COMP90051 Statistical Machine Learning - Project 1 Report

Team Name: Hunter
Xiaoyu Wu (1218098) Haohan Zhou (958579)

1 Introduction

Human action recognition has been widely researched in the field of computer vision in recent years. By using action recognition techniques to learn the sequence of movements of a human body, certain behaviour can be identified and categorized. This can be applied in various areas including health care, video monitoring, assisted living and so on[1].

The primary problem of this project is to identify human action recognition by classifying the behaviour based on a sample of 3D joints movements sequences. In addition, the distribution of behaviour categories is highly skewed in the given data, this leads to another problem --- imbalanced data classification. In this report, we implement classical machine learning and neural network methods to address these problems. The proposed models will be compared and evaluated regarding the overall performance of the Kaggle competition and the Accuracy(ACC) score.

2 Data Processing

The given training data contains 9388 sequences of 3D joint movements with 960 features, which belong to 49 classes. For validating the model, we randomly split 20% data from the given training dataset as the validation set and the rest of the data will be used for training models. Another test data with 2859 sequences will be used for generating the final classification and evaluation.

2.1 Data OverSampling:

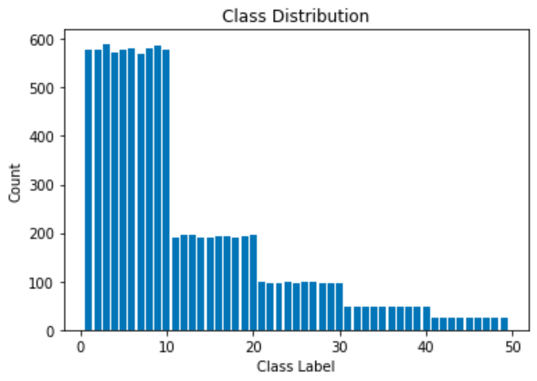

Figure 1. Class Distribution

Since the class of the dataset is imbalanced, it is possible for classifiers to focus on learning the patterns of majority classes. To handle this problem, we use oversampling, which makes synthesis data for minority classes so that all classes have similar numbers of instances.

We use 3 methods: SMOTE, borderline SMOTE and ADASYN. The basic idea of SMOTE is to generate new instances by making a convex combination of several points from the same class.
The generated instances will have a close Euclidean distance to instances of the same class. However, this method is sensitive to outliers. To decrease the influence of outliers, borderline will ignore the outliers when generating new instances, whereas ADASYN gives different weight to instances, fewer instances will be generated near those suspicious points. We can see that all of these three methods rely on geometric distance (Euclidean distance) between points, which means newly generated instances may not capture complex patterns. For example, in human action recognition, it might be important to calculate coordinate differences of joints between two consecutive frames in human action recognition. (Although features in this dataset are not coordinates)

2.2 Feature Engineering

After oversampling, we use standardization on both original and oversampled datasets to let all features have a similar impact. Then we apply feature selection and dimension reduction. There are 960 features, which is a large space. Reducing irrelevant features or compressing feature vectors to a low space may make training more efficient and quicker. As for feature selection, we use generic univariate feature selection, which selects the features based on univariate statistical tests (we use ANOVA F-value, which reflects the confidence of whether a feature is related to class). One weakness of this method is the assumption from statistical tests, for example, ANOVA assumes instances are normally distributed, violating this assumption might make the method less effective. We also use SelectFromModel, which selects features based on importance weight from the estimator. We choose an extra tree as the estimator, which is a similar method to the random forest, except random forest subsamples the input data with replacement, an extra tree uses original samples. Note that it is hard to make feature selection or dimensional reduction for CNN (Temporal convolution) and LSTM because those models need the input shapes of different timesteps to be the same.

3 Main Approaches

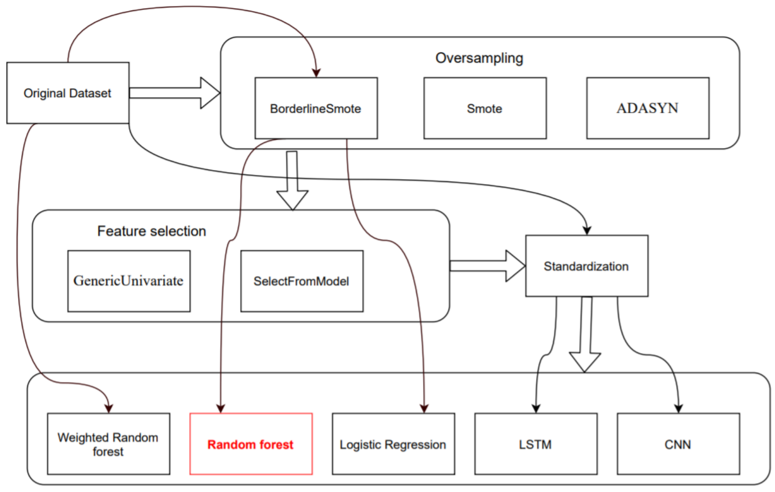

Figure 2. Experimental Pipeline

For the experimental process, our expected procedure was first to oversample the original data, then perform feature selection and lastly train the classifier. However, based on the findings from experiments and model performance, the decent results are actually output by the processes that follow the arrows, as shown in figure 2.

