Learning Chords Embeddings

Sephora Madjiheurem

July 22, 2016

1 Linear Model

1.1 Skip-gram Model: Recap

The skip-gram allows to efficiently learn high-quality distributed vector representations that capture precise syntactic and semantic word relationships [1]. We give here a short reminder of how the skip-gram model works.

We define a text as a sequence of words drawn from a finite vocabulary of size W. A word can be described as a "one-hot" vector $w_t \in \{0,1\}^W$, where exactly one entry is non-zero and the subscript t represent the position of the word in the text. Given a word w_t in a text, define the context of word w_t by $C(w_t) = \{w_{t+j}, -m \leq j \leq m, j \neq 0\}$, where m is the size of the context. We consider the conditional probability of a context given a word $p(w_{t+j}|w_t)$. The goal is to find word representations that are useful for predicting the surrounding words in a sentence. Formally, given a corpus of words of size T and the context of word w_t given by $C(w_t)$, the objective of the skip-gram model is to maximize the average log probability.

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{w\in C(w_t)} \log p(w|w_t)\,. \tag{1}$$

The parametrization of the skip-gram model uses the architecture depicted in Figure 1 (from Mikolov et al. [1]). In this model, each output is computed using softmax to obtain the posterior distribution of context words:

$$p(w_{t+j}|w_t) = \frac{\exp(v_{w_{t+j}}^T v_{w_t})}{\sum_{w=1}^{W} \exp(v_w^T v_{w_t})} \ , \tag{2}$$

where $-m \leq j \leq m, j \neq 0$, m is the size of the training context, v_w is the vector representation for w, and W is the number of words in the vocabulary.

Detailed derivations and explanations of the parameter learning for this original skip-gram model can be found in [3].

Because the computation cost of objective (9) is proportional to W which can be very large when working with text training data, Mikolov et al. [1] use instead an efficient approximation, known as negative sampling (see [2] for details).

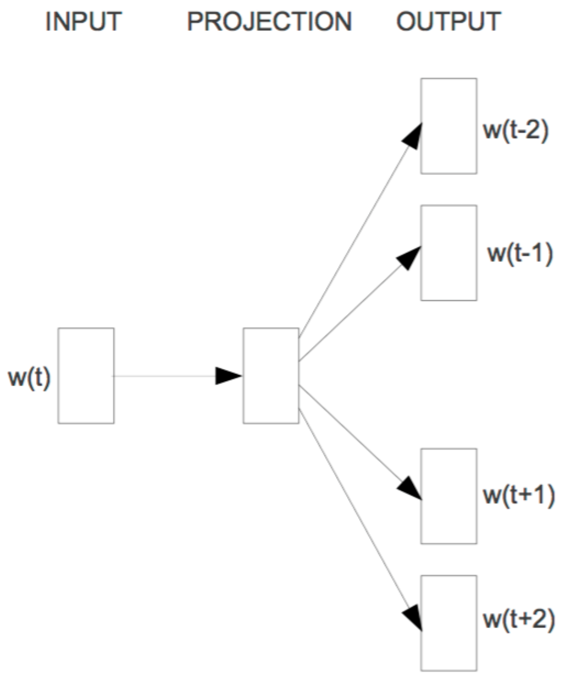

Figure 1: Skip-gram (Mikolov et al. [1])

1.2 Chord2Vec

Similarly to a text, we define a piece of music as a series of chords. A chord is a subset of notes drawn from a finite set of size N and can be represented by a binary vector $\boldsymbol{c} = \{c_1, c_2, \dots, c_N\} \in \{0,1\}^N$.

To adapt the skip-gram model to music data there are a few points that need to be considered:

1. A text can be represented as a sequence of words, where each word can be represented as a "one-hot" vector. In the case of music, we need a "many-hot" vector to represent a chord, as more than one note can be heard simultaneously.

