Audio-only bird classification using unsupervised feature learning

Dan Stowell and Mark D. Plumbley

Centre for Digital Music, Queen Mary University of London
dan.stowell@qmul.ac.uk

Abstract. We describe our method for automatic bird species classification, which uses raw audio without segmentation and without using any auxiliary metadata. It successfully classifies among 501 bird categories, and was by far the highest scoring audio-only bird recognition algorithm submitted to BirdCLEF 2014. Our method uses unsupervised feature learning, a technique which learns regularities in spectro-temporal content without reference to the training labels, which helps a classifier to generalise to further content of the same type. Our strongest submission uses two layers of feature learning to capture regularities at two different time scales.

1 Introduction

Automatic species classification of birds from their sounds has many potential applications in conservation, ecology and archival [11, 6]. However, to be useful it must work with high accuracy across large numbers of possible species, on noisy outdoor recordings and at big data scales. The ability to scale to big data is crucial: remote monitoring stations can generate huge volumes of audio recordings, and audio archives contain large volumes of audio, much of it without detailed labelling. Big data scales also imply that methods must work without manual intervention, in particular without manual segmentation of recordings into song syllables, or into vocal/silent sections. The lack of segmentation is a pertinent issue for both remote monitoring and archive collections, since many species of bird may be audible for only a minority of the recorded time, and therefore much of the audio will contain irrelevant information.

In this paper we describe a method for dramatically improving the performance of a supervised classifier for bird sounds, as submitted to the "BirdCLEF" 2014 evaluation contest. The method achieved the strongest performance among audio-only methods (i.e. methods that did not use additional information such as date or location).

Our method is the subject of a full-length article which can be read at [15]. In the following shorter presentation, we describe the method, which works on raw audio with no segmentation. We describe how we evaluated variants of our method and chose which variants to submit, and we consider the run-time of different stages of the workflow.

1.1 Spectral features and feature learning

For classification, audio data is often converted to a spectrogram-like representation, i.e. the magnitudes of short-time Fourier transformed (STFT) frames of audio, around 10 ms duration per frame. It is common to transform the frequency axis to a more perceptual scale, such as the Mel scale originally intended to represent the approximately logarithmic sensitivity of human hearing. This also reduces the dimensionality of the spectrum, but even the Mel spectrum has traditionally been considered rather high-dimensional for automatic analysis. A further convention, originating from speech processing, is to transform the Mel spectrum using a cepstral analysis and then to keep the lower coefficients (e.g. the first 13) which typically contain most of the energy. These coefficients, the Mel frequency cepstral coefficients (MFCCs), became widespread in applications of machine learning to audio, including bird vocalisations [13].

MFCCs have some advantages, including that the feature values are approximately decorrelated from each other, and they give a substantially dimension-reduced summary of spectral data. Dimension reduction is advantageous for manual inspection of data, and also for use in systems that cannot cope with high-dimensional data. However, as we will see, modern classification algorithms can cope very well with high-dimensional data, and dimension reduction always reduces the amount of information that can be made available to later processing, risking discarding information that a classifier could have used. Further, there is little reason to suspect that MFCCs should capture information optimal for bird species identification: they were designed to represent human speech, yet humans and birds differ in their use of the spectrum both perceptually and for production. MFCCs aside, one could use raw (Mel-)spectra as input to a classifier, or one could design a new transformation of the spectral data that would tailor the representation to the subject matter. Rather than designing a new representation manually, we consider automatic feature learning.

The topic of feature learning (or representation learning, dictionary learning) has been considered from many perspectives within the realm of statistical signal processing [1][10][4] [5]. The general aim is for an algorithm to learn some transformation that, when applied to data, improves performance on tasks such as sparse coding, signal compression or classification. This procedure may be performed in a "supervised" manner, meaning it is supplied with data as well as some side information about the downstream task (e.g. class labels), or "unsupervised", operating on a dataset but with no information about the downstream task. A simple example that can be considered to be unsupervised feature learning is principal components analysis (PCA): applied to a dataset, PCA chooses a linear projection which ensures that the dimensions of the transformed data are decorrelated [1]. It therefore creates a new feature set, without reference to any particular downstream use of the features, simply operating on the basis of qualities inherent in the data.

