Learning multi-labeled bioacoustic samples with an unsupervised feature learning approach [*]

Eneldo Loza Mencía
Knowledge Engineering Group
Technische Universität Darmstadt
eneldo@ke.tu-darmstadt.de

Jinseok Nam
Knowledge Engineering Group
Technische Universität Darmstadt
nam@kdsl.informatik.tu-darmstadt.de

Dong-Hyun Lee
sayit78@gmail.com

Abstract

Multi-label Bird Species Classification competition provides an excellent opportunity to analyze the effectiveness of acoustic processing and mutlilabel learning. We propose an unsupervised feature extraction and generation approach based on latest advances in deep neural network learning, which can be applied generically to acoustic data. With state-of-the-art approaches from multilabel learning, we achieved top positions in the competition, only surpassed by teams with profound expertise in acoustic data processing.

1 Introduction

Acoustic data is a common and natural representative of multilabel data, i.e. data in which an example, in this case an acoustic sample, can be mapped to several, non-exclusive classes or categories. Examples are the popular *emotions* benchmark with the objective of assigning emotions to music or the *hifind* dataset where the tasks is to identify used instruments, genres, moods, languages, styles etc. in songs [12]. In the Multi-label Bird Species Classification competition (NIPS4B) the task was to identify 87 birds, insects and amphibians in short audio recordings. Of course, these could appear in the same sample. More specifically, the objective was to maximize the area under the ROC curve (AUC) on 1000 unlabeled recordings, a common measure for the quality of a label ranking.

The challenge was thus two fold: On the one hand it was necessary to process the data in a way appropriate for machine learning approaches, since the data was only available in a raw format or in very basic preprocessed format. This article presents a combination of recent and state-of-the-art approaches from neural network and deep learning which allows an unsupervised generation of an aleatory number of features, appropriate for being processed by standard machine learning algorithms. It basically consists of random patching, a Denoising Autoencoder unit and subsequent convolution and represents a general approach for processing acoustic data.

On the other hand, it was essential to learn the data accurately in order to produce high quality predictions and to get the most out the provided information in form of input feature and binary (relevant/irrelevant) label information. We tried out three approaches: firstly, a pairwise ensemble of SVMs which actually is geared towards the base of AUC, the correct order of pairs of labels. The popular and effective LibSVM library was specifically adapted to allow pairwise learning and the modifications are made available. Secondly and thirdly, random decision trees and a single layer neural network were applied. The diversity of the classifiers ensured that the combination of the

[*] In proc. of int. symposium *Neural Information Scaled for Bioacoustics* joint to NIPS, Nevada, dec. 2013, Ed. Glotin H. et al.

predictions was effective. Our final ranking in the very competitive contest show that our acoustic prepocessing provides a good base for the following machine learning step and that the multilabel learner exhaust this.

2 A Multilabel Bird Species Classification Dataset

The dataset for the Multi-label Bird Species Classification competition contains 687 labeled training examples, including 100 noise samples which are not labeled with any bird species, and 1,000 unlabeled test examples for measuring the generalization performance. The recordings belong to 87 categories (bird species like the subalpine warbler), each of which is associated with approx. 13 training instances. The average labelset size is 2.00 excluding noise samples, maximally 6, and there are 265 distinct labelsets in the training data. The dataset comprises two format: one is in raw wav format in which bird songs and calls are recorded with distant insects, and the other is Mel-Frequency Cepstral Coefficients (MFCCs) of the wav files following preprocessing steps in [4] where each time frame is represented with 17 coefficients. Each audio clip in both train and test data varies in length.

3 Unsupervised Feature Generation and Extraction for Acoustic Clips

Let $\mathbf{s} \in \mathbb{R}^{m \times T}$ be the input vector for an audio clip where T is the total number of frames in time and each time frame t consists of an m dimensional feature vector. In our first attempts, we just padded (repeated) smaller samples so that in the end all samples had the same length $\max_i T_i = 1288$, resulting in 21,896 total number of features (referred to as *raw* dataset). However, the results were not satisfactory, thus we applied the following operations and methods from unsupervised feature learning. These were already successfully applied e.g. on image data, hence the question was whether they would work for acoustic data.