2. The set of notes is smaller than the vocabulary considered when working with text data.

A naive adaptation is to use a separate sigmoid function at the output layer to predict each note in a chord. This makes the conditional independence assumption between the notes in a context chord $\boldsymbol{c}$ given a chord $\boldsymbol{d}$, i.e

$$p(\boldsymbol{c}|\boldsymbol{d}) = \prod_{n=1}^{N} p(c_n|\boldsymbol{d})\,. \tag{3}$$

Using the weight matrix $M \in \mathbb{R}^{D\times N}$, we can compute a score for each note c_i in a chord:

$$h(i,\boldsymbol{d}) = M^T_{(:,i)} \frac{M\boldsymbol{d}}{\mathbb{1}^T\boldsymbol{d}}\,, \tag{4}$$

where $M_{(:,i)}$ denotes the i'th row of M and $\mathbb{1}$ is the vector of all 1's of size N.

$$p(c|d) = \prod_{n=1}^{N} \frac{1}{1+\exp(-v^T_{c_{t+j}} v_{c_t})}\ \ , \tag{5}$$

where $-m \le j \le m, j \neq 0$, m is the size of the training context, N is the number of different notes and v_c is the vector representation for chord c.

The second point, together with the fact that the computation of the objective is no longer dependent of number of possible context chords (since we are not using softmax, the sum over all possible context chords disappears) suggest that there might not be the need to use efficiency optimizations tricks as negative sampling.

2 Sequence autoencoding

2.1 Sequence-to-sequence: Recap

Sequence-to-sequence models allow to learn a mapping of input sequences of varying lengths to output sequences also of varying lengths [4]. It uses a neural network known as RNN Encoder-Decoder. Figure 2 depicts the model architecture. An LSTM encoder is used to map the input sequence

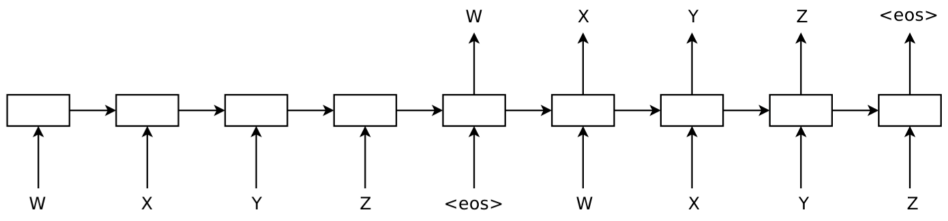

Figure 2: Sequence-to-sequence (Sutskever et al. [4])

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, and T' and T need not to be equal.

The objective is given by:

$$\max_{\theta} \frac{1}{|\mathcal{T}|} \sum_{(X,Y) \in \mathcal{T}} \log p(Y|X, \theta) \,, \tag{6}$$

where Y is a correct output given the input X and $\mathcal{T}$ 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 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:

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1}) \tag{7}$$

2.2 Chord-to-chord

The sequence-to-sequence model can be used to learn embeddings for chords, by training the model to learn the context of a given chord.

In this setting, a chord is represented as a sequence of notes in some fixed ordering: $c \subseteq N$, where N is the ordered set of all possible notes. A chord can have an arbitrary size.

The goal is then to estimate $p(n_1^{(j)}, \ldots, n_T^{(j)} | n_1, \ldots, n_{T'})$, where the n's are in N, $c_t = n_1, \ldots, n_{T'}$ is an input chord and $c_{t+j} = n_1^{(j)}, \ldots, n_T^{(j)}$ is the j^{th} neighbor of c.

If $C(c_t)$ denotes the set of chords that are in the neighborhood of the chord c_t, then the objective in (6) can be written as:

$$\max_\theta \frac{1}{W} \sum_{t=1}^{W} \sum_{c \in C(c_t)} \log p(c | c_t, \theta) \,, \tag{8}$$

Where W is the size of the corpus of chords in the training data.

2.3 Chord-to-chords

An alternative to estimating the probability of single context chord is to estimate the probability of the entire neighborhood $p(C(c_t)|c_t)$, by combining all context chord in one longer output sequence.

The corresponding objective is then given by:

$$\max_\theta \frac{1}{W} \sum_{t=1}^{W} \log p(C(c_t) | c_t, \theta) \,, \tag{9}$$

References

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

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc., 2013.

[3] Xin Rong. word2vec parameter learning explained. *CoRR*, abs/1411.2738, 2014. URL `http://arxiv.org/abs/1411.2738`.

[4] 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`.