Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 18 (2014) 133 – 139

Procedia Technology

International workshop on Innovations in Information and Communication Science and Technology, IICST 2014, 3-5 September 2014, Warsaw, Poland

Unsupervised Feature Pre-training of the Scattering Wavelet Transform for Musical Genre Recognition

Mariusz Kleć[a], Danijel Koržinek[a]

[a]*Polish-Japanese Institute of Information Technology, Multimedia Department, Warsaw, Poland.*

Abstract

This paper examines the utilization of Sparse Autoencoders (SAE) in the process of music genre recognition. We used Scattering Wavelet Transform (SWT) as an initial signal representation. The SWT uses a sequence of Wavelet Transforms to compute the modulation spectrum coefficients of multiple orders which was already shown to be promising for this task. The Autoencoders can be used for pre-training a deep neural network, treated as an features detector, or used for dimensionality reduction. In this paper, SAEs were used for pre-training deep neural network on the data obtained from jamendo.com website offering music on creative commons licence. The pre-training phase is performed in unsupervised manner. Next, the network is fine-tuned in supervised way with respect to the genre classes. We used GTZAN database for fine-tuning the network. The results are compared with those obtained with training neural network in a standard way (with random weights initialization).

Peer-review under responsibility of the Scientific Committee of IICST 2014

Keywords: musical genre recognition; deep learning; scattering wavelet transform; autoencoders; neural networks

1. Introduction

Music Information Retrieval (MIR) is a term that is often used to denote a variety of approaches and techniques used to solve numerous problems related to musical data. Whilst the name originated from its simple data mining roots, the field has rapidly grown in both quality and scope throughout the years. Some of the problems that the MIR community attempts to solve include classification and organization of music, recommendation systems and everything up to and including complex analysis of large musical databases by musical experts. Many of these problems have very tangible commercial premise, but most are related to the simple desire to understand basically how music functions by utilizing large databases and the power of computer processing.

Some of the first and foremost approaches to MIR relied solely on text analysis derived from data mining and Natural Language Processing (NLP). The source of this information was the meta-data usually attached to the music directly (as a part of the file) and indirectly (as a part of the larger databases linked to the file, e.g. the web).

This however, proved insufficient in many cases: either because of lack of properly annotated musical pieces or because of the concise and incomplete nature of this annotation. That is when the signal analysis started being used to

E-mail address: mklec@pjwstk.edu.pl, danijel@pjwstk.edu.pl

Peer-review under responsibility of the Scientific Committee of IICST 2014

doi:10.1016/j.protcy.2014.11.025

either fill in the gaps in the meta-data or create whole new levels of annotation unavailable before. There are numerous issues with automatic signal analysis systems: they lack precision, they require accurate ground-truth which is not always easily available and, most importantly, they are difficult to construct. Much of this was solved by projects that created fairly simple toolkits for signal processing, e.g. MIRtoolbox[1], jMIR2[2], Essentia3[3] and many others. These tools are then combined with various Machine Learning (ML) algorithms to create systems capable of solving these MIR tasks.

This paper deals with a common problem of determining the musical genre of a piece of audio based on its acoustic content alone. This problem is very well studied and contains many issues. Even if we are able to define the genre taxonomy, it may prove difficult to establish the actual ground-truth for the training and evaluation database. Both of these issues may vary significantly from expert to expert. Nevertheless, the simplicity of the problem definition makes it a very attractive benchmark even for people with no formal musical training.

As with any ML problem, one of the key issues is the data required for the training and evaluation of the system. While some data is freely available on-line, most quality databases are expensive. This is not an uncommon problem in ML, very similar to e.g. speech. For genre recognition, a very commonly used dataset is the GTZAN database [1]. Even though it has some shortcoming [2], it is widely used as a benchmark in many publications [3,4]. This database consists of 1000 musical files (30 s. lengths) organized in 10 genres (100 examples per each genre). For pre-training of the unsupervised features, a larger database, downloaded from the jamendo.com website, is used. Jamendo is a music sharing platform publishing music on a creative commons licence. A publicly available API allowed the authors to download over 80000 musical tracks. Nearly 10000 tracks were selected according to the genres taxonomy represented by GTZAN database (see section 3).

