Bird Classification using Ensemble Classifiers

Leng Yi Ren[1], Jonathan William Dennis[1], and Tran Huy Dat[1]

Institute for Infocomm Research, A*STAR, Singapore,
yrleng@i2r.a-star.edu.sg, jonathan-dennis@i2r.a-star.edu.sg,
hdtran@i2r.a-star.edu.sg

Abstract. This working note summarizes our submission to the LifeCLEF 2014 Bird Task which combines the outputs from a Python and Matlab classification system. The features used for both systems include Mel-Frequency Cepstral Coefficients (MFCC), time-averaged spectrograms and the provided meta-data. The Python subsystem combines a large ensemble of different classifiers with different subsets of the features while the Matlab subsystem is an ensemble of the Random Forest and Linear Discriminant Analysis (LDA) classifiers using local spectral and meta features. By combining this disparate set of features and classifiers, we managed to achieve a Mean Average Precision (MAP) score that is far superior to what is possible with any single classifier.

Keywords: ensemble,audio,classification

1 Introduction

LifeCLEF 2014 Bird Task (BirdCLEF)[1] involves 14027 audio recordings containing 501 bird species. In comparison, the 9th annual MLSP competition[3] involves 645 recordings with 19 bird species while the Neural Information Processing Scaled for Bioacoustics (NIPS4B) bird song competition[4] involves 1000 recordings with 87 bird species. The greatly increased size in both the available data and the number of target classes for BirdCLEF introduces new challenges.

The total size of the audio database for both the MLSP and NIPS4B challenges amounts to around 1GB while the BirdCLEF data is over 20GB in size. The winning methods for the MLSP and NIPS4B challenges performed image segmentation on the audio spectrograms followed by template matching to derive class models. Based on the difference in data size and the number of classes, it can be extrapolated that using the same methods for BirdCLEF will require at least a hundred fold increase in computation time. Given that the time between the release of the training data and the final test submission is only around three months, it would not be efficient to try to reproduce the above methods given our limited computing resources.

All of our experiments on BirdCLEF are performed on standard 4-core desktop PCs with 8GB of RAM thus there are severe constraints on the possible machine learning methods that can be applied. Instead of attempting to build a strong learner using well-engineered features, we chose to work on building a

number of weak learners with simple features and finding the optimal combination of such learners.

2 Front-end

We make use of both audio and meta data for our feature front-end as they provide complementary information to the classifier. The total memory used in the computation is heavily dependent on the size of the feature used thus it is important to select a compact representation.

2.1 MFCC

The MFCC features provided by the organizers consist of a single log energy coefficient and 15 cepstral coefficients, appended with the first and second derivatives for a total of 48 dimensions per time frame. For each audio clip, we perform a simple energy-based segmentation by dropping all the time frames where the value of the log energy coefficient is below the median value. The remaining frames are further reduced by taking the maximum value for each of the 48 dimensions to arrive at a final 48 dimension feature.

2.2 Spectral features

The spectral features we used are similar to the spectrum-summarizing features described in [3]. The audio clips are converted into spectrograms by windowing the audio into overlapping 200ms windows at 100ms intervals. The frequency dimension is binned into n dimensions using a simple mean. The time dimension is eliminated by taking the maximum value similar to the MFCC feature. Finally, the natural logarithm is applied to the n dimension spectral features to reduce its dynamic range.

2.3 Meta features

In addition to the audio features described above, we also make use of the meta-data represented by the following eight fields:

$$Latitude, Longitude, Elevation, Year, Month, Month + Day, Time, Author$$

The $Month + Day$ field combines the two-digit $Month$ and Day fields into a single four-digit field as comparing the Day fields across different months is illogical.

One of the challenges of BirdCLEF compared to MLSP which also provides meta-data in the form of a numerical location index[3] is the presence of unreliable and missing data. The completion of the meta-data fields by the BirdCLEF data contributors is optional with no enforced format. Unreliable information such as “afternoon” or “am” in the $Time$ field and completely missing entries have to be manually parsed into logical numeric values for use as features and their presence will affect the performance of the classifiers.

3 Back-end

Although the official labels for the audio data only gives a single bird species as the ground truth given by *ClassId*, the *BackgroundSpecies* field often list additional bird species. However, like the meta-data, this field is unreliable as there is no indication if no additional species are present or identified, or if the author simply did not complete this particular field. We can only assume that an empty *BackgroundSpecies* field indicates that only the single species given by *ClassId* is present.

Another problem that arises due to the *BackgroundSpecies* field is the presence of bird species that are outside of the 501 species to be classified. For the given training data, there are over three hundred additional species and we chose to retain the sixteen extra species that have over eleven examples each for a total of 517 target species. The remaining labels are ignored since there are insufficient examples to train proper models for them. By making use of the extra information from *BackgroundSpecies*, the number of examples for the 501 original target classes is also increased and the task becomes a multi-label classification problem.

