Chord2Vec - Learning Chords Embeddings

Coauthor1
Affiliation
Address
email

Coauthor2
Affiliation
Address
email

Coauthor3
Affiliation
Address
email

Coauthor
Affiliation
Address
email

Abstract

In natural language processing, the well-known *Skip-gram* model learns vector representations of words that carry meaningful syntactic and semantic information. In our work, we investigate whether similar high-quality embeddings can be found for symbolic music data. We introduce three NLP inspired models to learn vector representations of chords and we evaluate their performance. We show that an adaptation of the *sequence-to-sequence* model is by far superior to the other proposed model.

1 Introduction

The *word2vec* model by Mikolov et al. [2013] learns vector representations of words that carry syntactic and semantic information which has proven powerful in various natural language processing (NLP) tasks. In this work, we investigate whether similar embeddings can be found for symbolic music data.

In the spirit of the *Skip-gram* model [Mikolov et al., 2013], the heuristic we use to achieve this involves finding chord representations that are useful for predicting the temporally neighboring chords in our corpus of musical pieces. Given such a corpus of ordered musical chords $\mathcal{T} = \{\boldsymbol{d}_1, \boldsymbol{d}_2, \ldots, \boldsymbol{d}_T\}$ and a neighborhood of size $2j$ defined by $C(\boldsymbol{d}_t) = \{\boldsymbol{d}_{t+j}, -m \leq j \leq m, j \neq 0\}$, the objective is to maximize the average log probability with respect to some model parameters θ:

$$\max_{\theta} \frac{1}{|\mathcal{T}|} \sum_{\boldsymbol{d}_t \in \mathcal{T}} \sum_{\boldsymbol{c}' \in C(\boldsymbol{d}_t)} \log p(\boldsymbol{c} = \boldsymbol{c}' | \boldsymbol{d}_t, \theta)\,, \tag{1}$$

where for the remainder of this report, we assume $j = 1$. Whereas text can be represented as a sequence of words, each encoded by a “one-hot" vector, for music we need a “many-hot" vector to represent a chord. This is because more than one note may sound simultaneously.

In the next section 2 we introduce our three NLP inspired models. The first is based on a simple conditional independence assumption, the second relaxes this assumption, while the third adopts a more sophisticated *sequence-to-sequence* architecture. In section 3 we evaluate the models on five datasets, before summarising and suggesting directions for future work in section 5.

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

2 The Proposed Models

2.1 Bilinear model

Our first model is a simplistic adaptation of the *Skip-gram* model introduced by Mikolov et al. [2013] to the many-hot case. Here, we replace the last softmax layer in *Skip-gram* by a sigmoid layer with N outputs, to predict each note individually.

The architecture is a feed forward neural network consisting of input, hidden and output layers. At the input layer, a chord is encoded using a fixed length binary vector $\boldsymbol{c} = \{c_1, c_2, \ldots, c_N\} \in \{0,1\}^N$, where N is the number of notes in the vocabulary. In this chord representation, the entries that are set to 1 correspond to the notes that are *on* in the chord, whereas the notes that are *off* are set to 0.

The weights between the input layer and the hidden layer can be represented by a matrix $M \in \mathbb{R}^{D\times N}$. Similarly, the weights between the hidden layer and the output layer can be represented by a matrix $\tilde{M} \in \mathbb{R}^{N\times D}$.

The D-dimensional vector representation $\boldsymbol{v}_d$ of the associated chord $\boldsymbol{d}$ is simply the normalized sum of the columns of M that correspond to the notes occurring in $\boldsymbol{d} \in \{0,1\}^N$:

$$\boldsymbol{v}_d = M\frac{\boldsymbol{d}}{\|\boldsymbol{d}\|_1}$$

To compute the probabilities in (1) under this model, we make a conditional independence assumption between the notes in a context chord $\boldsymbol{c}$ given a chord $\boldsymbol{d}$, *i.e.*

$$p(\boldsymbol{c} = \boldsymbol{c}'|\boldsymbol{d}) = \prod_{i=1}^{N} p(c_i = c_i'|\boldsymbol{d})\,. \tag{2}$$

Using weights matrices M and $\tilde{M}$, we define a scoring function $h : \mathcal{N} \times \mathcal{C} \mapsto \mathbb{R}$ by

$$h(i, \boldsymbol{d}) = \tilde{M}_{(:,i)}\boldsymbol{v}_d\,, \tag{3}$$

where $\tilde{M}_{(:,i)}$ denotes the i'th row of $\tilde{M}$. We then model the required conditional probabilities as

$$p(c_i = 1|\boldsymbol{d}) = \sigma\big(h(i, \boldsymbol{d})\big), \tag{4}$$

where $\sigma(x) = 1/(1+\exp(-x))$ is the usual sigmoid function. This model is not expected to perform well, mainly because of the independence assumption between the notes in a chord. In fact, it is very likely that there is a strong dependency between the notes appearing in a chord. In the next section, we introduce a model that does not make such an independence assumption.