Firstly, we extract M_{tr} and M_{ts} random patches, totally $M = M_{tr} + M_{ts}$, whose size is $psz = m \times wnd$ from training and test data, respectively, where wnd denotes the size of window. An extracted patch is then normalized along the time frame axis which makes each coefficient has zero mean and unit variance. Secondly, the randomly sampled patches are concatenated to form training examples $\Phi \in \mathbb{R}^{psz \times M}$ for Denoising Autoencoder or DAE [14], which learns hidden representations from inputs in an unsupervised way. A DAE is a neural network architecture consisting of *encoder* f_{enc} and *decoder* g_{dec} with parameters $\theta = \{W, b, c\}$ to minimize the squared error loss function $\|\varphi - g_{dec}(W^T f_{enc}(W\tilde{\varphi} + b) + c))\|_2^2$ where $W \in \mathbb{R}^{F \times psz}$ is the weights matrix connecting visible units and hidden units, b and c are biases for hidden units and visible units, and $\tilde{\varphi}$ is the corrupt input by adding Gaussian noise $\mathbf{n} \sim \mathcal{N}(0, \sigma^2)$ to an input φ . Once training DAE is done, each column of the weights W^T acts as a feature detector. F feature detectors can be considered in total, and each feature detector has the same size as the randomly extracted patches, that is, the k^{th} feature detector is $W^T_{\cdot\cdot k} \in \mathbb{R}^{m \times wnd}$. Finally, we can obtain a fixed feature representation for an input signal $\mathbf{s}$ in terms of T while convolving it with learned feature detectors.

$$\mathbf{a}_k = f_{conv}\left(s * W^T_{\cdot\cdot k} + b_k\right) \qquad \mathbf{a}_{k+F} = f_{conv}\left(s * (-W^T_{\cdot\cdot k}) + b_k\right) \tag{1}$$

$$x_k = \sum_{j=1}^{T-wnd+1} a_{k,j} \qquad x_{k+F} = \sum_{j=1}^{T-wnd+1} a_{(k+F),j} \tag{2}$$

where $*$ stands for a 2D discrete convolution operator[1] and f_{conv} is to provide non-linearity to the convolved feature representations. We use ReLUs $f(x) = \max(0, x)$ for nonlinear function f_{conv}. In order to make use of the negative part as well as the positive part of inputs to ReLUs, we apply polarity splitting [2] in Eq. 1. Then, we sum up the convolved feature values $\mathbf{a}_k$ over time which is analogous to accumulated activations of $\mathbf{s}$ with respect to the feture detector $W^T_{\cdot\cdot k}$ (Eq. 2). For instance, x_k will be higher if a feature k defined by $W^T_{\cdot\cdot k}$ is detected many times in $\mathbf{s}$.

For training DAE, we extracted 100,000 17×80 patches randomly from training data and 100,000 from test data in the MFCC format. We then trained two DAE models with Gaussian noise $\sigma = 0.2$

[1]The convolution operator yields a matrix of size $1 \times (T - wnd + 1)$.

on the patches; the models have 400 and 800 hidden units resulting in 800 (*small*) and 2000 (*big*), respectively. We used ReLU for the *encoder* and sigmoid $f(x) = 1/(1+\exp(-x))$ for the *decoder*.

4 Multilabel Learning

Multilabel classification refers to the task of learning a function that maps instances $\mathbf{x}_i \in \mathcal{X}$ to label subsets or label vectors $\mathbf{y}_i = (y_{i,1}, \ldots, y_{i,n}) \in \{0,1\}^n$, where $\mathcal{L} = \{\lambda_1, \ldots, \lambda_n\}$, $n = |\mathcal{L}|$ is a finite set of predefined labels and where each label attribute y_i corresponds to the absence (0) or presence (1) of label λ_i. In the following, we will present the different learning algorithms we applied on the NIPS4b dataset.

Pairwise Support Vector Machines The most common approach for multilabel classification is to use an ensemble of binary classifiers, where each classifier predicts if an instance belongs to one specific class or not (*binary relevance* or BR). An alternative is to do *pairwise decomposition*. Here, one classifier is trained for each pair of classes, i.e., a problem with n different classes is decomposed into $\frac{n(n-1)}{2}$ smaller subproblems [6]. More precisely, for each pair of classes (λ_u, λ_v), $u < v$, we learn a binary base classifier $h_{u,v}$, whose training set is composed of examples for which λ_u is a relevant class and λ_v is an irrelevant class, or vice versa. All other examples are ignored for this particular subproblem [10]. During classification, all of the $\frac{n(n-1)}{2}$ base classifiers make a prediction for one of the both corresponding classes, which is interpreted as a full vote (0 or 1), hence resulting in a full ranking over the labels.[2]

Pairwise learning method is often regarded as superior to BR because it profits from simpler decision boundaries in the subproblems [6, 8]. The reason is that each of the pairwise classifiers contains fewer examples. In fact, it has also been shown that the complexity for training an ensemble of pairwise classifiers is comparable to the complexity of training a BR ensemble [6, 10]. During prediction, however, we have a quadratic number of classifiers we have to evaluate. But particularly for support vector machines this problem is alleviated by the fact that easier (sub-)problems lead to less support vectors and that support vectors can be shared among the pairwise SVMs.

