Musical Instrument Recognition

Karsten Knese #03620344 KARSTEN.KNESE@TUM.DE
Technische Universitaet Muenchen

Karol Hausman #03633065 HAUSMANKAROL@GMAIL.COM
Technische Universitaet Muenchen

Abstract

The goal of this project is to separate three different instruments, namely guitar, cello and piano. The minimal goal is to train a Machine Learning algorithm with a moderate number of training samples in order to build a mathematical decision boundary to distinguish those three instruments. Certainly, the solution is supposed to be extensible in a way that without a lot of effort further instruments can be recognized. In the beginning the algorithm is trained by a tone with a frequency of 440Hz; in a musical environment called a'. The point of the separating approach, which is presented in this document, is to extract the ratio between the recorded frequency(440Hz) and all peaks of further resonating frequencies.
The training is done by a classical SVM algorithm. A long-distance goal would be to develop a more robust system which can be relevant to various frequencies and to train the system with more instruments.

1. Motivation

The motivation of the project was to make a Machine Learning application that is not based on any artificially created or assumed values,but on the real measurements. As both authors play guitar they agreed on this topic in order to examine if it is possible to separate various instruments with the Machine Learning algorithms.

2. Prerequisites on music and sound theory

To be able to follow the upcoming solution from a technical side, it is crucial to briefly explain signal processing of recorded sounds as well as the basics properties of the instruments we played. In terms of audio it is vital to introduce notions that will be used in the further sections:

a tone is exactly one single continuous frequency (which usually does not sound good on its own)[1]. It consists exclusively of one single periodic (sine) frequency.

a clang is the overlapping of certain frequencies which build up a unique sound for humans. In contrast to noise, it is important to remark that a clang might not be consistent of harmonic frequencies, but always periodic.

noise comprises multiple frequencies which are neither harmonic nor periodic.

Figure 1 gives a visual description about the terms mentioned above. For further proceeding the greatest emphasis will be put on clangs, as it is crucial for the given problem. The instruments used in this paper are so-called harmonic instruments. Harmonic means that there are no randomly chosen frequencies overlapping, but only those which are multiples of the base frequency (440Hz in this case). Therefore for further notation the frequency on which the instruments are recorded will be called **base-tone**. Each multiples of the base frequency will be named **overtone**. Specifically, the overtones will be extracted around 880Hz, 1320Hz, 1760Hz. The answer for the question how to distinguish different instruments lies in the above mentioned overtones. Each instrument produces an unique magnitude of overtones. As the human ear is able to

[1] an audio example can be found in [(Wikimedia)]

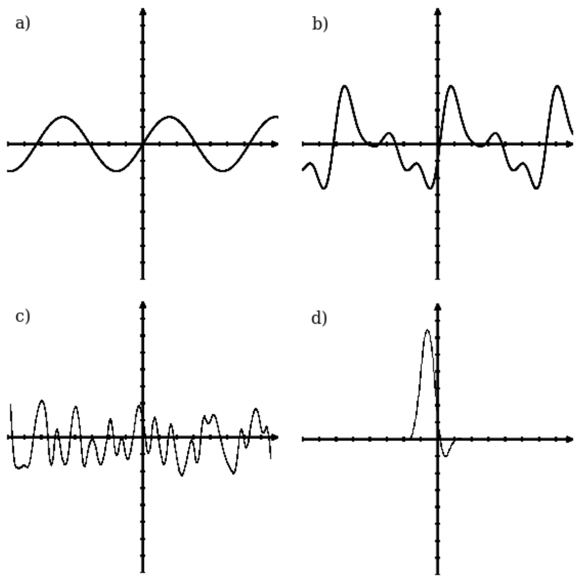

Figure 1. a) tone b) clang c) noise d) smack (negligible)