Recent work in machine learning has shown that unsupervised feature learning can lead to representations that perform very strongly in classification tasks, despite their ignorance of training data labels that may be available [4, 1]. This

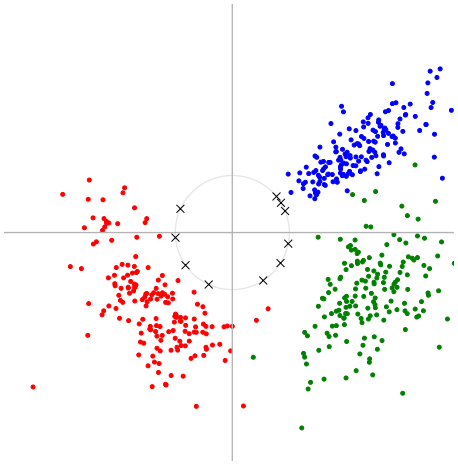

Fig. 1. Example of spherical k-means applied to a simple two-dimensional dataset. We generated synthetic 2D data points by sampling from three clusters which were each Gaussian-distributed in terms of their angle and log-magnitude (coloured dots), and then applied our online spherical k-means algorithm to find 10 unit vectors (crosses). These unit vectors form an overcomplete basis with which one could represent this toy data, projecting two-dimensional space to ten-dimensional space.

rather surprising outcome suggests that feature learning methods emphasise patterns in the data that turn out to have semantic relevance, patterns that are not already made explicit in the basic feature processing such as STFT.

Birdsong often contains rapid temporal modulations, and this information should be useful for identifying species-specific characteristics. From this perspective, a useful aspect of feature learning is that it can be applied not only to single spectral frames, but to short sequences (or "patches") of a few frames. The representation can then reflect not only characteristics of instantaneous frequency patterns in the input data, but characteristics of frequencies and their short-term modulations, such as chirps sweeping upwards or downwards. This bears some analogy with the "delta-MFCC" features sometimes used by taking the first difference in the time series of MFCCs, but is more flexible since it can represent amplitude modulations, frequency modulations, and correlated modulations of both sorts. In our study we tested variants of feature learning with different temporal structures: either considering one frame at a time (which does not capture modulation), multiple frames at a time, or a variant with two layers of feature learning, which captures modulation across two timescales.

2 Method

As discussed in Section 1.1, the aim of unsupervised feature learning is to find some transformation of a dataset, driven only by the characteristics inherent in that dataset. For this we use a method that has has shown promise in previous studies, and can be run effectively at big data scales: *spherical k-means*, described by [4] and first applied to audio by [5]. Spherical k-means is related to the well-known k-means clustering algorithm, except that instead of searching for cluster centroids which minimise the Euclidean distance to the data points, we search for unit vectors (directions) to minimise their angular distance from the data points. This is achieved by modifying the iterative update procedure for the k-means algorithm: for an input data point, rather than finding the nearest centroid by Euclidean distance and then moving the centroid towards that data point, the nearest centroid is found by cosine distance,

$$\text{cosine distance} = 1 - \cos(\theta) = 1 - \frac{A \cdot B}{\|A\|\|B\|}\,, \tag{1}$$

where A and B are vectors to be compared, θ is the angle between them, and $\|\cdot\|$ is the Euclidean vector norm. The centroid is renormalised after update so that it is always a unit vector. Fig. 1 shows an example of spherical k-means applied to synthetic data. Spherical k-means thus finds a set of unit vectors which represent the distribution of directions found in the data: it finds a basis (here an overcomplete basis) so that data points can in general be well approximated as a scalar multiple of one of the basis vectors. This basis can then be used to represent input data in a new feature space which reflects the discovered regularities, in the simplest case by representing every input datum by its dot product with each of the basis vectors [4, 5]:

$$x'(n,j) = \sum_{i=1}^{M} b_j(i)x(n,i)\,, \tag{2}$$

where x represents the input data indexed by time frame n and feature index i (with M the number of input features, e.g. the number of spectral bins), b_j is one of the learnt basis vectors (indexed by $j \in [1,k]$), and x' is the new feature representation. In our case, the data on which we applied the spherical k-means procedure consisted of Mel spectral frames ($M = 40$ dimensions), which we first normalised and PCA-whitened as in [5].

