Instrument Recognition by Audio Feature Extraction and Transformation
Advanced Learning Techniques 2021

Selma Bouchta
ENSAE Paris
selma.bouchta@ensae.fr

Emma Sarfati
ENSAE Paris
emma.sarfati@ensae.fr

Abstract

Instrument recognition, among many other audio recognition tasks, enjoys sturdy state-of-the-art accuracy performances. The objective is *a priori* simple; from numerical features based on the instrument audio file, we build a predictor to detect the right instrument in an audio. The challenge in audio information retrieval is however huge in general. Moreover, "classical" machine learning models have not proved to be super-efficient for the classification purpose. In this short report, we propose an implementation of (Lardeur et al., 2009) on the IRMAS dataset (Bosch et al., 2014)

1 Problem Framing

We face a multi-class classification problem. Precisely, we start with an audio file from which we aim at extracting a significant number of features, called $(x_1, \dots, x_d)$. Audio feature extraction is a specific procedure which calls on several signal processing knowledge. In this paper, we will use the descriptors in the original paper (Lardeur et al., 2009), as well as a few features described in (Peeters, 2004) and (Peeters). Once the features are successfully extracted, the goal is to classify the right instrument in the audio, that is predicting a target y for $y \in \{1, \dots K\}$ with K being the number of possible instruments. Letting $\mathcal{X} \subset \mathbb{R}^p$ the space of possible inputs and $\mathcal{Y} \subset \{1, \dots, K\}$[1] the space of possible outputs, we want to construct $f : \mathcal{X} \to \mathcal{Y}$ such that for $\mathbf{x} = (x_1, \dots, x_d)$, $f() \sim y$.

2 Litterature Review

The state-of-the-art is rich and is suprisingly made of well-known machine learning algorithm, but with tiny nuances for the training procedures. The first attempts toward automatic musical instrument recognition mostly focused on studying basic methodologies for computational modelling (Cemgil and Gurgen, 1998),(Kaminsky and Materka, 1995). Since years 2010, the maximum accuracy reached depends on the dataset used and the number of extracted features. For instance, Yu et al. in 2009 proposed a time-frequency audio features which achieved more than 90% accuracy (Yu and Slotine, 2009) with around 700 features. Such a number of features is however hard to retrieve and to incorpporate during training for a CPU. On the IRMAS dataset, current state-of-the-art performance is a 79% accuracy with well-tuned Support Vector Machines algorithm (Racharla et al., 2020) for the simple multi-class problem where the target contains only one instrument. More elaborated deep-learning based algorithms, with testing data containing polyphonic music (hence the target contains multiple instruments) have also been used and reached a great performance, however less impressive than for the monophonic testing. This approach relies on deep convolutional neural networks (Han et al., 2017). In the paper that is studied in this short report, the approach used lies in a modified training protocol of Support Vector Machines model (Lardeur et al., 2009)

3 Experiments Protocol

Note that **the authors did not provide any code or implementation details of their algorithms**. In our implementation, some details might not be similar to the one originally used by the authors.

3.1 Dataset

We worked with the IRMAS dataset[2], which is a famous dataset used for instrument recognition

[1] we suppose the target is label-encoded, that is class 1 is for violon, class 2 is for saxophone and so on.

[2] The dataset can be downloaded at https://zenodo.org/record/1290750#.YJBIHi0itbV

task. The training data containt around 6000 Waveform audio files, with each file being a sample of an instrument. The instruments considered are: cello, clarinet, flute, acoustic guitar, electric guitar, organ, piano, saxophone, trumpet, violin, and human singing voice. This dataset is derived from the one compiled by Ferdinand Fuhrmann in his PhD thesis (Fuhrmann, 2012). For the test set, the original IRMAS dataset provides audio samples in which we can find more than one instrument. To simplify matters and to follow the same outline as in most of the papers we read, we decided to work exclusively with the IRMAS training set, that is, we will only evaluate our algorithm on monophonic samples. In this configuration, we handle a multi-class classification problem (rather than a multi-label/multi-output one).

3.2 Feature extraction

Our goal was to extract as much relevant features as possible from the dataset. Many features come from the spectrum of the audios which is obtained by :

$$X(k) = \sum_{n=0}^{N-1} x(n) \exp\left(j2\pi \frac{k}{N} n\right) \quad \forall k \in [0, N]$$

Where $x(n)$ is the signal at the sampling rate and k the discrete frequency on N points. We managed to gather 50 features including the following ones:

- ***The spectral centroid:*** a measure that indicates where the center of mass of the spectrum is located. It is calculated as the weighted mean of the frequencies present in the signal, determined using a Fourier transform, with their magnitudes as the weights.

  $$cgs = \frac{\sum_{k=1}^{N} f(k) S(k)}{\sum_{k=1}^{N} S(k)}$$

  where $S(k)$ is the spectral magnitude at frequency bin k and $f(k)$ is the frequency at bin k.