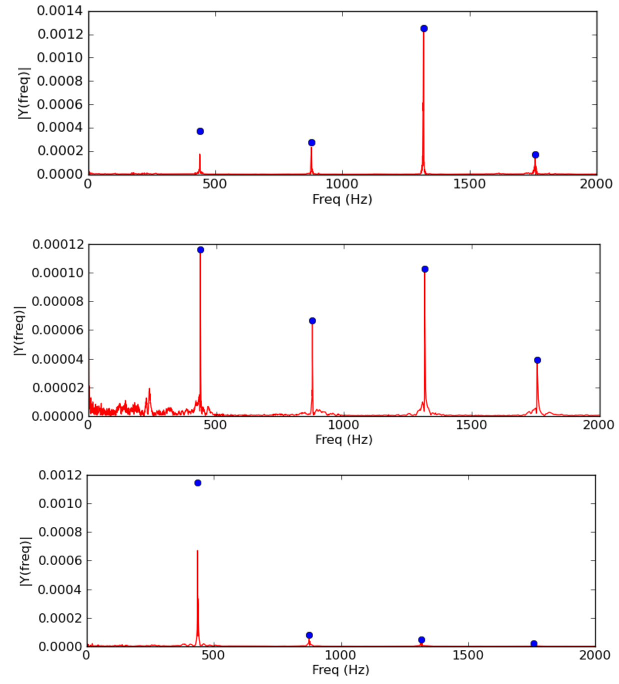

Figure 2. frequency peaks from top to bottom: cello, guitar, piano

recognize this magnitude, people can distinguish the instruments. Figure 2 shows the frequency spectrum of cello, guitar and piano. It is clearly visible that the overtones, marked with the blue points, are unique in terms of magnitude for each instrument. The magnitude and the ratio between the overtone and the base tone is used as the feature since it happens to be crucial to recognize the instruments.

3. Feature Modelling

3.1. Audio recording

Obviously, the first step is to gather (enough) samples in order to start the research. In total 303 samples were recorded which cover three different instruments playing an a' on 440Hz. The recorded sounds vary in terms of loudness, duration and threads[2]. What can be seen in the further sections, the time variation can lead to massive loss of accuracy since it takes certain time for each instrument until all overtones are fully resonating. The result of recording and sampling is presented on figure 3.

3.2. Feature extraction

In the last paragraph the difference of the various instruments as the ratio between the overtones frequencies and the base tones was introduced. It can be seen on figure 3 that the amplitude in decibels over time was recorded. It is important to conduct the transformation from the periodic wave spectrum into the frequency spectrum. It can be done with a technique called Fast Fourier Transformation (FFT). Basically, with the help of FFT sample continuous periodic time data can be transfered into discrete frequency data[3].The used implementation was inspired by [(FFT, 2011)]. Once the FFT was applied to the data figure 2 was obtained. The next step is to define a model for further computation. Regarding the fact that the notes of 440Hz were recorded there was a need to search for peaks around $(n * 440Hz) \pm 220, n \in [1; 4]$. After observing a branch of samples four major peaks can be identified. Hence the ratio between each overtone and the base tone were taken. Eventually, a feature model with three values for each sample was obtained.

$$f = \begin{pmatrix} ratio1 & ratio2 & ratio3 \end{pmatrix}$$

After looking at the data it could be noticed that guitar and cello were relatively close in the feature space whereas piano shows much wider-spread values. To

[2]the variation of threads is only applicable on guitar and cello.

[3]Further detailed information please refer to [(Rife & Vincent, 1969)]