Multilabel LibSVM Because of this and because SVMs trained in a pairwise fashion already obtained state-of-the-art results on standard benchmark datasets [12] in previous works [11], we decided to use the very popular and effective SVM software library LibSVM [1] for training our pairwise SVMs. However, during preliminary experiments, we found out that just plugging in LibSVM was not feasible since a simple experiment on the dataset apparently required more than 20 GB of memory. The reason is that the used Java interface copies every training instance each time for every base learner. Additionally, each of the 3741 LibSVM instantiation could request up to 40 MB of cache. We thus extended LibSVM directly in order to support the pairwise learning of multilabel data. Our extension does not copy a training instance more than once and also shares a common cache for Kernel computations, so that we managed to perform an experiment in less than 250 seconds for training 618 instances and 9 seconds for testing 68 instances and with less than 100 MB of memory (worst cases, respectively) despite the quadratic number of models to be trained, stored and evaluated. The LibSVM modifications and interfaces for the multilabel learning toolkit MULAN [13] are available from http://www.ke.tu-darmstadt.de/resources/multilabellibsvm.

Random Decision Trees Zhang et al. [15] recently proposed to use ensembles of random decision trees (RDTs) for learning multilabel data and also provide a software library.[3] The main idea is to generate k_1 RDTs with random attribute tests at the inner nodes and maximal depth k_2. Comparably small values of k_1 and k_2, around 10 or 20 and maximally 100, are sufficient in their experiments. During the extremely fast training, the leafs incrementally collect statistics about the label distributions $\mathbf{y}$ which passed all tests to the leafs. Hence, each RDT predicts an average distribution, which is subsequently averaged over all trees. RDTs are very suitable for data with a high number of examples and labels, since the costs are bounded by the selection of k_1 and k_2. However, they may have problems with high number of features and particularly sparse features, which is not the case for NIPS4B.

[2]Ties in the final votes counting are broken by using the prior probabilities of the labels.

[3]http://www.dice4dm.com/

Neural Networks with a Single Hidden Layer Neural networks (NNs) have attracted increasing interest in recent years thanks to success of NNs with multiple levels of trainable feature extractors, namely *deep learning* in various domains such as object recognition and speech recognition. In order to achieve state of the arts performance, one usually trains deep neural networks on a large amount of training examples or initialize parameters by pretraining the networks on *unlabeled* data in an unsupervised manner, followed by learning whole parameters including a classification layer using labeled instances. As the bird species classification dataset has only 587 labeled examples and only 11 positive training instances are available per label, on average, we decided to train NNs with a only single hidden layer rather than ones with multiple hidden layers.

The single hidden layer NNs perform surprisingly well when we combine them with AdaGrad [3], which makes it possible to adapt the learning rate per parameter, and Dropout [7] to prevent overfitting and hence improving generalization performance. The output $\hat{\mathbf{y}}$ of NNs for a given training example $\mathbf{x}$ is computed by using the following composition of non-linear functions $\hat{\mathbf{y}} = f_o(W^{(2)} f_h(W^{(1)}\mathbf{x} + b^{(1)}) + b^{(2)})$ where $f_o(x) = 1/(1+\exp(-x))$ and $f_h(x) = \max(0, x)$ are activation functions for the output layer and the hidden layer, respectively. At the output layer, we compute the cross entropy error $CE(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{i=1}^{n} y_i \log(\hat{y}_i) + (1 - y_l)\log(1 - \hat{y}_i)$ where $\hat{y}_i$ is the predicted score for label λ_i. We run Stochastic Gradient Descent (SGD) to train NNs with 1,000 hidden units for 50,000 epochs, which corresponds to 300,000 parameter updates, and use mini-batches of size 100 for computing gradients.

5 Experimentation

In order to estimate the performance on the public and private test set, we performed 10 fold cross validation on the available labeled training data.