The first publication on recognizing genres in the GTZAN database was published in 2002 and utilized Gaussian Mixture Models, reporting the accuracy of 61% [1]. Using Deep Neural Networks, the authors in [3] achieved the accuracy of 83% using the standard MFCC features on a 50/25/25 split for training/validation/evaluation sets accordingly. In [5] the authors utilized a special Wavelet-derived feature technique known as the Scattering Wavelet Transform obtaining an impressive 89.3% accuracy (10-fold cross-validation) using a simple SVM classifier. Finally, in [6] the authors combined the SWT features with the power of sparse-representation based classifier to achieve the score of 91.3% accuracy.

This paper is authors' first step in achieving the results of [6] by utilizing Sparse Autoencoders in the pre-training phase of a Multi-Layer Perceptron Neural Network. Furthermore, a much larger database is being used in the pre-training phase to boost the somewhat small data set.

2. Background

This section includes background information of various components used in the experiments described in this paper.

2.1. Unsupervised Feature Learning

Training an Artificial Neural Network (ANN) with multiple layers (i.e. more than 2 or 3 hidden layers) using backpropagation produces sub-optimal results in most practical situations. This is caused by the weakness of the gradient descent optimization method where gradients that are computed by backpropagation rapidly diminish in magnitude as the depth of the network increases. As a result, the final layers don't get meaningful training data [7]. Moreover, even 'shallow' topologies often get stuck in local minima due to heuristic nature of the algorithm. This problem was well known and studied for decades.

A breakthrough happened in 2006 when G. E. Hinton introduced a fast learning algorithm for training, what he named, Deep Belief Networks [8]. This method uses a greedy layer-wise training to train one layer at a time in an unsupervised manner. This step is called pre-training and its aim is to prepare the weights of the model in such a way

[1] https://www.jyu.fi/hum/laitokset/musiikki/en/research/coe/materials/mirtoolbox
[2] http://jmir.sourceforge.net
[3] http://essentia.upf.edu/

that they better represent local feature states. Following that the final fine-tuning of the weights using labeled data creates a model which performs far better than one that is trained on randomly initialized weights alone.

This unsupervised pre-training approach started a new research direction called "deep learning". Deep learning takes advantage of unlabeled data to learn a good representation of the features space [9] - each layer representing another abstraction of the features pre-trained from a layer before. Layer-wise, bottom-up pre-training, one layer at a time, is possible by incorporating Restrictive Boltzman Machines (RBM) or Autoencoders (AE) [10]. Stacking RBMs or AEs (as features detectors) form a "deep structure" which can be fine-tuned using gradient-based optimization methods with respect to labeled data (i.e. supervised training).

2.2. Sparse Autoencoders

An Autoencoder (AE) is an ANN with an odd number of hidden layers, where the number of units in the output layer is set to be equal to the number of units in the input. In other words, AEs try to reconstruct the input at the output passing data through hidden layers. If the number of hidden units is lower than the number of the input/output units, or when special constrains are applied to the network (e.g. sparsity), the hidden layers form a bottleneck of the network. This bottleneck, during training, forces the network to learn a compressed representation of the input. G. E. Hinton used the AE as a method for dimensionality reduction which performs better than PCA [11].

One of the constraints that can be applied to AE training is trying to reconstruct the input from its corrupted version. This is the basic idea behind Denoising Autoencoders [12]. Another type of AEs, used in this paper, are the Sparse Autoencoders (SA). The idea behind them is to enforce activations of hidden units to be close to the zero for most of the time during training. This can be achieved by applying the measure of Kullback-Liebler Divergence (KL) to the cost function:

$$KL = \rho \log \frac{\rho}{\hat{\rho}} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}} \tag{1}$$

$$J_{\text{sparse}}(W, b) = J(W, b) + \beta * KL(\rho \,\|\, \hat{\rho}) \tag{2}$$

KL measures the difference between the two distributions: $\hat{\rho}$ which represents the average activations of hidden units over the training set and ρ representing the target distribution. $J_{\text{sparse}}(W, b)$ denotes the sparse cost function with respect to weights W and biases b . Because we want to keep hidden units inactive most of the time, the target distribution should be set close to zero. In our experiments, described below, the target distribution ρ was always set to 0.1. In other words, we wanted to enforce $\hat{\rho} = \rho$. In order to penalize an average activation of hidden units deviating too much from its target value of ρ, a special penalty term β was introduced to control the weight of the sparsity term.

