**InstRecognition Tool**

**Vladimir Gajčin, SW-49/2014**

1. Motivation

Motivation for this project lies in the potential ability to use the solution in various number of ways that help with music production. For example, in software that extracts the sample of a certain instrument from a multi-instrument sample. The first step is the recognition.

2. Research questions

In this project I wanted to see how instrument classification behaves in the “real world”: When there are multiple (more than two) classes, the dataset consists of excerpts from real songs and the testing data sometimes contains samples which have more than one instrument.

For that, I used [IRMAS](https://www.upf.edu/web/mtg/irmas) dataset, which was a perfect fit for the problem I was trying to solve.

3. Related work

This problem has not been solved by many, although there is [one project](https://github.com/AbubakarSarwar/Instrument-Classification-of-IRMAS-Dataset) which used K-Nearest Neighbours, Random Forest, Decision Tree, Bagging and Naive Bayes classifiers, with K-Fold, Leave One Out and Holdout cross-validation methods. It had a higher accuracy, but was limited to only 4 of 11 instruments in total. It used the same dataset, albeit not fully, which had an impact on accuracy.

4. Methodology

I used K-Nearest Neighbours and Random Forest Classification algorithms. It used 97 nearest neighbours when it was to predict the class, with Minkowski distance, which is a generalization of both Euclidean and Manhattan distances. Random Forest algorithm uses multiple decision trees which were constructed based on data and Gini impurities. As a prediction class it outputs the class which was most predicted across all decision trees.

For cross-validation, I used Leave One Out, K-Fold and Holdout methods. In order to measure the accuracy of both algorithms, I used Precision, Recall and F1 Score.

For features, I used Mel-Frequency Cepstral Coefficients, in total 20 of them.

More details about the concepts, implementation and code can be found at this link: <https://github.com/gagi3/instrecognition-tool/blob/master/IRT.ipynb>

5. Discussion

KNN and Random Forest Classification algorithms performed similarly on testing data, in terms of precision. For both, it was around 0.46, but recall and F1 score were a bit higher for KNN. However, cross-validation showed that Random Forest Classification was potentially a better fit for this particular problem, because the mean accuracy for cross-validation methods was significantly higher.

Further improvements could be made by improving the feature selection process. Mel-Frequency Cepstral Coefficients perhaps are not the best choice for solely representing the input data. By increasing the number of features, we increase dimensionality which will bring problems on its own. But by decreasing the number of features, we risk having features that are very similar and could not therefore be used for training the model successfully.

Another possible problem with this solution lies in the way training and testing data differ from one another. Training data consists of 2-second single-instrument audio samples, but testing data consists of longer (and sometimes multi-instrument) samples. If we were to split a single testing sample into individual 2-second samples, and then performing testing on each, with the output class being the class that was predicted the most within the individual 2-second samples, we might get better predictions and improve the overall precision.

Bootstrap aggregating (bagging) meta-algorithm could also be used for improving the prediction accuracy, in addition to Random Forest Classification.

6. References

<https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier>

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier>

<https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html#sklearn.model_selection.KFold>

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html>

<https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html>

<https://librosa.github.io/librosa/generated/librosa.feature.mfcc.html?highlight=mfcc#librosa.feature.mfcc>