3.1 Python

The classifiers used in the Python subsystem are built from the scikit-learn toolkit (*sklearn*)[5]. In order to build an ensemble of classifiers, we require the classifiers to output probability estimates for each of the target classes so that the final output is simply a mean of the individual classifier outputs. The subsequent computation of the Mean Average Precision (MAP) is also straightforward with such class probability outputs as opposed to max voting. Out of the available classifiers in the toolkit, we selected the following list to evaluate:

```
ExtraTreesClassifier, RandomForestClassifier,
KNeighborsClassifier, LogisticRegression, SGDClassifier,
AdaBoostClassifier, GradientBoostingClassifier, SVC, GaussianNB,
BernoulliNB, LDA
```

From the above list, only $ExtraTreesClassifier$, $RandomForestClassifier$ and $KNeighborsClassifier$ are able to handle multi-label classification by default. The $OneVsRestClassifier$ is used to transform the remaining binary classifiers into multi-label multi-class classifiers.

3.2 Matlab

Two classifiers are used in the Matlab subsystem: Random Forests using the $TreeBagger$ function and LDA using the $ClassificationDiscriminant.fit$ function, both of which were trained in a one-vs-rest configuration. Both of these classifiers output probability estimates through the *predict* function and the final output is the mean of the individual classifier outputs. The $TreeBagger$

function was modified such that each tree in the forest was trained with a balanced amount of negative data, as this was found to considerably improve the speed of the classification without affecting performance. In particular, each tree was trained with all of the positive class data, and a random subset of the negative data containing twenty times as much as the positive data. The number of variables to sample at each node was set to 3, while the number of trees was set to 200 for the final evaluation run.

4 Experiment Setup

To evaluate the performance of a feature and classifier combination, 10-fold cross-validation is used on the given training data. The objective is to maximize the MAP score of the system.

4.1 Python subsystem

From the list of feature front-ends and the selected classifiers from *sklearn*, combinations of feature and classifier pairs are evaluated. This step took up the bulk of our time as it involves the optimizing of feature settings, classifier settings and the combinations of the two to build the final ensemble classifier. Out of the available classifiers, the following six were selected:

1. Linear Discriminant Analysis (LDA), LDA in *sklearn*
2. Logistic Regression (LR), $LogisticRegression$ in *sklearn*
3. Support Vector Machines (SVM) with Radial Basis Function (RBF) kernel, SVC in *sklearn*
4. AdaBoost (AdaB), $AdaBoostClassifier$ in *sklearn*
5. K-Nearest Neighbours (KNN), $KNeighborsClassifier$ in *sklearn*
6. Random Forests (RanF), $RandomForestClassifier$ in *sklearn*

For the feature front-end, three combinations were selected:

1. $Meta + MFCC + \Delta MFCC + Spec10 + Spec100_{30}$, used with LDA, LR and SVM
2. $Meta + MFCC + Spec100_{30}$, used with AdaB and KNN
3. $Meta$, used with RanF

$MFCC$ refers to the first 16 dimensions of the described 48 dimension feature while $\Delta MFCC$ are the next 16 dimensions (indices 17 to 32) which represents the first derivatives or deltas. $Spec10$ refers to the spectral features with $n = 10$ and $Spec100_{30}$ are the first 30 dimensions of the spectral features with $n = 100$. $Meta$ refers to the eight dimension meta features.

4.2 Matlab subsystem

Only one feature configuration is used with the Matlab subsystem. Using the same terminology as above, the features can be denoted $Meta + Spec40$, with $n = 40$ dimensions used for the spectral feature concatenated with the eight dimensions of the meta features. However, instead of taking the maximum over the whole time frame, as with the Python subsystem, local spectral features are generated by taking the maximum within selected local windows. Each window is 0.5 seconds long, and the step is equivalent to the frame increment. The energy of each window is calculated simply as the average log-energy over the window, and is then used as a metric for selecting the windows for classification. Only those windows that are represent local maxima in the energy function are retained, with the remaining windows discarded. The idea is that the windows containing local maxima in the energy will correspond to local instances of the bird song in the clip, hence provide a complimentary source of information to the global features used in the Python subsystem.

5 Results and Discussion

Although we train the classifiers on 517 target species, the results presented only consider the actual 501 target species (MAP with $k = 501$).

5.1 Python subsystem

The results shown in Table 1 give a rough breakdown of the individual components of the Python subsystem. For the sake of brevity, the other combinations of the six selected classifiers are not shown. The *Audio* and $Meta$ columns show how each classifier performs when the complementary meta or audio features are removed. For all but the KNN classifier, combining both audio and meta

Table 1. Selected results for 10-fold cross-validation for Python subsystem

LDA	LogR	SVM	AdaB	KNN	RanF	Audio	Meta	Both
x						0.191	0.075	0.220
	x					0.197	0.076	0.232
		x				0.199	0.085	0.236
			x			0.085	0.110	0.133
				x		0.130	0.177	0.164
					x	-	0.192	-
x	x	x	x	x		0.220	0.195	0.274
x				x	x	-	0.195	0.290
x	x	x	x	x	x	-	0.200	0.313

features is shown to improve the classifier performance. Although the KNN classifier is shown to give a better MAP score with the meta feature alone, when combined with the other classifiers in an ensemble, adding the audio features is found to result in a better score overall. It is interesting to note that using the 8-dimension meta features alone with the chosen classifiers can give a reasonable score of 0.200, highlighting the value of using meta-data for classification tasks.