Evaluation Measures The competition submissions were evaluated by computing the area under the ROC curve of the label rankings and then averaging over the instances. This measure can be defined as

$$AUC(\mathbf{y}, \hat{\mathbf{y}}) = \frac{1}{|P||N|} \sum_{\lambda_i \in P} \sum_{\lambda_i \in N} [[\hat{y}_i > \hat{y}_j]] + \frac{1}{2}[[\hat{y}_i = \hat{y}_j]] \tag{3}$$

where $[[x]]$ denotes the indicator function and $\hat{y}_i$ the predicted score for λ_i, e.g. the inverted ranking position. It is obvious that $1 - AUC$ corresponds to the popular *ranking loss* used for evaluating multilabel classification [5]. There are several discrepancies in computing this measure, e.g. sometimes the second term is skipped and tied pairs are arbitrarily counted as wrong or correct, or sometimes test instances with an empty labelset are skipped. We compute the score for each instance, i.e. we additionally set $AUC(\emptyset, \hat{\mathbf{y}}) = 1$, but note that for the cross validation results it is easy to obtain the other less optimistic version with $AUC' = (687 \cdot AUC - 100)/587 = 1.17 \cdot AUC - 0.17$. However, this does not explain the discrepancies between the estimated AUC values and the values on the test set, since our best 0.94 would be only reduced to 0.93.

Results Table 1 shows our estimated results and the AUC values on the public and private test set. The first observation is that our preprocessing approach substantially improved the ranking quality over using the provided raw MFCC features. The achieved improvement is greater than the possible improvement by any other tried approach or combination of approaches. This demonstrates the applicability and effectiveness for acoustic data of our neural network based unsupervised feature generation process. Before heading to the comparison between the used approaches, we also note that there is an important discrepancy between the CV estimations on the training set and the test set results which cannot be explained by overfitting or differences in computing AUC (cf. Sec. 5).

We see that the pairwise LibSVM approach (SVM), the random decision trees (RDT) and the neural network (NN) with a single hidden layer obtain similar results on the test sets, with a small advantage for the NN approach. This submission obtained the 5th rank on the public test set and the 8th rank on the private test set.[4] With the arrival of the *big* dataset the last day of the competition, used for training the RDTs, and some struggling in merging teams and results, so that only the predictions of the SVMs with $\gamma = 0.5$ and the RDTs could be merged, we managed to reach the 4th and 6th

[4]http://www.kaggle.com/c/multilabel-bird-species-classification-nips2013/leaderboard/public & ../private.

Table 1: Results for the different multilabel approaches and settings in terms of AUC, estimated on the training data via 10 fold cross validation (with standard deviation) or a train/test split of 600/86 and computed on the public and private test set. Training and prediction times are given in seconds. The features column indicates which feature set was used. The second block shows post-competition results.

Approach	features	CV	public	private	training	predicting
SVM $C = 2000\ \gamma = 10^{-3}$	raw	0.8994	–	–	681.8	49.76
SVM $C = 10^6\ \gamma = 0.5$	small	0.93883 ± 0.0180	0.89202	0.88967	60.18	19.39
SVM $C = 10^6\ \gamma = 1$	small	0.93915 ± 0.0179	0.89130	0.88996	60.96	19.44
RDT 50000 trees	big	0.93718 ± 0.0189	0.89129	0.88195	13444.44	290.83
NN	big	0.92595 ± 0.0200	0.89650	0.89374	5585.15 (GPU)	0.01 (GPU)
SVM & RDT	–	–	**0.90104**	**0.89525**	–	–
SVM $C = 5 \cdot 10^4\ \gamma = 0.1$	big	0.94022 ± 0.0181	0.88699	0.88710	119.71	45.98
SVM $C = 2 \cdot 10^5\ \gamma = 0.1$	big	0.93976 ± 0.0178	0.88752	0.88696	109.26	45.96
SVM & RDT & NN	–	–	**0.90331**	**0.89824**	–	–
RDT & NN	–	–	0.89903	0.89279	–	–
SVM & NN	–	–	0.89807	0.89556	–	–

positions on the public and private leaderboard, respectively. The ranking merging effect had a small impact on absolute numbers, but a considerable effect on the test set ranking due to the high competitiveness in the contest.

It seems clear that merging rankings exploits the diversity of the underlying classifiers by reinforcing predictions if the individual classifiers agree and by (tendentially) correcting rankings if for some instances some of the rankers fail. For binary decision ensembles it can be shown that the accuracy approximates 1 with increasing number of voters, though assuming a certain diversity (in the sense of probabilistic independence) [9, Sec. 4.2.1]. We could confirm this when joining predictions of classifiers of the same family, which did not lead to any improvement. But as the post-competition results in the second half of Table 1 show, combining different approaches almost always improved the AUC. Indeed, if we had submitted a joined prediction of all three approaches, we would have been ranked 4th on both test sets. This is just below the three competitors using advanced and sophisticated acoustic signal processing techniques relying on expert knowledge and on own processing of the raw acoustic data, as reflected by the relatively big gap between them (with AUC greater than 0.91) and the rest of the competitors.