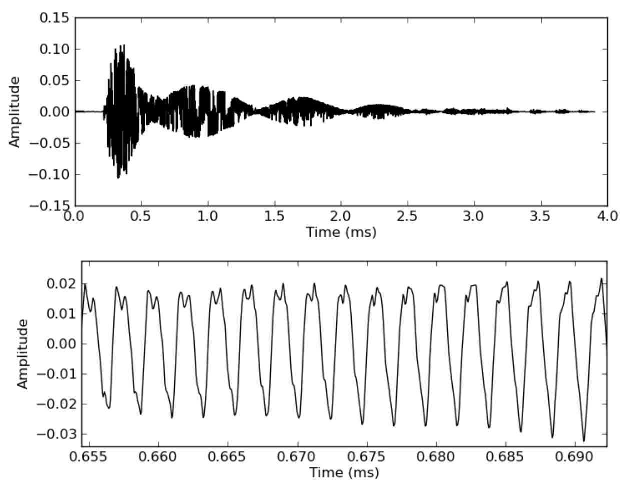

Figure 3. First picture: piano record. X=time, Y=amplitude measured in dB
Second picture: enlarged snippet of piano record. Clearly visible overlapping periodic frequencies.

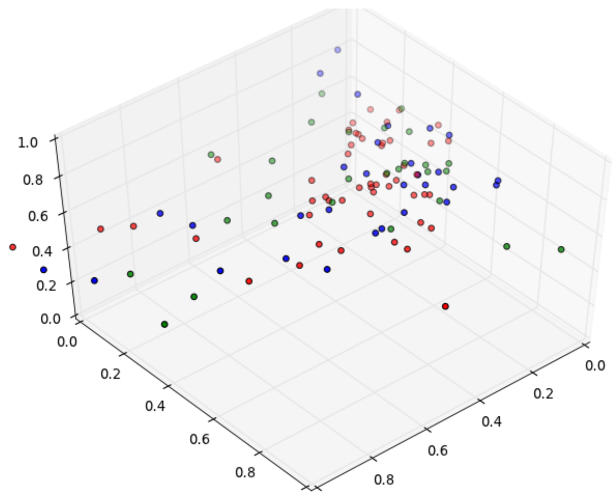

Figure 4. The distribution of three clangs. Red=a', Green=c", Blue=d". There are no clusters for the sound played.

have all the features in a common range between approx. $[-1;1]$the feature scaling algorithm was applied. The result of these operations was the following feature vector.

$$f_i = \left(\frac{(f_i - \mu_f)}{\sigma_f}\right)$$

where μ_f and σ_f are the mean and the standard deviation of f. Using these features has a great advantage. As the model was restricted to the ratio of the frequencies, it is applicable for other notes as well (not only for a' sound anymore). The model was also tested with the records of a c' on 520Hz and d' on 585Hz and there was not seen the lack of feature information. This means no additional computation steps to transpose 520Hz to 440Hz are needed. Figure 4 shows the distribution of those features for three clangs. It can be noticed that all clangs are nearly equally distributed. After applying all of the methods mentioned above the feature space on figure 5 was obtained.

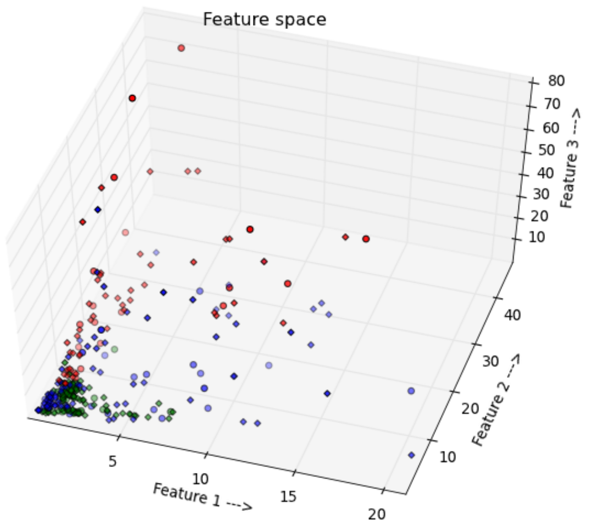

Figure 5. Uniform distribution three clangs. Red=a', Green=c", Blue=d". We can see that there are no clusters of clangs.