We also tested configurations in which the input data was not one spectral frame but a sequence of them—e.g. a sequence of four spectral frames at a time—allowing the clustering to respond to short-term temporal patterns as well as spectral patterns. We can write this as

$$x'(n,j) = \sum_{\delta=0}^{\Delta}\sum_{i=1}^{M} b_j(\delta,i)x(n+\delta,i)\,, \tag{3}$$

where Δ is the number of frames considered at a time, and the b are now indexed by a frame-offset as well as the feature index. Alternatively, this can be thought of

as "stacking" frames, e.g. stacking each sequence of four 40-dimensional spectral frames to give a 160-dimensional vector, before applying (2) as before. In all our experiments we used a fixed $k = 500$, a value which has been found useful in previous studies [5].

The standard implementation of k-means clustering requires an iterative batch process which considers all data points in every step. This is not feasible for high data volumes. Some authors use "minibatch" updates, i.e. subsamples of the dataset. For scalability as well as for the potential to handle real-time streaming data, we instead adapted an online streaming k-means algorithm, "online Hartigan k-means" [12, Appendix B]. This method takes one data point at a time, and applies a weighted update to a selected centroid dependent on the amount of updates that the centroid has received so far. We adapted the method of [12, Appendix B] for the case of spherical k-means. K-means is a local optimisation algorithm rather than global, and may be sensitive to the order of presentation of data. Therefore in order to minimise the effect of order of presentation for the experiments conducted here, we did not perform the learning in true single-pass streaming mode. Instead, we performed learning in two passes: a first streamed pass in which data points were randomly subsampled (using reservoir sampling) and then shuffled before applying PCA whitening and starting the k-means procedure, and then a second streamed pass in which k-means was further trained by exposing it to all data points. Our Python code implementation of online streaming spherical k-means is available on request.

As a further extension we also tested a *two-layer* version of our feature-learning method, intended to reflect detail across multiple temporal scales. In this variant, we applied spherical k-means feature learning to a dataset, and then projected the dataset into that learnt space. We then downsampled this projected data by a factor of 8 on the temporal scale (by max-pooling, i.e. taking the max across each series of 8 frames), and applied spherical k-means a second time. The downsampling operation means that the second layer has the potential to learn regularities that emerge across a slightly longer temporal scale. The two-layer process overall has analogies to deep learning techniques, most often considered in the context of artificial neural networks [1], and to the progressive abstraction believed to occur towards the higher stages of auditory neural pathways.

2.1 Classification workflow

Our full classification workflow started by converting each audio file to a standard sample-rate of 44.1 kHz. We then calculated Mel spectrograms for each file, using a frame size of 1024 frames with Hamming windowing and no overlap. We filtered out spectral energy below 500 Hz, a common choice to reduce the amount of environmental noise present, and then normalised the root-mean-square (RMS) energy in each spectrogram.

For each spectrogram we then optionally applied the noise-reduction procedure that we had found to be useful in our NIPS4B contest submission [14], a simple and common median-based thresholding. The Mel spectrograms, either noise-reduced or otherwise, could be used directly as features. We also tested

