COMP90051 Statistical Machine Learning - Project 1

Crippling Regression (Group 69)
Tuan Khoi Nguyen (1025294), Si Cheng (1156375), Kuoyuan Li (1072843)

Abstract

This report will describe the systematic process of data preprocessing and classifier construction for the given task of action classification for 2959 movement sequences out of 49 possibilities, namely imbalance handling, evaluating classic Machine Learning classifiers and ensembles, and most notably, presenting a new model to the problem, which altogether have brought the team to the Top 20 on the Kaggle competition [1].

1 Overview and Preprocessing

With the data coordinates given for 16 frames, this human action recognition task can be identified as time series or sequence multi-class classification problem. This suggests that an instance can have the following features that are effective for classification: dynamic spatial patterns that shows a trend at any time in the data, or constant statistical information of the sequence such as mean and overall displacement.

1.1 Dataset Overview

A quick exploitation on the data shows that label distribution is imbalanced, with more than half of the instances belonging to only the majority 10 out of 49 actions. As a result, conventional multi-class classification models that see too few examples from minority classes may not generalise well. Therefore, adjustment is needed to increase the representation of low-occurrence labels. Feasible methods for evaluation includes Weighted Random Sampling, SMOTE Oversampling and Random Undersampling.

Weighted Random Sampling Weighted random sampling alleviate class imbalance by sampling training data according to weights associated with each class, which assign higher weights to minority classes and otherwise. However, as instances are repeated or deleted, the training data may lose representation, and overfit as a result.

Synthetic Minority Oversampling Technique (SMOTE) & Random Undersampling (RU) SMOTE overcomes overfitting problem of having too many repeating instances, by generating synthetic data systematically within the label's space boundaries [1]. It is also stated that the technique performs the best when combined with random undersamping of majority labels. Tuning is needed to get the most suitable sampling sizes, which are the method's hyperparameters.

1.2 Hyperparameter Tuning and Data Representation Criteria

Adversarial Validation With numerous balanced datasets, a method to assess data representation is needed. Adversarial validation [2] measures the similarity between balanced training dataset and testing dataset by attempting to distinguish them through binary classification. The lower the accuracy, the more similarity between the datasets, and the training dataset will better represent the testing dataset.

Grid Search Adversarial Validation (GSAV) To find the optimized set of sizing for SMOTE/Undersampling modification, the number of instances to undersample the majority labels and to oversample the minority labels are tuned using 3×3 grid search, evaluated with adversarial validation.

Result evaluation

Result shows that downsampling high occurrence labels to 300 samples and oversampling underrepresented labels to 180 samples has the best data representation.
However, when comparing the best choice to original data as shown in Figure 1, despite improving the representation of minority labels, the training data has loss in overall representation as a trade-off. This factor is taken into consideration, which later becomes the inspiration for our final Kaggle model.

[1] Ranked 20 public leaderboard, co-ranked 20 private leaderboard.

2 Learner Models

The adversarial validation result of 70% shows that the chosen training data does not represent the testing set well. Therefore, in this stage, we focused on exploring a wide range of models that can analyse the features well, without fitting too close to the training data.

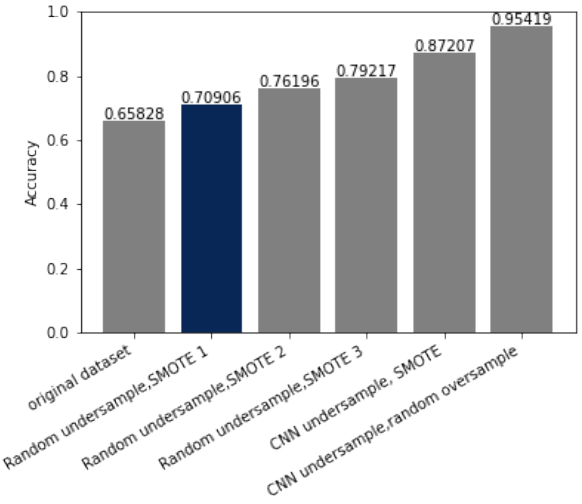

Figure 1: Adversarial validation accuracy on different datasets. Blue column indicates the chosen dataset, which has the best representation.

2.1 Standalone Models

Baseline & Benchmark Gaussian Naive Bayes & Logistic Regression are set as baseline models for this task, due to the probabilistic models' low performance in high-dimensional data. For benchmarking, 2 tree-based ensembles - Random Forest and eXtreme Gradient Boosting are used as the benchmark models, due to their nature of combining outputs from different feature subsets to form more robust predictions.

Long Short-term Memory (LSTM) LSTM, widely used in learning sequential data, is an improved version of Recurrent Neural Network which prevents information loss through ReLU activation. The model we built contains 16 timesteps, each corresponds to one time frame. At each time step, 60 features of the corresponding frame were firstly fed to a one-layer CNN and then to a two-layer LSTM.

MiniRocket MiniRocket is an ensemble learner dedicated for time series data, which runs a number of convolutional filters with varied parameters to extract the most amount of features in a sequence [3]. With different results each run, more robust version of this model runs a number of estimators at the same time, and uses voting to get the final prediction.

2.2 Stacking Ensembles

To utilise the overfitting reduction of data, we tried to combine multiple observations to form a more robust classifier.

Weighted Sample Stacking The model has 5 LSTM base learners trained on different subsets of training set that has a specified distribution, and a Multi Layer Perceptron with 3 fully connected layers as meta classifier. Through specifying the weightings, the model can be controlled to be better at recognizing a specific label.

