ATIAM 2017 - ML Project
Embedding music for automatic composition spaces

Mathieu Prang, Philippe Esling[*]
Institut de Recherche et Coordination Acoustique Musique (IRCAM)
UPMC - CNRS UMR 9912 - 1, Place Igor Stravinsky, F-75004 Paris
`{mathieu.prang, esling}@ircam.fr`

Abstract

This project aims to develop new representations for symbolic music generation and automatic composition. Whereas the previous approaches are based on known mathematical rules, you will try to develop a more empirical model through machine learning. Your goal is to represent musical symbols in a space that carry semantics relationships between them, called *embedding space*. This approach allows to extract new descriptive dimensions that may be relevant for music analysis and generation. To that end, you will use Convolutional Neural Networks in order to capture features of the piano-roll representation of musical pieces. First, you will propose a toy dataset that has to follow some semantical rules in order to evaluate your embedding spaces. To that end, you will need to define what are the contextual rules that could mimic semantic relationships in music. Then, you will extend these approaches by using Recurrent Neural Networks to train your models with a symbolic prediction task.

1 Introduction

In the past decades, the field of *computer music* has precisely targeted the problem of understanding musical concepts. Indeed, it is with this information that we can provide tools to help composers and listeners but also define methods of analysis and composition that improve our musical knowledge. Nowadays, a wide variety of approaches have been developed but on a large scale comparison, we can see that they all largely rely on one crucial point: *the way we represent music*.
In this project, you will use the machine learning framework to represent musical symbols as points (vectors) in a space, whose distances mirror the semantic similarity between notes in the same way that have been already done for natural language Mikolov et al. [2013], with advanced models such as GloVe Pennington et al. [2014]. This kind of learning space could be very valuable for the musical analysis and composition field. First, this could lead to various analysis and knowledge inference tools. Furthermore, it could be an efficient new representation of music for many creative application. Finally, this continuous space could provide melody generation or transformation directly from it.

1.1 Convolutional Neural Networks (CNN)

Convolutional NNs are an extension of classical NN that exploit the properties of *stationarity*. There are four main operations in this kind of network, each processed by a different layer (Figure 1).

Convolution The first layer is the convolution operator. Its primary purpose is to extract features from the input matrix. Indeed, each units k in this layer can be seen as a small filter determined by the weights W_k and bias b_k that we convolve across the width and height

[*]https://esling.github.io/

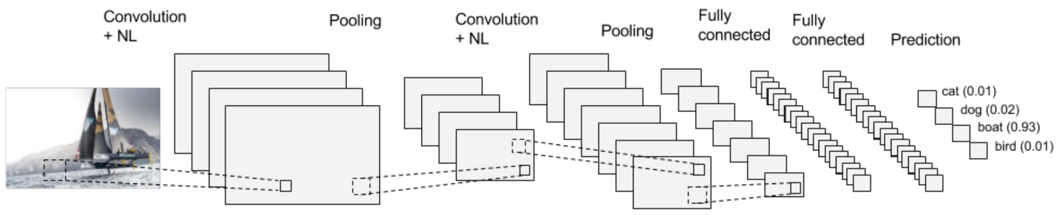

Figure 1: Convolutional neural network with two convolutional layers each followed by a pooling layer and two fully connected layers for classification.

of the input data x. Hence, this layer will produce a 2-dimensional activation map h^k, that gives the activation of that filter across every spatial position

$$h^k_{ij} = (W^k * x)_{ij} + b_k \tag{1}$$

With the discrete convolution for a 2D signal defined as

$$f[m,n] * g[m,n] = \sum_{u=-\infty}^{\infty} \sum_{v=-\infty}^{\infty} f[u,v]g[m-u,n-v] \tag{2}$$

The responses across different regions of space are called the *receptive fields*. During the training process, the network will learn filters that activate when they see some recurring features such as edges or other simple shapes. By stacking convolutional layers, the features in the upper layers can be considered as higher-level abstraction such as composed shapes.

Non-linearity We have to introduce *non-linearities* (NL) in the network in order to model complex relationships. Hence, before stacking every feature maps in order to obtain the output activations we apply a non-linear function like the recently proposed ReLU (Rectified Linear Unit) defined by $output = max(0, input)$ Dahl et al. [2013].

Pooling or subsampling Spatial pooling (also called subsampling or downsampling) allows to reduce the dimensionality of each feature map. The principle behind the pooling operation is to define a spatial neighborhood (such as a 3×3 window) and take the largest elements (*max-pooling*) or the average (*average-pooling*) of all elements in that window. In that way, we progressively reduce the size of the data representation to make it more manageable.

Classification Based on the highest level features in the network, we can use these to classify the input into various categories. One of the simplest way to do that is to add several fully-connected layers. By relying on this architecture, the CNN will take into account the combinations of features similarly to the multi-layer perceptron.

2 Toy dataset

Your first assignment, in order to gain an understanding of the problem, is to create toy datasets that will be used to test different models. Here we will consider *melodies* and *scores* for a *single instrument*, that can be *multiphonic* (ie. able to produce single notes and chords alike). As we work on specific data which are multiphonic melodies and progressions from single instruments, we propose to create a toy dataset that follows the behavior of chord progressions. Thus, we must focus on two main aspects: music theory basics and semantic relationships. This toy dataset will be used to analyze the models in simplified setups in order to understand their behavior.

