grows with unlimited leaf nodes in best-first fashion.
- min_impurity_split = 0.05 : It represents a threshold such that if the impurity is above the threshold the node will split.
- warm_start : False : The solution of the previous call is not reused and a whole new forest is used to fit.

B. Random Forest Classifier

Random Forest is a classifier that work using the principle idea of what is known as "ensemble" learning. As we saw while discussing the Decision Trees, they are susceptible to overfitting on the training data. In order to mitigate this fundamental problem we end up using algorithms like random forest. Basically Random Forest Classifier creates multiple trees that are not highly correlated and then each tree, like a decision tree, gives out a decision or a classification. The overall classification is dictated by the mode of the multiple trees or in other words the plurality of the classifications made. The key here is for the trees to be uncorrelated, and higher the number of uncorrelated trees higher will be the precision. Of course, we certainly need to consider the trade-off between computation and precision required.

The following hyper-parameters were used for training this model: Parameters:

- n_estimators = 100 : It represents the no. of trees in the forest.
- criterion = 'gini' : This parameter is used to measure the split quality using Gini Impurity.
- min_impurity_decrease: 0.05 : If the split has an impurity decrease greater than or equal to the value specified, the node will split.
- min_samples_split = 2 : It represents the minimum samples required to split.
- bootstrap = True : It determines whether bootstrap samples are used for building trees.

C. K Nearest Neighbours Classifier

K neighbours or also popularly known as K nearest neighbours is an effective algorithm that gives good results for classification tasks. KNN basically depicts all the cases present and the unseen or the new cases that come in are classified based on the similarity measure. Input to this algorithm is the value of how many nearest neighbours to be considered while classifying with confidence. The end classification depends on the majority class of its k neighbours. In order to select the value of K, it is essential to look at the data. More often than not, larger values of K produce better results as they nullify the impact of noise on classification.

The following hyper-parameters were used for training this model: Parameters:

- n_neighbors = 80 : It is the default number of neighbours used by k neighbours classifier.
- weights = 'uniform' : It represents the weight function for prediction. 'uniform' indicates that all the points in every neighbourhood are equally weighted.
- algorithm = 'kd_tree' : It determines the algorithm to be used for predicting the nearest neighbours.
- p = 2 : It is the power parameter used for the Minkowski metric. Here 2 means it is equivalent to euclidean distance.

D. Decision Trees Classifier

As the aim of our application is classification, let us look at how this algorithm helps. Decision trees are tree-like models with their root at the top. It is a topology of decisions and how we arise to those decisions. If we take a path on a decision tree, it can also be viewed as a series of control statements. At every stage we make these decisions or better called as refining the classification. By the end of the process, as we reach the leaves of the decision tree we have a classification on our hand. One key factor while working with decision trees is, they are susceptible to overfitting and hence we need to know when to stop splitting the tree. Though a little unstable, decision trees are a great way to classify as they are easy to represent and comprehend.

The following hyper-parameters were used for training this model: Parameters:

- criterion = "gini" : This parameter is used to measure the split quality using Gini Impurity.
- max_depth = 0 : The parameter determines the maximum depth of the tree. Zero indicates no upper bound to it.
- min_samples_split = 2 : It represents the minimum samples required to split.
- min_impurity_splitfloat = 0.05 : It represents a threshold such that if the impurity is above the threshold the node will split.

IV. Results

On each of the above described models, we performed 10 fold cross validation and final training and testing was performed on these models with a train test split of 33% and the final trained model is stored in the form of a pickle file. All the experimentation was done on Google Colaboratory to maintain uniformity and the following are the results obtained:

A. *Extra Trees Classifier*

10 fold cross validation on complete dataset:
Time taken = 28.3 seconds
Accuracy on validation set = 46.333%

Training and Testing model on 33% split:
Time taken = 3.69 seconds
Accuracy of Trained Model = 99.784%
Precision = 0.998
Recall = 0.998
F Score = 0.998

B. *Random Forest Classifier*

10 fold cross validation on complete dataset:
Time taken = 3 min 44 seconds
Accuracy on validation set = 54.393%

Training and Testing model on 33% split:
Time taken = 17 seconds
Accuracy of Trained Model = 99.038%
Precision = 0.991
Recall = 0.991
F Score = 0.991

C. *K Nearest Neighbours Classifier*

10 fold cross validation on complete dataset:
Time taken = 11.3 seconds
Accuracy on validation set = 13.744%

Training and Testing model on 33% split with nearest neighbours = 80:
Time taken = 7.54 seconds
Accuracy of Trained Model = 87.749%
Precision = 0.879
Recall = 0.877
F Score = 0.877

D. *Decision Trees Classifier*

10 fold cross validation on complete dataset:
Time taken = 21.2 seconds
Accuracy on validation set = 50.867%

Training and Testing model on 33% split:
Time taken = 1.68 seconds
Accuracy of Trained Model = 96.687%
Precision = 0.967
Recall = 0.967
F Score = 0.967

V. Conclusion

We conclude our assignment by deploying this application on Amazon Web Service's EC2 instance. We have provided the public IP of our server in the README.md file along with the running instructions. Thank you Prof Ayan Banerjee for this assignment.

References

[1] Swain, Philip H., and Hans Hauska. "The decision tree classifier: Design and potential." IEEE Transactions on Geoscience Electronics 15.3 (1977): 142-147.
[2] Liaw, Andy, and Matthew Wiener. "Classification and regression by randomForest." R news 2.3 (2002): 18-22.
[3] Garreta, Raul, and Guillermo Moncecchi. Learning scikit-learn: machine learning in python. Packt Publishing Ltd, 2013.
[4] Cunningham, Padraig, and Sarah Jane Delany. "k-Nearest Neighbour Classifiers–." arXiv preprint arXiv:2004.04523 (2020).
[5] Lafta, Raid, et al. "A fast Fourier transform-coupled machine learning-based ensemble model for disease risk prediction using a real-life dataset." Pacific-Asia Conference on Knowledge Discovery and Data Mining. Springer, Cham, 2017.
[6] Weste, Neil, David J. Burr, and Bryan D. Ackland. "Dynamic time warp pattern matching using an integrated multiprocessing array." IEEE transactions on computers 8 (1983): 731-744.