2.2 Autoregressive model

To overcome the necessity of making the independence assumption between the notes appearing in a chord while retaining tractability, we introduce another model inspired by the The Neural Autoregressive Distribution Estimator (NADE) which models the distribution of high-dimensional vectors of discrete variables [Larochelle and Murray, 2011]. We decompose the context chord probability distribution according to the chain rule, so that

$$p(\boldsymbol{c} = \boldsymbol{c}'|\boldsymbol{d}) = \prod_{i=1}^{N} p(c_i = c_i'|\boldsymbol{d}, c_{<i})\,, \tag{5}$$

where $c_{<i} = \{c_1, \ldots, c_{i-1}\}$. We define a new scoring function

$$h(i, \boldsymbol{d}, c_{<i}) = W_{:,i} \cdot (\boldsymbol{v}_d + \boldsymbol{v}_{c_{<i}})\,, \tag{6}$$

where $M, L \in \mathbb{R}^{D\times N}$ and $W \in \mathbb{R}^{N\times D}$ are weight matrices, $\boldsymbol{v}_{c_{<i}} = \sum_{j<i} L_{:,j}c_j$ and $\boldsymbol{v}_d = M\frac{\boldsymbol{d}}{\|\boldsymbol{d}\|_1}$ is the D-dimensional vector representation of our chord. As before we let

$$p(c_i = 1|\boldsymbol{d}) = \sigma\big(h(i, \boldsymbol{d}, c_{<i})\big). \tag{7}$$

Hence, like NADE we retain tractability while relaxing the conditional independence assumption.

2.3 Sequence-to-sequence model

Finally, we propose to adapt the famous *sequence-to-sequence* model [Sutskever et al., 2014] to the task of learning chord embeddings. In language models, RNNs are useful for predicting future elements of a sequence given prior elements. The *sequence-to-sequence* model builds on top of language models to create translation models that operate on two sequences – the input sequence and the translated output sequence. It looks at each element of the sequence and tries to sequentially predict the next elements of the output sequence. We then suggest to train a *sequence-to-sequence* model to predict the elements in the temporally neighboring chords given an input chords. In this setting, a chord c is represented as an (arbitrary length) ordered sequence of notes, so that $c_i \leq c_{i+1} \in \{1, 2, \ldots, N\}$.

Appropriately, *sequence-to-sequence* models learn a mapping of input sequences of varying lengths to output sequences also of varying lengths, using a neural network architecture known as an RNN Encoder-Decoder. An LSTM encoder is used to map the input sequence to a fixed length vector, and another LSTM decoder is then used to extract the output sequence from this vector. The general goal is to estimate $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$, where $x_1, \ldots, x_T$ and $y_1, \ldots, y_{T'}$ are the input and output sequences, respectively. The objective is

$$\max_{\theta} \frac{1}{|\mathcal{T}|} \sum_{(\boldsymbol{x},\boldsymbol{y})\in\mathcal{T}} \log p(\boldsymbol{y}|\boldsymbol{x}, \theta)\,, \tag{8}$$

where $\boldsymbol{y}$ is a correct output given the input $\boldsymbol{x}$, $\mathcal{T}$ is the training set and θ is the set of the model parameters. The encoder and decoder are jointly trained to maximize the objective according to θ.

The model estimates the conditional probability $p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T)$ by first obtaining the fixed-length vector representation $\boldsymbol{v}$ of the input sequence (given by the last state of the LSTM encoder) and then computing the probability of $y_1, \ldots, y_{T'}$ with the LSTM decoder:

$$p(y_1, \ldots, y_{T'}|x_1, \ldots, x_T, \theta) = \prod_{t=1}^{T'} p(y_t|\boldsymbol{v}, y_1, \ldots, y_{t-1}, \theta) \tag{9}$$

In our setting, we form the $\boldsymbol{y}$ by concatenating elements of the neighbourhood chords. We appropriately insert a special symbol (say, ϵ) which serves as an end of chord marker (similar to an end of sentence marker in NLP). For example, if $\boldsymbol{d} = \boldsymbol{c}_i = (60, 64, 67)$ and the neighbourhood consists of chords $\boldsymbol{c}_{i-1} = (59)$ and $\boldsymbol{c}_{i+1} = (62, 65, 69)$, then we have $\boldsymbol{x} = (60, 62, 67)$ as our input sequence $\boldsymbol{y} = (59, \epsilon, 62, 65, 69, \epsilon)$ as our output sequence. We take the corresponding $\boldsymbol{v}$ as the embedding for $\boldsymbol{d}$.