Exercice 1 - Designing a toy dataset By looking at the results from GloVe at `https://nlp.stanford.edu/projects/glove/`, you can see that the system is able to discover *linear substructures* from natural text processing (Section 2). By looking at the *man-woman* and *comparative-superlative* figures, you can see that metric relationships are found by the model. Here, we will focus on trying to find the same type of relations inside symbolic music.

In the field of symbolic music generation, some typical constructions exist, while other remain under-studied. Therefore, defining a toy dataset is an open question and your proposed solution will be evaluated on the design and justification of your choices. Hence, we highly recommend that you use both your personal knowledge, music theory analysis books and eventually to propose some innovative approaches to the question. An interesting example of toy dataset for image is the *dSprites* dataset `https://github.com/deepmind/dsprites-dataset`

In this case, the following questions must be answered separately (and therefore, produce two sets of scripts and data), for *music theory progressions* (eg. scales and others) and *linear substructures*.

1. Describe some seminal and prototypical examples that should minimally be understood by any system trying to tackle the question of musical inference
2. Describe the sets of linear substructures akin to textual information that could be found in symbolic music
3. Describe situations in which these sequences require multiple scales of time
4. Code different *procedural functions* that could generate large sets of examples following all of your observations in the previous questions
5. Generate your sets of data (midi melodies) in a structured way
6. Explain how these sets can be labeled and analyzed

3 Models and expected work

In order to understand the models and to be able to code these by yourself, we will start by understanding the basic CNN and RNN, and perform their implementation by relying on the *Keras* framework for Python. Tutorials are available at `http://keras.io`

Exercice 2 - Learning Keras and understanding CNN and RNN As you might see in the Keras tutorials, this framework propose some pre-developed layer for both CNNs and RNNs. However, this might not help you to understand their inner functioning. Therefore, we will try to better understand those architectures and to code their simpler version by ourselves. Here, we will rely on the tutorial `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. Regarding the CNN, their development is now so obvious that you can simply perform the basic Keras tutorial on these at `https://keras.io/getting-started/sequential-model-guide/`

1. Based on the description of the tutorial, try to develop your own RNN and own LSTM layer (using Keras layer definition `https://keras.io/layers/writing-your-own-keras-layers/`). You can also use other sources of information.
2. Compare your models to the RNN and LSTM already provided in Keras
3. Train all models on your toy datasets and compare performance
4. Analyze and explain the behavior of the models for different properties
5. Find interesting visualizations to understand this behavior

3.1 Word embeddings for music

The GloVe model proposed in the original paper Pennington et al. [2014] proposes to extend the typical Word2Vec approach by relying on a frequentist approach. Here, the relationships are defined by the frequencies of co-occurences between elements. We will start by coding a typical prediction framework, and then try to extend this model to frequencies

Exercice 3 - Understanding the reference paper Here, we refer to `https://medium.com/@thoszymkowiak/how-to-implement-sentiment-analysis-using-word-embedding-and-convolutional-neural` as a base tutorial and also the embedding layers that are already pre-defined in Keras `https://keras.io/layers/embeddings/`

1. Re-implement the tutorial given and compare to the Embedding layer
2. Extend and apply both models to musical data
3. Train all models on your toy datasets and compare performance

4. Analyze and explain the behavior of the models for different properties
5. Find interesting visualizations to understand this behavior
6. Extend your approach to the GloVe model
7. Extend this approach to compare different prediction types (predicting one element with several, predicting several with one, or performing random removals)

3.2 Improving the model with temporal attention

One of the key aspect in all models lies in the definition of the architecture of the model. However, these can be easily replaced by any model of any complexity, depending on the task at hand. Regarding music, a very interesting framework is that of *attention models* in recurrent networks that put different importance on the data based on an attention vector.

Exercice 4 - Extending to temporal attention To understand the different approaches to attention and sequential temporal processing in recurrent neural networks, you can read `https://distill.pub/2016/augmented-rnns/` for a high-level insight. Then, we will rely specifically on the tutorial given at `https://medium.com/datalogue/attention-in-keras-1892773a4f22`

1. Based on the tutorial, implement the attention recurrent encoder/decoder Keras
2. Replace your previous model with your new model
3. Train all models on the toy datasets and evaluate their results
4. Apply the same set of visualizations as Exercice 3
5. Analyze and explain the behavior of the models

3.3 Creative applications

The whole goal of embedding spaces is to obtain organized sets that would facilitate automatic composition. For instance, one of the most direct application is to put a whole piece of music and obtain its path (set of coordinates). Then, we can re-compose a new piece of music by performing any kind of geometric operation (rotation, translation) on this path.

Exercice 4 - Proposing applications This question is open to any propositions that could produce interesting musical results. We will evaluate this on the design, justification and functionalities offered by the different applications that you propose.

1. Propose different usages of the embedding spaces for musical creativity
2. Implement these toy applications that could help composers or encourage creativity.

References

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

George E Dahl, Tara N Sainath, and Geoffrey E Hinton. Improving deep neural networks for lvcsr using rectified linear units and dropout. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 8609–8613. IEEE, 2013.