2.3. Autoencoder Implementation

The neural network with mini batch stochastic gradient descent (SGD) was developed in Matlab from scratch without using any toolboxes. The core of the code was written according to the guidelines presented in CS294A Lecture notes [13]. Additionally, the part of the code responsible for gradient calculation is compatible with minFun[4] function that uses L-BFGS [14] optimization algorithm. This algorithm uses a limited amount of computer memory and in this paper was used for training the Autoencoders to improve the training speed. The code, besides having implemented square-error cost function, was extended to operate on cross entropy error function which is described in [15]. The regularization term of $\|W\|^2$ was added to the cost error function which tends to decrease the magnitude of the weights and helps prevent overfitting. Weight decay parameter, denoted later as λ, is used to control the relative importance of the regularization.

2.4. Scattering Wavelet Transform

Most of the research behind MIR relies on Mel-Frequency Cepstral Coefficients (MFCCs), which are a Fourier-based feature set designed specifically for analyzing speech and music. MFCCs are calculated as the Fourier transform

[4] www.di.ens.fr/ mschmidt/Software/minFunc.html

of the logarithm of the Fourier transform of the signal that was partitioned using standard windowing techniques (like in the STFT). The resulting features can be used to estimate a smoothed spectral envelope that is robust to small intra-class changes, but loses information [16].

Unlike the Fourier transform, which decomposes the signal into sinusoidal waves of infinite length, the Wavelet transform encodes the exact location of the individual components. The Fourier transform encodes the same information as the phase component, but this is usually discarded in the standard MFCC feature set. This means that Fourier-based methods are very good at modeling harmonic signals, but are very weak at modeling sudden changes or short term instabilities of the signal - something that Wavelets seem to deal with very well. That is why Wavelets are usually the prime candidate for analyzing noisy signals like EKG or EEG data.

Until recently, however, Wavelets got very little attention from the MIR community, because they failed to outperform the well-known and fine-tuned MFCC feature sets used for decades. In [16] Mallat introduced the Scattering Wavelet Transform (SWT) which works by computing a series of Wavelet decompositions iteratively (the output of one decomposition is decomposed again) producing a transformation which was both transformation invariant (like the MFCC), but also didn't lose any information, which is proved by producing an inverse transform (something which cannot be done using MFCCs without loss) . In [5] the SWT is used in the problem of phoneme classification and musical genre recognition. The paper also points out a similarity between the multilayer structure of the SWT and a Deep Belief Network. This would hint at a certain level of redundancy of using DBNs with SWT features, but the paper by Chen [6] demonstrates that certain improvements can still be achieved using the self-organization properties of the unsupervised pre-training phase in such classifiers.

3. Data Preparation

Two databases were used in the experiments. First is the well-known GTZAN[5] dataset consisting of 1000 musical files, each 30 seconds in length. They are categorized into 10 genres with 100 musical pieces per category (rock, blues, classical, country, disco, hip-hop, jazz, metal, pop, reggae). The second data collection was obtained from the jamendo.com website which offers music ready to download for free due to the Creative Commons license[6]. A publicly available API[7] allowed to download over 80000 musical tracks together with meta-data in an XML format. The meta-data contains, among other features, a genre association of each file. There are three attributes containing this information: "album genre", "track genre" and "tags". The "album genre" and "track genre" contain ID3 genre names and the "tags" can contain genres and other information (without restrictions) as annotated by users.

The goal was to create a database 10 times bigger than GTZAN and organized in the same way. From the 80000 files, only those that belonged to one of the 10 musical genres were taken into consideration. To avoid ambiguities all the files were passed through a couple of filters. Initially, files which had the same values in all attributes were immediately accepted. This assumption gave the highest probability that a particular file belongs to the given genre. For the genres that thusly resulted in less than 1000 musical files (this occurred with blues, country and reggae which are more specific than pop or rock) the filter was made less restrictive. First, only "track genre" and "album genre" had to be equal to choose a song (ignoring the tags) and if there were still too few songs, only "track genre" was considered and the rest of the attributes were ignored. This generated a list of 9966 musical files organized in 10 musical genres with nearly 1000 track per genre.