One vs Rest (OvR) Random Forest This model uses a multi-class random forest classifier as the meta classifier and 49 random forest binary classifiers as the base learners. Each base classifier only needs to learn the representative features of one class, which is more simple than training one classifier for all classes.

Stacking Cross Validation For each standalone model, this method uses cross validation to generate predictions on the training set. Each prediction becomes a feature in the training data, while the model's predictions on test set will be test data. These new sets will train and predict through a meta Categorical Naive Bayes classifier. This probabilistic meta-classifier not only helps to take in diverse information, but also identifies where a false prediction could belong to.

SMOTE/Undersampling Comprehensive Knitted Stacking (SUCKS) Based on the observation of data representation trade-off mentioned in Section 1, we present SUCKS as a new model to this project. SUCKS further reduces the variance during training, by training on different datasets generated on the same parameters of SMOTE/RU, but on different random seeds, and combine the results via weighted voting. This will force the model not to miss out any data during undersampling, and knit different trend identifications together during training, similar to boosting method.

3 Setting Up

This section will describe our preparation process for the training stage, going in detail from the beginning to retrieving the results that are to be evaluated in this report.

Environment To speed up the training, we prepared a Virutal Machine instance on Microsoft Azure platform, with 6 CPU cores and 28GB of RAM. We also run concurrent training on local devices, in order to speed up the processes.

Data Engineering Given that the data will be run on traditional models, and the number of training instances may not be enough for neural networks to learn the features due to high dimensionality, we proceed to do engineering to give our models a good head start. Observation on specific data points from the same action show that they share information in the frequency domain, such as distance between peaks and gradients, yet have different magnitudes. Therefore, we normalized the data to the range of [0,1], using column-wise min-max scaler in order to make the trends more detectable.

Validation We were not able to fully evaluate the test set accuracy on the Kaggle public leaderboard, as it only contains 30% - a small proportion of the data, hence needed other accuracy assessments to rely on. Through Stacking Cross Validation, we were able to generate a nested training prediction for most models that were used in the base layer to compare against the ground truth. For other models, a hold out of 10% (939 sequences) were preserved for validation.

Hyperparameter tuning GSAV is done for choosing SMOTE/RU sample size, and 25% Hold-out Random Search is used to find the best LSTM model. For MiniRocket, initial accuracy evaluation shows that the model no longer increase it's accuracy after 15 estimators. Other models will have their parameters set to default value.

Running the models Each single model and base-specific ensembles are run one by one and have few of their results submitted to Kaggle to estimate data generalisation. For Stacking Cross Validation, all single models will be used as base classifier. SUCKS will repeat the same stacking process again for other datasets, and decide the final predictions by voting. More weight can also be manually tuned for predictions with well-known accuracy.

4 Results & Discussion

Technique Category	Model Names	Raw Data		SMOTE Data	
		Hold-Out/CV	Test	Hold-Out/CV	Test
Single Model	Logistic Regression	0.21	/	0.39	0.27057
	Gaussian Naive Bayes	0.21	/	0.31	/
	Random Forest	0.40	0.27170	0.60	0.34949
	XGBoost	0.43	0.23900	0.72	0.35062
	LSTM	0.45	0.18376	0.50	0.31567
	MiniRocket – Single model	/	/	0.63	0.34385
	MiniRocket – Voting 3	/	/	0.64	0.35738
	MiniRocket – Voting 15	/	/	0.67	0.38782
	MiniRocket – Voting 60	/	/	0.68	0.38668
Ensembles	Weighted Stacking	0.45	0.34160	0.64	/
	OvR Random Forest	0.43	0.34498	0.64	0.35963
	Stack	/	/	0.71	0.40811
	SUCKS	/	/	/	0.41600

Table 1: Overall Result Table of all training models

• Overall, observation shows that all models returned a much lower accuracy on test data, proving the dataset's lack of representation. Largest differences are found in the tree models, likely because they only make predictions on separate features rather than looking at the data in sequence. Sequential models and ensembles has smaller accuracy gap, proving that either they were able to detect sequential features, or was able to gather different observations to avoid overfitting.
• Benefited from being an ensemble of other models and the training on multiple balanced datasets, SUCKS outperforms other models in alleviating the representation loss of SMOTE/RU for generalisation improvement. Thus, it is chosen as the final submission to the Kaggle competition. The model yielded **40.592%** accuracy on the private testing dataset, only 1% lower than the public testing. This small gap is consistent with our experiment observations and shows that our model has overcome the problem of bad data representation.

Difficulties A downfall of SUCKS is that it takes the most computation time to train one dataset, making it prone to fatal error. Due to an Azure faulty crash, along with limited time and resource constraint, we were only able to generate 10 datasets. We believe with more data and better resources, the model would be able to flourish more in this task.

Future Improvement Due to the time and resource constraint, we have to miss out a few ideas for the task. This includes using `tsfresh` - a package that automate feature extraction on sequences, maximising the potential of SUCKS through generating more datasets, or testing further methods that diversify the training process, either instance or feature-wise. Current results can be used further when more data is given, where feed-forward neural networks models are expected to perform better, SUCKS and other ensembles will also require fewer datasets or partitions to cover all data representations.

References

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.

[2] Konrad Banachewicz. Adversarial validation and other scary terms, 2017.

[3] Angus Dempster, Daniel F. Schmidt, and Geoffrey I. Webb. MiniRocket. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. ACM, August 2021.