4. Classification

After preprocessing the data, it was decided to use the Support Vector Machine algorithm for classification. In order to obtain fast and reliable results the common and popular library libsvm [(lib, 2011a)] was applied. It comes with a several command line tools as well as a handy python interface. The complete dataset is presented in the following table. In order to apply cross validation algorithm the data was divided into a training set and a test set with a commonly used ratio of 70% training and 30% test.

N	cello	guitar	piano	training	test
303	126	110	67	87/79/46	39/31/21

The dataset has to be modified to a certain format before getting processed by libsvm. The format consists of the following structure:

<label> <i>:<x_i> <i+1>:<x_i+1> [...]

where *label* $\in$ $\mathbb{Z}$, i $\in$ $\mathbb{N}[increasing]$, x_i $\in$ value of feature i. After training the SVM with multiple kernel parameters there was gained a decision boundary illustrated in the picture 6. In order to find

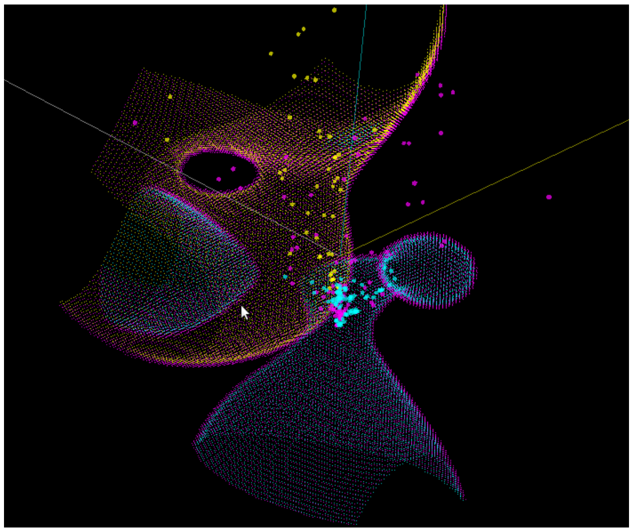

Figure 6. decision boundaries with rbf kernel

an optimal parameters for the kernel **Grid Search** was applied on the training set. The parameters were validated via a n-fold Cross-Validation process. In the SVM approach there are basically two parameters which can be optimised. The first parameter is the bias parameter C which penalizes the misclassification so that following SVM equation becomes optimal.

$$\frac{1}{2}||\mathbf{w}||^2 + C\sum_{i=1}^{m} \xi_i$$

The second parameter comes from the duality of solving SVM equations with the Laplace constraints, where kernels happen to be more efficient. The assumption was that the Gaussian Kernel (*rbf Kernel*) was the best for this dataset, so the parameter σ seem to be most effective in the following equation.

$$\exp(-\frac{||x-y||^2}{2\sigma^2})$$

By applying Grid Search the results were $C = 65536$ and $\sigma = 8$ and the Cross-Validation at optimum of about 85%. However, to gain a real valuable number predicted were the datasets, which are unknown for the model. In the prediction phase 39 cello, 31 guitar and 21 piano samples of each instrument were separated into a test set for validating the model. Figure 7 shows the iterations of Grid Search which found the local optimum shown on the figure. With this approach the reasonable guess about setting the parameters can be obtained. In total the validation was executed 91(= 39 + 31 + 21) times. In other words, the label is known and the model which was just trained is supposed to return the right label. If the expected label differs from the predicted one, the test failed.
The overall results for the SVM approach are listed in table 4.