Out of each file, a 30 second fragment starting at 30 seconds from the beginning of the file (to skip the potential problems occurring in the beginnings of some tracks) was extracted and down-sampled to 22050Hz to match the GTZAN format.

The features were extracted from the files using the ScatNet toolbox[8]. The SWT transform was computed to the depth of 2, as this was shown as the optimal setting in [5]. The first layer contained 8 wavelets per octave of the Gabor kind and the second had 2 wavelets per octave of the Morlet type. The window length was set to 740ms. After the

[5] http://marsyas.info/download/data_sets/
[6] https://creativecommons.org/licenses/
[7] http://developer.jamendo.com/v3.0
[8] http://www.di.ens.fr/data/software/scatnet/

transformation we obtained 81052 training examples from GTZAN and 802925 training examples from JAMENDO database - each with 747 features. The resulting databases can be acquired by contacting the authors.

4. Experiments

Our initial experiment were based around a simple logistic regression classifier followed by a Multilayer Perceptron (MLP) with different topologies. Next, a SA was pre-trained on the Jamendo data and its weights used in one of the MLP networks to verify if the network will perform better.

The GTZAN dataset was split into three subsets: 50% of randomly chosen samples was used for training, 25% for validation and 25% for final evaluation. The neural network was trained to predict a label for each frame. Maximum voting was used to predict a label for the whole track. Final results are calculated in the form of the error rates of wrongly classified tracks in the test set. The data from the validation set did not take part in training, but its cost value was monitored during training for early stopping. The training was terminated when the cost value on the validation set has not been decreasing by more than 1e-4.

4.1. Logistic Regression and MLP

The first neural network was simple logistic regression with 747 units in the input and 10 at the output. Next, the number of hidden layers was gradually increased by up to the 5 hidden layers. First hidden layer contained 747 units (the same as input) but the following had 400 units. All the hidden layers had a log-sigmoid transfer function except the first which had the hyperbolic tangent transfer function. We used the fallowing settings for training neural network cases: λ:1e-4, lr: 3e-3, batch-size: 300. The results are presented in Table 1.

Model type	Topology	Error rate
Logistic Regression	747-10	19.2%
MLP	747-747-10	16%
MLP	747-747-400-10	16%
MLP	747-747-400-400-10	16%
MLP	747-747-400-400-400-10	15.6%
MLP	747-747-400-400-400-400-10	26.8%

Table 1. Training the neural networks with different topologies without pre-training

4.2. Sparse Autoencoder

The Sparse Autoencoder with a topology of 747-400-747 was trained on the JAMENDO database using the L-BFGS optimizer in 300 epochs. The target value of ρ equaled 0.1. In order to estimate the best parameters of β (penalty term for sparsity constrain) and α (strength of regularization), the activations of the hidden units (being stimulated by the data) were treated as new features of the data and passed to the logistic regression classifier. Best accuracy on that test using GTZAN database determined the best parameters for training of the SA.

For visualization purposes, separate test was performed based on training AE only on the first 85 features of the data. The values of the weights from this test form feature detectors and are presented in Figure 1. In production version of the experiments, the AEs were trained using all 747 dimensions of data. These weights were then used between the second and the third layer of the MLP with all the other weights initialized randomly. The results are presented in Table 2.

5. Conclusions and Discussion

The initial experiments show how adding multiple layers does not provide much benefit in a standard MLP with a backpropagation optimization algorithm. One hidden layer gives significant improvement over simple logistic regression, but the following layers give very little, and after a while the result becomes even worse due to the huge