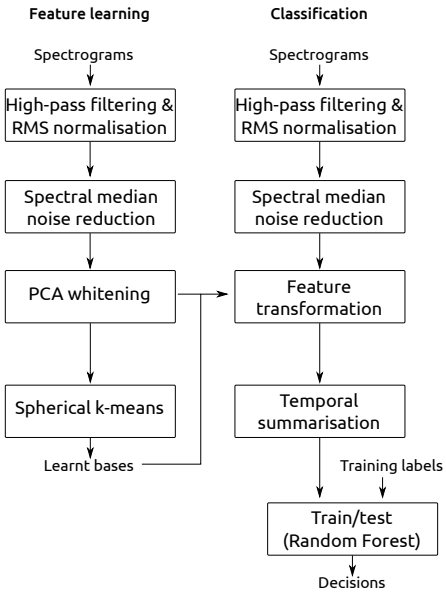

Fig. 2. Summary of the classification workflow, here showing the case where single-layer feature learning is used.

their reduction to MFCCs (including delta features, making 26-dimensional data), and their projection onto learned features, using the spherical k-means method described above. For the latter option, we tested projections based on single frame as well as on sequences of 2, 3, 4 and 8 frames, to explore the benefit of modelling short-term temporal variation. We also tested the two-layer version based on the repeated application to 4-frame sequences across two timescales.

The feature representations thus derived were all time series. In order to reduce them to summary features for use in the classifier, we used the simple approach of summarising each feature dimension independently by its mean and standard deviation. These are widespread but are not designed to reflect any temporal structure in the features; however, note that our multi-frame learnt features intrinsically capture some fine-scale temporal information.

To perform classification on our temporally-pooled feature data, then, we used a random forest classifier [2]. Random forests and other tree-ensemble classifiers perform very strongly in a wide range of empirical evaluations [3], and were used by many of the strongest-performing entries to the SABIOD evaluation contests [8, 7]. For this experiment we used the implementation from the Python `scikit-learn` project. Note that `scikit-learn` v0.14 was found to have a specific issue preventing training on large data, so we used a pre-release v0.15 after verifying that it led to the same results with our smaller datasets. We did

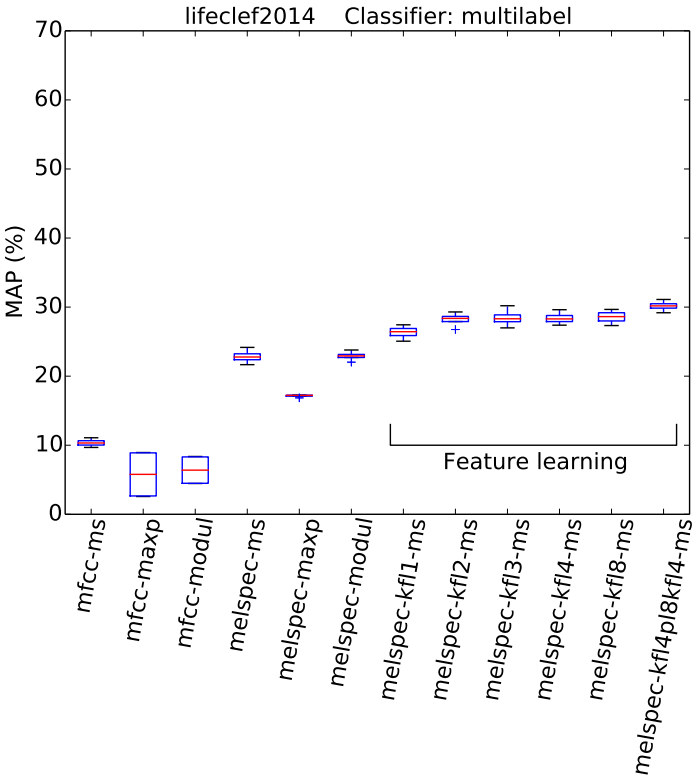

Fig. 3. MAP statistics, summarised for each feature-type tested. Each column in the boxplot summarises the crossvalidated scores attained over many combinations of the other configuration settings tested (for the full multi-class classifier only). The ranges indicated by the boxes therefore do not represent random variation due to training data subset, but systematic variation due to classifier configuration.

not manually tune any parameters of the classifier. Fig. 2 summarises the main stages of the workflow described.