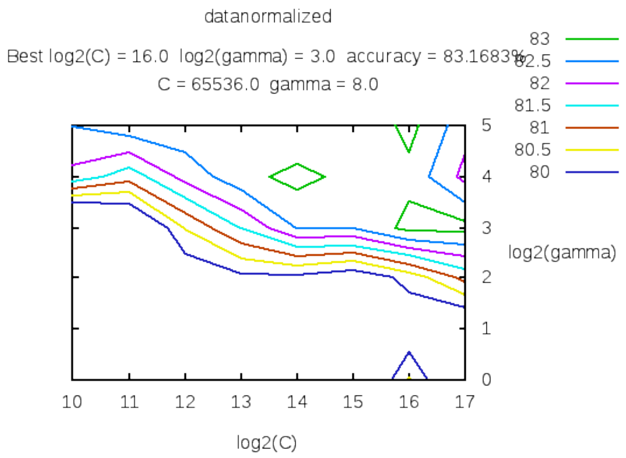

Figure 7. Plot of Grid Search iterations

	shorten guitar	normal guitar
Cello success	97.43%	97.43%
Cello fail	2.57%	2.57%
Guitar success	67.74%	100.0%
Guitar fail	32.26%	0%
Piano success	90.47%	95.23%
Piano fail	9.53%	4.77%
Total success	85.72%	97.33%
Total fail	14.28%	2.67%

Table 1. Results of SVM approach. First column shows result with corrupt guitar data. Samples were too short for evolving all harmonics and there they lack of unique information. Seconds column shows new records especially focused on normal long guitar samples. The result is tremendous.

One interesting result about this table is the conclusion to the used dataset. It can be observed that both piano and cello are very well recorded with 97% accuracy, whereas the guitar performs rather poorly with only 67% accuracy. Taking into consideration the difference between the two results it can be seen that one poorly performing instrument has noticeable impact on other instruments even though they haven't been changed.

To get a short insight of the development an abstract code construct in algorithm 1 is presented. These are the crucial steps to perform and neglect all unimportant details.

Algorithm 1 SVM Approach

```
Input: training data $X$, test data $Y$
for $x_i$ do
    fft = getFFT($x_i$)
    featureTrainingVector = extractOverToneRatio(fft)
end for
for $y_i$ do
    fft = getFFT($y_i$)
    featureTestVector = extractOverToneRatio(fft)
end for
featureTrainingVector = normalize(featureTrainingVector)
featureTestVector = normalize(featureTestVector)
SVM.addFeatureVector(featureTrainingVector)
model = SVM.train()
predictedLabel = SVM.predict()
for $y_i$ do
    actualLabel = $y_i$.label
    predictedLabel = model.predict($y_i$)
    if actualLabel $\neq$ predictedLabel then
        print test failed for $y_i$
    else
        print test succeeded for $y_i$
    end if
end for
```

5. Conclusion

During the work with the sound files and the Machine Learning tools, we got familiar with the importance of valuable recordings and precise preprocessing. As we mentioned at the beginning, the duration of the samples is of the highest importance while attempting this problem. It takes a certain time for each instrument to evolve all overtones.Unfortunately, our first recordings of guitar contained many samples that were too short. This made it almost impossible to identify the instrument correctly based on the overtone ratio alone. Table 4 shows the contrast in classification results of short guitar samples versus normal guitar samples.
In order to make our system more robust against the shorter guitar recordings we had to introduce more features to help distinguish the samples more efficiently, independent of the overtones.
In order to find a possible solution we studied the spectrogram shown in figure 8. A spectrogram shows the FFT over time. In contrast to the standard FFT which is in $\mathbb{R}^2$, the spectogram is in $\mathbb{R}^3$. Unfortunately, the spectrogram brought no contribution to our problem as we saw that every harmonic overtone started immediately with the base tone. One possible addition in

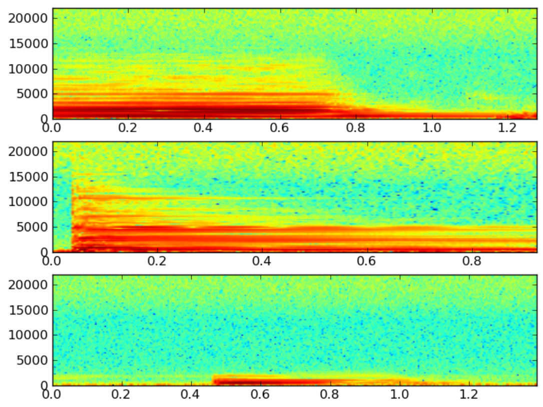

Figure 8. The spectogram shows the simultaneous start of all harmonics for each instrument.

the feature space is the change of energy over time. This method is often called ADSR (Attack, Decay, Sustain, Release) in the field of synthesis. Basically, it is a method to track an amplitude of sound over time. Figure 9 visualizes the different stages, where A is increasing, D is decreasing, S remains stable and R releases the energy.