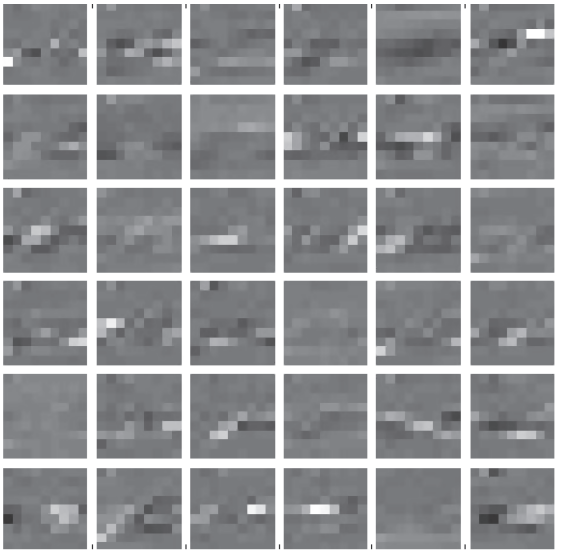

Fig. 1. Weights of hidden units of the trained Sparse Autoencoder on the JAMENDO database with its 85 first dimensions. Each square denotes one hidden unit. The grey colors represents the weights connected to the particular hidden unit. Each hidden neuron can be treated as feature detector by taking activation of its weights being stimulated by the other data - in this case by GTZAN data.

Model type	Topology	Error rate
MLP	747-747-400-10	16%
MLP	747-747-400-400-400-10	15.6%
MLP + SA	747-747-400-10	15.2%

Table 2. Results achieved by utilizing SAE and without it.

search space. Using a simple Sparse Autoencoder, however, improves the result even in the simplest case and even outperforms the best MLP by a little margin. There are a few issues to solve however. The SA seems to adapt better to only the first 85 features of the SWT transform. These correspond to the spectral component of the transform. The higher order features seem to be much more sparse and more experiments need to be performed to fully utilize the potential of this technique. The SA was utilized to pre-train the second set of weights. Pre-training the first set of weights (between the input and the first hidden layer) did not improve the classification rate. The authors suspect that the normalization of the feature space plays a role in the adaptability of the SA. Finally, the SA is used in pre-training of a single layer of the MLP only. To construct a fully functional DBN, the same method should be used for all the layer of a MLP iteratively. The authors hope to achieve this in the near future.

6. Acknowledgments

We would like to thank Krzysztof Marasek, Thomas Kemp and Christian Scheible for their support. This work was funded by a grant agreement no. ST/MN/MUL/2013 at the Polish-Japanese Institute of Information Technology.

References

[1] Tzanetakis, G., Cook, P.. Musical genre classification of audio signals. Speech and Audio Processing, IEEE transactions on 2002;10(5):293–302.
[2] Sturm, B.L.. The gtzan dataset: Its contents, its faults, their effects on evaluation, and its future use. CoRR 2013;abs/1306.1461.
[3] Sigtia, S., Dixon, S.. Improved music feature learning with deep neural networks 2014;.
[4] Panagakis, I., Benetos, E., Kotropoulos, C.. Music genre classification: A multilinear approach. In: ISMIR. 2008, p. 583–588.

[5] Andén, J., Mallat, S.. Deep scattering spectrum 2013;.
[6] Chen, X., Ramadge, P.J.. Music genre classification using multiscale scattering and sparse representations. In: Information Sciences and Systems (CISS), 2013 47th Annual Conference on. IEEE; 2013, p. 1–6.
[7] Glorot, X., Bengio, Y.. Understanding the difficulty of training deep feedforward neural networks. In: International Conference on Artificial Intelligence and Statistics. 2010, p. 249–256.
[8] Hinton, G., Osindero, S., Teh, Y.W.. A fast learning algorithm for deep belief nets. Neural computation 2006;18(7):1527–1554.
[9] Bengio, Y.. Learning deep architectures for ai. Foundations and trends® in Machine Learning 2009;2(1):1–127.
[10] Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.. Greedy layer-wise training of deep networks. Advances in neural information processing systems 2007;19:153.
[11] Hinton, G.E., Salakhutdinov, R.R.. Reducing the dimensionality of data with neural networks. Science 2006;313(5786):504–507.
[12] Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research 2010;11:3371–3408.
[13] Ng, A.. Sparse autoencoder. CS294A Lecture notes 2011;:72.
[14] Skajaa, A.. Limited memory bfgs for nonsmooth optimization. Master's thesis, Courant Institute of Mathematical Science, New York University 2010;.
[15] Bishop, C.M., et al. Neural networks for pattern recognition 1995;.
[16] Mallat, S.. Group invariant scattering. Communications on Pure and Applied Mathematics 2012;65(10):1331–1398.