- ***The order-p spectral bandwith:*** computed through

  $$\mathrm{B}_p = \left(\sum_k S(k) \left(f(k) - f_c\right)^p\right)^{\frac{1}{p}}$$

  where $S(k)$ is the spectral magnitude at frequency bin k, $f(k)$ is the frequency at bin k, and f_c is the spectral centroid.

- ***The spectral spread:*** given by

  $$std = \sqrt{\frac{\sum_{k=1}^{N} \left(f(k) - cgs\right)^2 S(k)}{\sum_{k=1}^{N} S(k)}}$$

- ***The spectral contrast:*** it considers the spectral peak, the spectral valley, and their difference in each frequency subband.

- ***The spectral frequency roll-off:*** it is the frequency below which a specified percentage of the total spectral energy lies. (we considered 85%)

- ***Energy levels:*** The energy of a signal is the total magnitude of the signal. We wanted to convert an amplitude spectrogram into dB-scaled spectrogram to get energy levels.

- ***The tempo:*** we estimated the tempo in Beats Per Minute (BPM).

- ***The statistical moments:*** we computed the mean, standard deviation, minimum, maximum, median and quartiles of the frequencies of each signal, as well as other statistical moments.

- ***The peaks:*** The number of peaks in each audio

- ***The pitch:*** The analysis of autocorrelation is a mathematical tool for finding repeating patterns. For musical signals, a repeated pattern can correspond to a pitch period. We can therefore use the autocorrelation function to estimate the pitch in a musical signal.

There were many other features extracted. A dataframe was created with those features, the audios and the label. We then proceeded to the model.

3.3 Model

3.3.1 Support Vector Machines

Support Vector Machines were first introduced by Vapnik in 1992 (Boser et al., 1992). The main idea of this learning algorithm lies in the representation of the training data, which allows us to solve a convex optimization problem. We consider our dataset of the form $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)_{i \in [\![1,n]\!]}\}$, with $y_i \in \{-1, +1\}$ - a classification task. We consider moreover linear functions in feature space of the form $f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x} + b$. For our classification task, we which that $y_i f(\mathbf{x}_i) > 0$. The minimum of this quantity, $\min_i y_i f(\mathbf{x}_i)$ is called the margin,

i.e the gap between data points and the classifier boundary. SVM algorithm simply searches for $\mathbf{w}$ and b that will maximize this margin. Mathematically, this is formulated as (with a normalization with $\|\mathbf{w}\|$)

$$\arg\max_{\mathbf{w},b} \min_{i} \frac{y_i \left(\mathbf{w}^\top \mathbf{x}_i + b\right)}{\|\mathbf{w}\|}$$

other mathematical reformulation leads to a convex opitmization problem which is solved using classical tools such as dual reformulation. The feature vectors satisfying the problem are called the support vectors, denoted $\mathbf{s}_i$. For the support vectors, it stands that $y_i \left(\mathbf{w}^\top \mathbf{s}_i + b\right) = 1$.

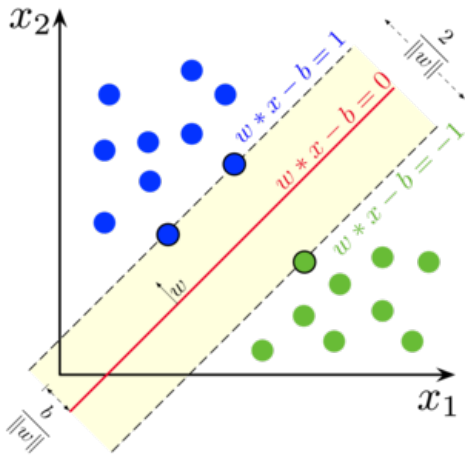

Figure 1: SVM illustration. Source: Wikipédia.