3 Experiments

We consider five datasets of varying complexity, taken from Boulanger-Lewandowski et al. [2012]:

- **JSB Chorales**: 382 chorales by J.S. Bach with the split of Allan and Williams [2005].
- **Nottingham**: 1200 folk tunes with chords instantiated from the ABC format.
- **MuseData**: electronic library of orchestral and piano classical music from CCARH.
- **Piano-midi.de**: classical piano MIDI archive split according to Poliner and Ellis [2007]
- **Mix**: the union of all the above.

The polyphony varies from 0 to 15 and the average number of simultaneous notes is 3.9. The range of notes spans the whole range of piano from A0 to C8, *i.e.* $N = |\mathcal{N}| = 88$. We also augmented the training data as follows: we transpose each piece by ϕ semi-tones, for $\phi = -6, -5, \ldots, 4, 5$.

3.1 Baselines

In addition to the previously mentioned models, we compare our results with the following models. **Random** is the simplest baseline with uniform $p(c_i = c_i'|d) = 0.5$. **Marginal** is the smoothed empirical distribution of the notes in the training data, $p(c_i = 1|d) = \frac{z_i+\alpha}{|\mathcal{T}|+\alpha}$ where z_i is the number of the occurrences of note i in the training set and $\alpha = 1$ is the smoothing constant.

3.1.1 Implementation Details

The linear *chord2vec* model and the autoregressive *chord2vec* were trained in batches of size 128 with Adam Optimizer using $D = 1024$. Each model is optimized for 200 epochs, but the final model is the one leading to the lowest validation error. Our sequence-to-sequence model architecture is a multi-layer LSTM with 2 layers of 512 hidden units each. To allow sequences of varying length, standard bucketing and padding methods are used. We append to each target chord an end-of-sequence symbol. We implemented all three models using Google's Tensorflow library.

4 Results

The negative log likelihoods on the test data for all models and on all data sets are presented in Table 1. First, we observe that our simple linear model, despite the independence assumptions, is able to actually learn something beyond the marginal distribution of the training set. Secondly, as expected, we observe that the autoregressive model achieves better scores than the simple linear model on all data sets, confirming our hypothesis that the notes in a chord are not independent of each other. Lastly, it appears that the /seqtoseq/ model is by far the strongest model overall.

Table 1: Average negative log likelihood per chord for the test set.

Model	JSB Chorales	Nottingham	MuseData	Piano-midi.de	Mix
Random	61.00	60.99	60.99	60.99	60.99
Marginal	12.23	10.44	23.75	17.36	15.26
Linear c2v	9.77	5.76	15.41	12.68	12.17
Autoreg. c2v	6.18	3.98	14.49	10.18	7.42
Seq2Seq c2v	1.11	0.50	1.52	1.78	1.22

5 Summary and Future work

Our study has revealed that the /seqtoseq/ model is dramatically superior to our simpler methods. However, in the case of text classification problems, it has been shown that simpler architectures can lead to results that are comparable to deep learning models in terms of accuracy, but much faster for training and evaluation [Joulin et al., 2016]. With this in mind, a direction for future work would be to develop another model based on the *Skip-gram* model but without making the assumption of independence between the notes occurring in a chord.

One could also visualize the learned vectors by using dimensionality reduction techniques. An inspection of such low dimension vectors could help understanding whether the embedding vectors capture interesting musically relevant information.

Another interesting direction involves combining our *sequence-to-sequence* based chord embedding model with an additional temporal recurrence, in order to model entire pieces of music.

References

Moray Allan and Christopher KI Williams. Harmonising chorales by probabilistic inference. 2005.

Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. 2012.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *CoRR*, abs/1607.01759, 2016. URL `http://arxiv.org/abs/1607.01759`.

Hugo Larochelle and Iain Murray. The neural autoregressive distribution estimator. *JMLR: W&CP*, 15:29–37, 2011.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

Graham E Poliner and Daniel PW Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Applied Signal Processing*, 2007(1):154–154, 2007.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL `http://arxiv.org/abs/1409.3215`.