Prior to the contest, we did not have access to the held-out testing data, so for evaluation we split the training dataset into two equal partitions and performed two-fold crossvalidation. We also tested model averaging: namely, we tested a meta-classifier which simply averaged over the outputs from up to 16 different configurations of our main system.

3 Results

Figure 3 illustrates the cross-validated MAP scores we obtained from six non-learnt feature representations (based on simple MFCC and Mel spectra) and from six learnt feature representations. The learnt representations consistently and

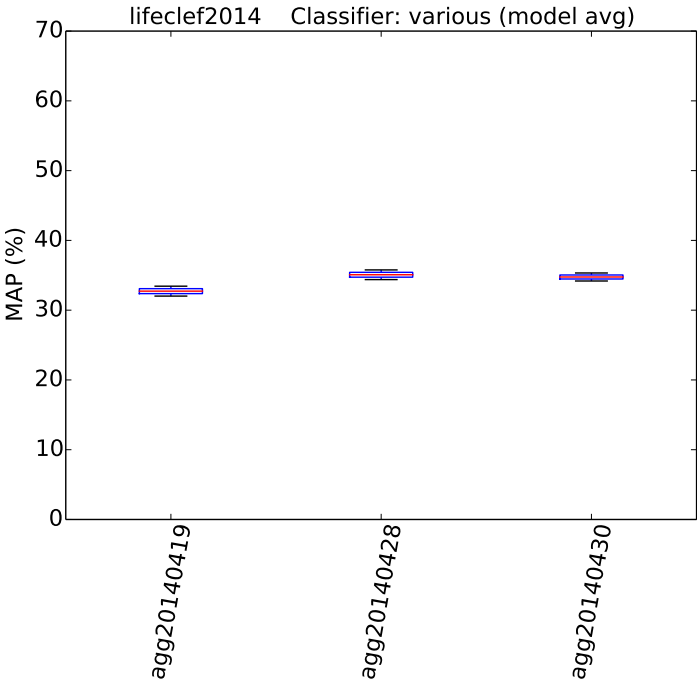

Fig. 4. As Figure 3, but for the model-averaging runs that we tested. Each column represents a different set of models included in the averaging.

strongly outperformed the other approaches. The difference between the learnt representations was small in terms of performance, although for this data it is notable that the *two-layer* variant (rightmost column) consistently outperformed the single-layer variants. Even though the BirdCLEF challenge is a single-label classification challenge, we found that training the random forest as a multilabel classifier gave slightly better results.

Model averaging yielded improved performance in general (Figure 4), although it was not clear from our own tests whether model averaging or a single classifier would achieve the highest point performance.

We measured the total time taken for each step in our workflow, to determine the approximate computational load for the steps (Fig. 5). The timings are approximate—in particular because our code was modularised to save/load state on disk between each process, which impacted particularly on the "classify" step which loaded large random forest settings from disk before processing. Single-layer feature learning was efficient, taking a similar amount of time as did the initial feature extraction. Double-layer feature learning took more than double this, because of the two layers as well as performing max-pooling downsampling. Training the random forest classifier took longer on the learned features due to

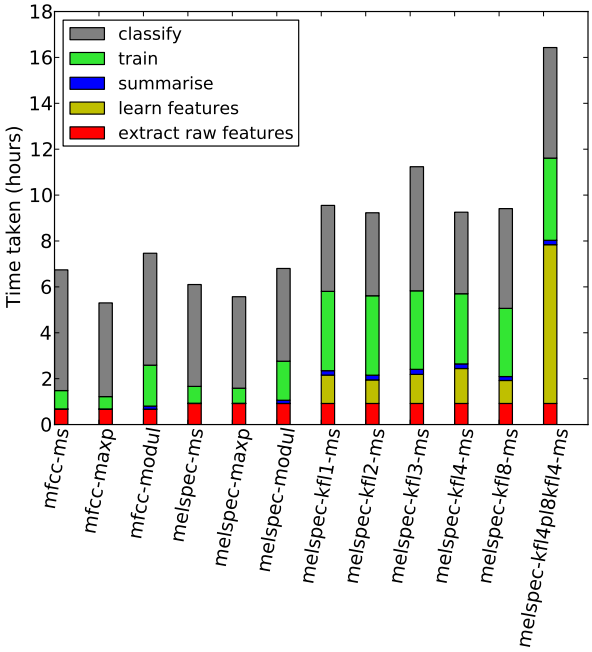

Fig. 5. Times taken for each step in the process. Note that these are heuristic "wall-clock" times measured on processes across two compute servers. Each measurement is averaged across the two folds and across two settings (noise reduction on/off) across the runs using the multilabel classifier and no decision-pooling.