A regularization term is often added to prevent overfitting. If the data are not linearly separable, a kernel function can be used to map the d-dimensional input feature space into a higher dimension space where the two classes become linearly separable. Denoting α_i the Langrange multipliers and n_s the number of support vectors, it can be proved that the optimal classifier f is given by $f(\mathbf{x}) = \sum_{i=1}^{n_s} \alpha_i y_i k\left(\mathbf{s}_i, \mathbf{x}\right) + b$. The SVM algorithm is the state-of-the-art model for instrument recognition and is also the most commonly used (Essid et al., 2004). As a baseline, we trained a SVM classifier on our training data using scikit-learn (Pedregosa et al., 2011). We used the One versus One approach as we deal with a multi-class classification task. We used the Radial Basis Function as a kernel, $k(x, x') = \exp\left(-\frac{\|x-x'\|^2}{2p\gamma^2}\right)$. We tuned our hyper-parameters, namely the regularization parameter C and the RBF parameter γ. We set $C = 1000$ and $\gamma = 0.05$. Results are presented in next big section.

3.3.2 Paper contribution: virtual audio SVMs

The main contribution provided by the authors is the nuance that is brought during the training of the SVM. We first need to introduce three sounds effects that will be applied on our feature data (more precisely they will be applied to the support vectors). The first one is the reverberation, which corresponds to a persistence of sound after the sound is produced. The second one is equalization, the process of adjusting the balance between frequency components within the audio signal. The third one is compression, which aims at reducing the dynamic of the signal. Based on the procedure proposed by (Decoste and Cristianini, 2003), the authors design a new SVM training algorithm.

1– do SVM training on the original data $\mathcal{X}$;
2– mark the audio frames corresponding to the support vectors found, apply the sound effects to them and extract the selected features from the transformed audio frames, thus creating the set of virtual training examples $\hat{\mathcal{S}}$;
3– re-train the SVMs using the original feature vectors $\mathcal{X}$ plus all the virtual ones in $\hat{\mathcal{S}}$ created using all the effects.

The audio transformation on the support vectors were carried using the library `pysndfx`[3]. We then followed exactly what is proposed in the paper, that is we trained a classical SVM on the features obtained in part 3.2, we applied 3 sound effects on the original audio files, re-extracted the features on these new audio, merged the new dataframe with the old one and applied a final SVM on this new data. This algorithm is referred as VASVM in the paper. In the next section, we discuss the results obtained for both methods.

4 Results

We first evaluated our SVM and VASVM using the accuracy of the models. Accuracy for a multiclass classifier is calculated as the average accuracy per class and it is an appropriate metric when the classes are balanced, which is our case. The accuracy for a binary classification problem is given by

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

[3] see the GitHub repo of the library: https://github.com/carlthome/python-audio-effects

This accuracy was computed for each of the OneVsOne Classifier and we took the mean. For the VASVM approach, we computed the accuracy for two test sets: first, on the same test set as the classical SVM model in order to compare them properly. Second, as the VASVM dataset is larger due to the modified audio files that were added, we tested the model on a larger test set, to be consistent with the new size. These accuracies are referred to as VASVM1 and VASVM2 respectively.

Model	SVM	VASVM1	VASVM2
Accuracy	0.6780	0.8998	0.8645

Table 1: Test accuracy results.

A second interesting evaluation measure is the confusion matrix, which helps us visualize the repartition of the false and true classifications.

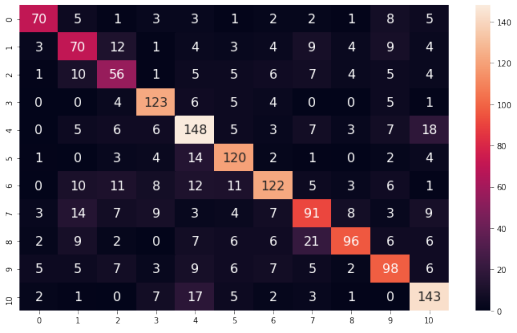

Figure 2: Confusion matrix for the SVM.

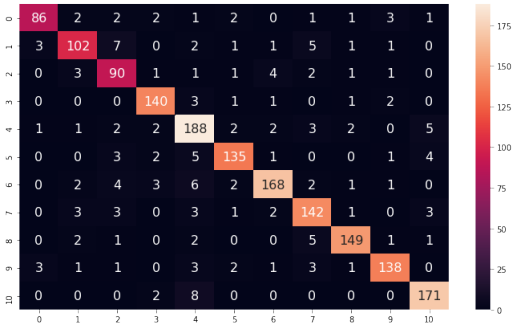

Figure 3: Confusion matrix for VASVM1.

We can directly compare the two confusion matrix above as they refer to the same test base. We can see that the number of correctly-predicted classes is globally larger for the VASVM model. Indeed, the diagonal of the matrix presents higher numbers.