6. Tools

We implemented the whole source in python. The complete set of projects can be found and downloaded in [(Hausman & Knese)]. Thereby we used a couple of major python libraries for matrix calculation, audio processing and spectrum computation. Namely, we used those following python libraries:

scipy / numpy It might be the most common library for scientific calculation. It finds massive use in linear algebra, statistics and signal processing applications [(sci)].

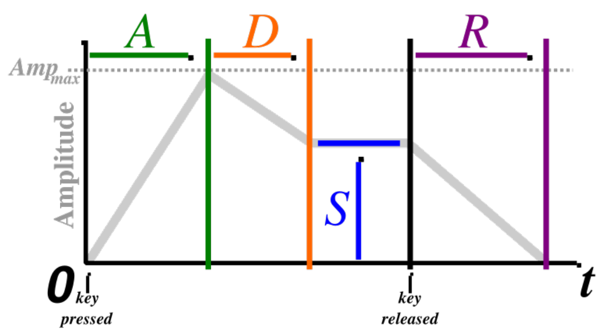

Figure 9. Four stages of ADSR

matplotlib A powerful library to develop all kinds of plots. We developed the 3D plots with the help of matplotlib [(mat)].

scikits.audiolab This library was used to process our records. It is able to deal with all common audio formats like wav, aiff, ogg, flac [(aud)].

libsvm This is the most essential and powerful library we used during this project. It encapsulates the whole SVM proceedings like optimisation and covers all the math behind it. Though it is basically written in C, due to its popularity it got multiple add-ons and interfaces to other languages like python, Matlab [(lib, 2011a)].

It is worth it to briefly mention two additional application we used in the process of solving this problem.

grid.py It is an in-house command line tool of the libsvm package which allows to apply gridsearch on a given dataset and to plot the results. The gridsearch plot in this document is made by this tool. It is really powerful to give an idea of the value of collected data.

svm-toy-3D It is another tool we used to visualize 3D SVM decision boundaries. Unfortunately, it is written in Java and comes with some third-party dependencies. This was the reason why we did not to include this plot into our application [(lib, 2011b)].

7. Outlook and Future Work

The real value of this attempt complements with further projects like *tone-recognition* by Ross Kidson and Fang Yi Zhi. Since their program is able to extract notes from various instruments we can use their output to predict the matching instrument. Thus, one could be able to extract a single track of a song and apply configuration to it. For example one operator could turn off one instrument. It would perfectly allow to create backing tracks for people to play-along with a song replacing the sound of a certain instrument with their own sound.

References

scikits.audiolab 0.11.0. http://pypi.python.org/pypi/scikits.audiolab/.

matplotlib. http://matplotlib.sourceforge.net/.

Scientific tools for python. http://www.scipy.org/SciPy.

Plot frequency spectrum with scipy. http://glowingpython.blogspot.com/2011/08/how-to-plot-frequency-spectrum-with.html, 2011.

A library for support vector machines. http://www.csie.ntu.edu.tw/~cjlin/libsvm/, 2011a.

creating 3d svm decision boundaries. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/svmtoy3d/, 2011b.

Hausman, Karol and Knese, Karsten. Instrument recognition source. http://sourceforge.net/p/instrumentrecog/wiki/Home/.

Rife, D.C. and Vincent, G. A. Use of the discrete fourier transform in the measurement of frequencies and levels of tones. 1969.

Wikimedia. Sine 440hz ogg sample. http://upload.wikimedia.org/wikipedia/commons/2/2f/440.ogg.