3.1 Logistic Regression

To obtain further understanding of data, the performance of preprocessing methods mentioned above, we first start with the logistic regression model as a baseline model due to its faster and easier implementation. Since we have 49 classes and the class distribution is imbalanced which requires a model to perform multi-classification on the imbalanced data. However, logistic regression is designed to handle binary classification by default and cannot be applied to the imbalance data directly. Therefore, we train the multinomial logistic regression on the upsampled data by using SMOTE techniques and perform hyperparameter tuning. By doing this, we find that the overall model validation accuracy scores are around 0.2 which also do not vary much among parameters tuning and the model is easy to overfit. Then we move on to the models that would handle imbalanced data multi-class classification more efficiently.

3.2 Random Forest (RF)

After extensive research on the method of handling imbalanced data, we find that random forest is widely exploited as it is robust to the outliers and is affected less by the noise. Also, it can perform random feature selection during the induction of trees, does not require feature scaling as pre-work and can be powerful to handle high dimensional data. However, since the random forest is designed to reduce the overall rate of error, it may value more on the accuracy of the majority class[2]. Thus, in this report, we utilize two main methods to assist random forest handling imbalanced data.

3.2.1 Sampling Method

In this part of the method, we oversample the training data by using various SMOTE methods mentioned before to provide relatively balanced data for the random forest classifier. After using the grid search method for hyperparameter tuning, we observe that the model is optimized and able to achieve the highest ACC score of 0.35666 on the Kaggle competition when `'n_estimator'` = 750 and `'min_samples_split'` = 4 with borderline SMOTE sampled data.

3.2.2 Cost-Sensitive Method

In the cost-sensitive method, to overcome the imbalanced problem we change the class weight to penalize the minority class misclassification. Since there are 49 classes in total, instead of manually assigning the weight for each class, we tried `'balanced'` and `'balanced_subsample'` for the parameter `'class_weight'` which is supported by the random forest classifier. The overall accuracy scores performed by these 2 values of `'class_weight'` are close to each other but they both achieve slightly lower performance than the classifier with the sampling method.

3.2.3 Analysis for Random Forest Methods

Other than the above two main methods for the standard RF, we also apply a balanced RF classifier. The overall accuracy of this approach is obviously lower than all other standard RF classifiers. This is because the balanced RF classifier randomly undersamples each bootstrap sample to reach class balance which may result in the loss of information, as the majority class is reduced. The reason why the weighted method in Section 3.2.2 does not achieve higher performance is that the 'class_weight' is introduced to achieve class balance by increasing the minority class weight and decreasing the majority ones based on a certain weighting formula. During this reweighting process, the artificial minority examples will be introduced and possible valuable information may be removed from the majority class.

While with the help of borderline SMOTE, the minority will be oversampled with the nearest neighbours algorithm and ignoring the outliers to generate more meaningful instances, and also remove the redundancy from the majority. By comparing the way of handling class imbalance and ACC score, the RF classifier with the sampling method outperforms the others.

3.4 Neural network

The neural network is a popular method and usually achieves a high ranking in many Kaggle competitions. We try two types of the neural network: temporal convolution and LSTM, both of them have the ability to handle sequential input. We split 960 features into 16 timesteps, each timestep includes 60 features, so the input will be a 16*60 matrix. We also add 1 fully connected layer after temporal convolution and LSTM layer, this might capture some useful patterns from the output of the previous layers. During training, we found that standardized features make training faster. In addition, we also found that oversampling has little influence on accuracy improvement. Unfortunately, these two methods did not perform well. Temporal convolution got 42 validation accuracy, however, when it comes to the test set, acc drops to 32%. As for LSTM, it has 35% validation accuracy, since it performs worse than temporal convolution, we did not choose it as a candidate. One possible reason for the poor performance of neural networks is: the features are structural, 20 joints can be represented as a graph, adjacent nodes might influence each other. Flatten the whole graph into one 60 dimension vector might not capture structural patterns. One possible improvement is using building graph neural networks to capture patterns of 20 joints and let outputs of graph neural networks be inputs of LSTM or temporal convolution.

4 Conclusion and Error analysis

In this report, we build multiple classifiers to categorize human behaviour from 3D joints movement sequences and handle the data imbalance. We analyze each model by comparing its advantages and drawbacks regarding its properties. Based on the overall performance and ACC score on the Kaggle competition, the RF classifier with borderline SMOTE is selected as our final approach.

Although our final approach outperforms the others we have shown, it only achieves 0.35666 / 0.5(max possible accuracy) on the Kaggle competition which means there is more for advancement. The reason for the final RF model not having further improvement is that we mainly rely on the random feature selection supported by the RF during the classification and we have not developed an effective enough feature extraction method that considers the specialty of the sequential movement of 3D joints. Therefore, RF does not sufficiently learn the characteristics of each behaviour and may not connect them with joint sequence movements together. To improve this, we may consider transforming and combining the value of the x,y and z axes of each feature into one vector, and try to find the similarity and characteristic of sequential joint movements for each certain behaviour. This may help machine learning models to have a more accurate understanding of features and classification outcomes. Or we can try other advanced methods such as a combination of graph neural network and LSTM.

References

[1] Ahad, M., Ahmed, M., Das Antar, A., Makihara, Y. and Yagi, Y., 2021. Action recognition using kinematics posture feature on 3D skeleton joint locations. *Pattern Recognition Letters*, 145, pp.216-224.

[2] C. Chen, L. Breiman, and A. Liaw, "Using Random Forest to Learn Imbalanced Data," *Semantic scholar*, 01-Jan-1970. [Online]. Available: https://www.semanticscholar.org/paper/Using-Random-Forest-to-Learn-Imbalanced-Data-Chen/2138b37bfced70599d26dfccbf93a8e7a4b7ad85. [Accessed: 07-Sep-2021].