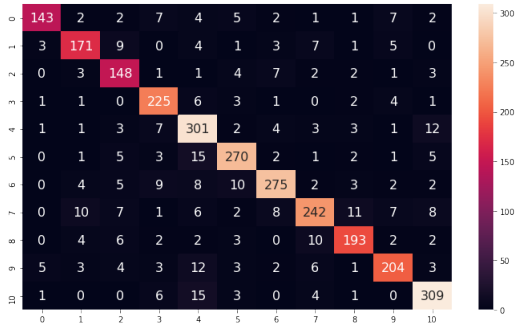

Figure 4: Confusion matrix for VASVM2.

We can see that the diagonal values are way better than in the SVM and VASVM1 confusion matrices. This shows greater predictions. However, we cannot directly compare them, given that the test set is larger for the VASVM2.

5 Discussion

It seems that we succeeded in reproducing the model that was proposed in the original paper (Lardeur et al., 2009). We however studied this approach for monophonic music only, and this is what we saw in the major share of the litterature. For the paper, it also seems that the monophonic testing was prefered, but this was not precised by the authors. As a comparison, we could have tested our model on polyphonic music for testing, which would have forced us to one-hot encode the target vector as the training would have been a multi-label one. For this kind of approach, it would have been interesting to test a deep-learning based approach with a Softmax output to predict probabilities of played instruments. It remains that for monophonic music, the model proposed by the authors achieves great results and provides improvement with respect to a simple SVM approach. However, a few remarks can be made:

– The model results in a larger training base. This is a form of data augmentation, as in deep learning. In this sense, it seemed inevitable that the results would improve.

– The model is very sensitive to the kernel function that was used. The data seems to be highly non-separable, as the linear kernel did not run properly on our machine. Moreover, the regularization parameter plays an important role; a low value of it leads to a low accuracy.

– We managed to extract less features than in the paper. They extracted 401 features against 50 in

our work. A high number of features can give better predictions but can also lead to noise. We face a trade-off between better approximation and adding noise.

6 Conclusion

To conclude, we managed to extract many features from audios and to reproduce the model presented in the paper. The model clearly gives better results in terms of accuracy than a classic SVM model. However, as we discussed, we only focused on monophonic musics and it would have been interesting to try our model on polyphonic music by changing a few set ups. To extend our approach, we could have tried many different machine or deep learning models. But the particularity of the support vector machines is that the support vectors give us the opportunity to understand the meaningful datapoints for the model, in the sense that they are at the boundary between two classes. Audio transformation was applied on these support vectors. As a comparison, we could have tried to follow this process with an unsupervised approach, such as a k-means, and apply the audio transformation to the cluster centroids datapoints, as an equivalent of the support vectors.

References

Geoffroy Peeters. NSY122 extraction d'information audio. http://recherche.ircam.fr/anasyn/peeters/pub/cours/Peeters_2011_CNAM_1.pdf.

Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA. Association for Computing Machinery.

I. Kaminsky and A. Materka. 1995. Automatic source identification of monophonic musical instrument sounds. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 1, pages 189–194 vol.1.

Ali Cemgil and F. Gurgen. 1998. Classification of musical instrument sounds using neural networks.

Dennis Decoste and Nello Cristianini. 2003. Training invariant support vector machines. *Machine Learning*, 46.

Geoffroy Peeters. 2004. A large set of audio features for sound description (similarity and classification) in the cuidado project.

Slim Essid, Gaël Richard, and Bertrand David. 2004. Musical instrument recognition on solo performances. In *2004 12th European Signal Processing Conference*, pages 1289–1292.

Guoshen Yu and Jean-Jacques Slotine. 2009. Audio classification from time-frequency texture. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1677–1680.

M. Lardeur, S. Essid, G. Richard, M. Haller, and T. Sikora. 2009. Incorporating prior knowledge on the digital media creation process into audio classifiers. In *2009 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 1653–1656.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830.

Ferdinand Fuhrmann. 2012. *Automatic musical instrument recognition from polyphonic music audio signals*. Ph.D. thesis, Universitat Pompeu Fabra.

Juan J. Bosch, Ferdinand Fuhrmann, and Perfecto Herrera. 2014. IRMAS: a dataset for instrument recognition in musical audio signals.

Yoonchang Han, Jaehun Kim, and Kyogu Lee. 2017. Deep convolutional neural networks for predominant instrument recognition in polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 25(1):208–221.

Karthikeya Racharla, Vineet Kumar, Chaudhari Bhushan Jayant, Ankit Khairkar, and Paturu Harish. 2020. Predominant musical instrument classification based on spectral features. *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN)*.