Table 2 shows the approximate computation time for the Python subsystem. One of our objectives is to design a system with a short turnaround time as the search space for the system parameters is huge. With each new dataset, it is likely that the optimal system configuration in terms of feature and classifier choices will be different thus it will be necessary to retune the entire system from scratch. There is a large discrepancy in the computation time required for

Table 2. Approximate computation time for 10-fold cross-validation for Python subsystem components

LDA	LogR	SVM	AdaB	KNN	RanF	Total
5 min	15 min	1h 10 min	11 min	17 sec	8 min	1h 45 min

each of the classifiers chosen that is partly attributed to the different feature combinations used. The strength of using an ensemble of classifiers is the option to mix and match different combinations of the components to suit particular demands for performance and computation cost. The three fastest classifiers (LDA, KNN and RanF) combine to give an ensemble which runs in less than 15 minutes with a respectable score of 0.290. In contrast, removing RanF from the six classifier ensemble only saves 8 minutes but reduces the score from 0.313 to 0.274. There are no general rules determining the performance of each specific combination, especially if the dataset is changed, thus it is usually necessary to perform an exhaustive search to find the optimal system.

Another strength of the ensemble classifier is how the individual classifiers complement each other. For most of the classifiers tested, it is possible to improve the individual score at the cost of computation time by changing parameters such as increasing the number of iterations or the number of neighbours to query. However, these improvements do not necessarily transfer to the ensemble classifier when they are combined as the individual gains are made obsolete through the contributions of the other classifiers. Depending on the classifiers used, it is often possible to replicate a computation-intensive performance gain from a classifier.

5.2 Matlab subsystem

The results shown in Table 3 give a breakdown of the individual components of the Matlab subsystem using the combined $Audio + Meta$ features. It can be seen that the LDA classifier is performing relatively poorly, perhaps due the

Table 3. Classification results for 10-fold cross-validation for the Matlab subsystem

LDA	RanF	Audio + Meta
x		0.119
	x	0.216
x	x	0.222

number of false positives in the features. These false positives arise from using local features that are extracted from local maxima in the energy, and hence may represent noise or other information that is not related to the target bird species. On the other hand, the random forest classifier is able to select the features that best separate the target classes, and this allows it to handle a large number of false positives in the labelled data. Combining the two classifiers together increases the score, giving an MAP of 0.222 on the cross-validation.

5.3 Combined system

Table 4. Final results for proposed system

	Python	Matlab	Combined
with *BackgroundSpecies*	0.284	0.166	0.289
without *BackgroundSpecies*	0.267	0.159	0.272

Due to time constraints, we did not manage to evaluate the performance of the combined Python and Matlab system through cross-validation. Table 4 shows the final results for our proposed systems on the evaluation data. Compared to Table 1, the Python subsystem shows a 10% relative loss with the inclusion of *BackgroundSpecies*. This might be due to the use of a lower k value when evaluating the MAP score or a mismatch between the training and evaluation data. The combined system is found to improve on the Python subsystem score slightly despite the poor performance of the Matlab subsystem.

Our system was trained as a multi-label classifier and this is reflected by the lower score when the background species are removed (MAP 2). Interestingly, we are the only group that has a lower MAP 2 score than MAP 1, indicating that we are the only group that focussed only identifying the background species as well as the dominant species.

6 Conclusion

The challenges presented by BirdCLEF include the large amount of provided data, the large number of target species to be classified, incomplete or inaccurate multi-class labels and missing and unreliable meta-data. The limited time

between the release of the training data and the final submission and the limited computing resources available to our group are additional obstacles we faced. With these limitations in mind, we designed an ensemble classifier that combines the output from a number of individually weak learners using simple features that are fast to train and test. The final system is shown to provide greatly improved results over the component classifiers and the fast turnaround time allows the setup to be quickly calibrated for new tasks and datasets.

References

1. Goëau, H., Glotin, H., Vellinga, W. and Rauber, A.: LifeCLEF Bird Identification Task 2014. CLEF working notes 2014
2. Cappellato, L., Ferro, N., Halvey, M., and Kraaij, W., editors (2014): CLEF 2014 Labs and Workshops, Notebook Papers. CEUR Workshop Proceedings (CEUR-WS.org), ISSN 1613-0073, http://ceur-ws.org/ Vol-1180/
3. Briggs, F., et al.: The 9th annual MLSP competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. Machine Learning for Signal Processing (MLSP), 2013 IEEE International Workshop on, pp.1,8, 22-25 Sept. 2013
4. Glotin H, LeCun Y, Artières T, Mallat S, Tchernichovski O, et al: Proc. of Neural Information Processing Scaled for Bioacoustics: from Neurons to Big Data - NIP4B, 2013
5. Pedregosa, F., et al.: Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, vol 12, 2825–2830, 2011