the higher dimensionality. However, once the system was trained, the time taken to classify new data was the same across all configurations.

We submitted decisions from our system to the LifeCLEF 2014 bird identification challenge [9]. In that evaluation, our system attained the strongest audio-only classification results, with a MAP peaking at 42.9% (Table 1, Figure 6), ten percentage points stronger than other audio-only classifiers that were submitted. (Only one system outperformed ours, peaking at 51.1% in a variant of the challenge which provided additional metadata as well as audio.) We submitted the outputs from individual models, as well as model-averaging runs using the simple mean of outputs from multiple models. Notably, the strongest classification both in our own tests and the official evaluation was attained not by model averaging, but by a single model based on two-layer feature learning. Also notable is that our official scores, which were trained and tested on larger data subsets, were substantially higher than our crossvalidated scores, corroborating our observation that the method works particularly well at high data volumes.

System variant submitted	Cross-val MAP (%)	Official MAP (%)
`melspec-kfl3-ms`, noise red., binary relevance	30.56	36.9
Average from 12 single-layer models	32.73	38.9
`melspec-kfl4pl8kfl4-ms`, noise red., binary relevance	**35.31**	**42.9**
Average from 16 single- and double-layer models	35.07	41.4

Table 1. Summary of MAP scores attained by our system in the public LifeCLEF 2014 Bird Identification Task [9]. The first column lists scores attained locally in our two-fold split. The second column lists scores evaluated officially, using a classifier(s) trained across the entire training set.

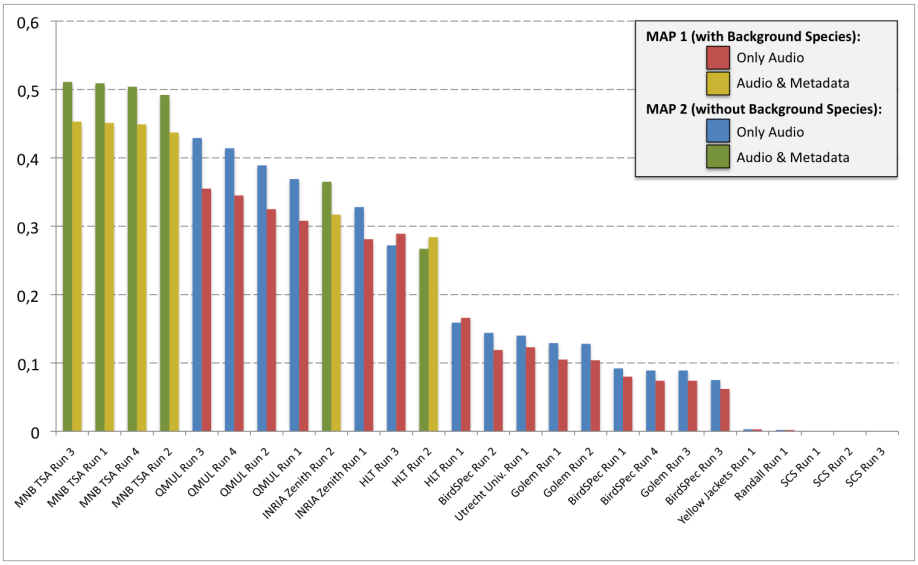

Fig. 6. Official plot of evaluation scores. Ours are the four labelled "QMUL".