However, please note that our best approaches with almost 0.93 AUC had roughly 10 wrongly paired labels per instance (cf. Eq. 3). It holds that this number e equals $\sum_{i=1}^{|P|} r_i - |P|(|P|+1)/2$ with r_i being the ranks of the positive labels in P, thus for examples with $|P| = 1$ the label is on average ranked at the 11th position, for two labels e.g. on positions 5 and 6. On the other hand, additional evaluations show that for approx. 64.6% of the instances a relevant label was predicted on the first position (one-error loss), and that we could obtain an F1-score of 75% using perfect thresholding. Remind however, that our CV results are overestimations.

6 Conclusions

We have presented a general and unsupervised approach for processing acoustic data, particularly for short recordings of birds', insects' and amphibian sounds. It is based on recent findings and state-of-the-art approaches from the field of neural networks and deep learning. The generated features, which basically are activation signals by using learned feature detectors, achieved an important improvement over using the unprocessed MFCCs in terms of AUC.

The three applied multilabel approaches, which we applied on the data, were highly suited for the particular task and carefully optimized so that we were able to obtain top results in the competition. By combining the individual approaches we could exploit the diversity among them and obtain the 4th rank on the public test set and 6th position in the final ranking. Unfortunately, we did not manage to combine all three classifiers on time, since this would have allowed us to obtain 0.90 AUC and hence the overall 4th rank, right after the three solutions based on expert knowledge and therefore unreachable with our means.

We see some space for improvement in the pairwise learning approach, which currently ignores examples in the label overlaps, and the neural network approach, which still has a lot of unexplored degrees of freedoms for optimizing. However, the narrow range of AUC results in the top 10 (excluding the top 3) indicates that we already got nearly the most out of the provided acoustic representation and multilabel learning. Next steps hence include to find new representations directly from the raw data, and we believe from our work with the birds sounds dataset that supervised and unsupervised techniques from neural networks and deep learning can make important contributions to this.

References

[1] Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Manual, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[2] Adam Coates and Andrew Ng. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning*, pages 921–928, 2011.

[3] John C. Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.

[4] O. Dufour, T. Artières, H. Glotin, and P. Giraudet. Clusterized mel filter cepstral coefficients and support vector machines for bird song identification. In *The 1st International Workshop on Machine Learning for Bioacoustics (ICML 2013)*, pages 89–93, Atlanta, USA, june 2013. Glotin H. et al.

[5] Şeyda Ertekin and Cynthia Rudin. On equivalence relationships between classification and ranking algorithms. *Journal of Machine Learning Research*, 12:2905–2929, November 2011. ISSN 1532-4435.

[6] Johannes Fürnkranz. Round robin classification. *Journal of Machine Learning Research*, 2:721–747, 2002.

[7] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[8] Chih-Wei Hsu and Chih-Jen Lin. A comparison of methods for multi-class support vector machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.

[9] Ludmila I. Kuncheva. *Combining Pattern Classifiers : Methods and Algorithms*. Wiley-Interscience, 2004. ISBN 0471210781.

[10] Eneldo Loza Mencía and Johannes Fürnkranz. Pairwise learning of multilabel classifications with perceptrons. In *Proceedings of the 2008 IEEE International Joint Conference on Neural Networks (IJCNN-08)*, pages 2900–2907, Hong Kong, 2008. IEEE. ISBN 978-1-4244-1821-3. doi: 10.1109/IJCNN.2008.4634206.

[11] Eneldo Loza Mencía, Sang-Hyeun Park, and Johannes Fürnkranz. Efficient voting prediction for pairwise multilabel classification. *Neurocomputing*, 73(7-9):1164 – 1176, March 2010. ISSN 0925-2312.

[12] Grigorios Tsoumakas. Mulan: A java library for multi-label learning, dataset repository. Website, January 2012. URL http://mulan.sourceforge.net/datasets.html. last accessed at 2013-12-01.

[13] Grigorios Tsoumakas, Eleftherios Spyromitros Xioufis, Jozef Vilcek, and Ioannis P. Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

[14] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103, 2008.

[15] Xiatian Zhang, Quan Yuan, Shiwan Zhao, Wei Fan, Wentao Zheng, and Zhong Wang. Multi-label classification without the multi-label cost. In *Proceedings of the Tenth SIAM International Conference on Data Mining*, April 2010.