4 Conclusions

Current interest in automatic classification of bird sounds is motivated by the practical scientific need to label large volumes of data coming from sources such as remote monitoring stations and sound archives. Unsupervised feature learning is a simple and effective method to boost classification performance by learning spectro-temporal regularities in the data. It does not require training labels or any other side-information, it can be used within any classification workflow, and once trained it imposes negligible extra computational effort on the classifier. The principal practical issue with unsupervised feature learning is that it requires large data volumes to be effective. However, this exhibits a synergy with the large data volumes that are increasingly becoming standard.

In our experiments, learnt features strongly outperformed MFCC and Mel spectral features. In the BirdCLEF 2014 contest, our system was by far the

strongest audio-only submission, outperforming even some systems which made use of auxiliary metadata.

Acknowledgments

We would like to thank the people and projects which made available the data used for this research—the Xeno Canto website and its many volunteer contributors—as well as the SABIOD research project for instigating the contest, and the CLEF contest hosts.

This work was supported by EPSRC Leadership Fellowship EP/G007144/1 and EPSRC Early Career Fellowship EP/L020505/1.

References

1. Bengio, Y., Courville, A., Vincent, P.: Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence 35(8), 17981828 (2013)
2. Breiman, L.: Random forests. Machine Learning 45(1), 5–32 (2001)
3. Caruana, R., Niculescu-Mizil, A.: An empirical comparison of supervised learning algorithms. In: Proceedings of the 23rd international conference on Machine learning. pp. 161–168. ACM (2006)
4. Coates, A., Ng, A.Y.: Learning feature representations with k-means. In: Neural Networks: Tricks of the Trade, pp. 561–580. Springer (2012)
5. Dieleman, S., Schrauwen, B.: Multiscale approaches to music audio feature learning. In: Proceedings of the International Conference on Music Information Retrieval (ISMIR 2013) (2013)
6. Digby, A., Towsey, M., Bell, B.D., Teal, P.D.: A practical comparison of manual and autonomous methods for acoustic monitoring. Methods in Ecology and Evolution 4(7), 675–683 (2013)
7. Fodor, G.: The ninth annual MLSP competition: First place. In: Proceedings of the International Conference on Machine Learning for Signal Processing (MLSP 2013). p. 2pp. IEEE (2013)
8. Glotin, H., LeCun, Y., Artières, T., Mallat, S., Tchernichovski, O., Halkias, X. (eds.): Neural Information Processing Scaled for Bioacoustics, from Neurons to Big Data. USA (2013), http://sabiod.org/NIPS4B2013_book.pdf
9. Goëau, H., Glotin, H., Vellinga, W.P., Rauber, A.: LifeCLEF bird identification task 2014. In: CLEF Working Notes 2014 (2014)
10. Jafari, M., Plumbley, M.D.: Fast dictionary learning for sparse representations of speech signals. IEEE Journal of Selected Topics in Signal Processing 5(5), 1025–1031 (2011)
11. Laiolo, P.: The emerging significance of bioacoustics in animal species conservation. Biological Conservation 143(7), 1635–1645 (2010)
12. McFee, B.: More like this: Machine learning approaches to music similarity. Ph.D. thesis, University of California, San Diego (2012), http://cseweb.ucsd.edu/~bmcfee/papers/bmcfee_dissertation.pdf
13. Stowell, D., Plumbley, M.D.: Birdsong and C4DM: A survey of UK birdsong and machine recognition for music researchers. Tech. Rep. C4DM-TR-09-12, Centre for Digital Music, Queen Mary University of London (Aug 2010), http://c4dm.eecs.qmul.ac.uk/papers/2010/Stowell2010-C4DM-TR-09-12-birdsong.pdf

14. Stowell, D., Plumbley, M.D.: Feature design for multilabel bird song classification in noise (nips4b challenge). In: Proceedings of NIPS4b: Neural Information Processing Scaled for Bioacoustics, from Neurons to Big Data (2013)
15. Stowell, D., Plumbley, M.D.: Automatic large-scale classification of bird sounds is strongly improved by unsupervised feature learning. Pre-print (2014), http://arxiv.org/abs/1405.6524