WORD2VEC SKIPGRAM MODEL FOR MUSIC SEQUENCE PREDICTION.

Hilda Ibriga

Introduction

This document contains part one of the music sequence prediction project. Here, I use a wor2vec skipgram model to predict the next song in a user playlist sequence.

The word2vec skipgram was the model selected in my first attempt at solving this problem due to its simplicity compared to more state-of-the-art neural network models architectures such as RNN (LSTM). In part two of the project, I implement a Long Short Term Memory (LSTM) model which produce much better results as its architecture is well suited for sequential dataset. However, I was curious to see how a simple model would fare for this problem and therefore started with the word2vec skipgram.

Project Goal description:

The task in this project is to predict/recommend the 30^{th} song that a group of users of an online music streaming app would listen to, given the sequence of previous 29 songs they have listen to. For each user, the prediction result is to be presented as a list of 10 artists guesses order by preference

Data:

The dataset for this project consists of artists played by users on an online streaming service. The data contains 972 users and a repertoire of 3888 unique artists into total. A sequence of 29 artists for each user is provided in the data. The data is presented as a 972 x 29 dataframe where each row represents a user and each column (j) represent the j_th artist listened to by a user.

Model: Work2vec Skipgram Model

I approached the music prediction problem in the way I would approach the problem which motivated the creation of the wor2vec algorithm. However instead of words and sentences as data, I consider my data to be artists and sequences of songs. The goal was therefore to build a neural network model which given a sequence of artists (context) predicts the most likely artist (center) associated to the context (sequence of artists previously played). Below I provide a full description of the skipgram model that was used for training and prediction for this project.

Preparing the Data:

Step 1- Divide and Conquer: As mentioned above, the full data contains 972 lines corresponding to 972 users. I used the method of "divide and conquer" and ran the skipgram model on subsets of the data in parallel on the scholar cluster.

The 972 users were divided into 15 subsets of 60 users and one extra subset of 72 users. A total of 16 subsets were therefore used for building the model. This number of subsets were chosen to allow both training and prediction for each subset to be completed within 35 minutes. The subsets were obtained

using the following schema: the first 60 users data in the dataset were assigned to the first subset, the second set of 60 users to the second subset. This procedure was repeated for all 15 subsets and the last 72 users remaining were assigned to the 16^{th} subset.

A better method would have been to generate the 16 subsets by randomly drawing 60 users however this method would involving keeping track of the original ordering in the data in order to rearrange them into the original order for submission in kaggle.

Since the repertoire of unique artists in the data is large (3888), I believe that training the data on smaller subsets improved the model prediction power. This is because having the full data would mean having a one hot encoding vector of size 3888 which could potentially make both training and prediction less accurate.

Step 2 - Build a playlist repertoire (Vocabulary): After the data was divided into subsets, for each subset, following the wor2vec method, I created a list of unique artists appearing in each subset. I call this list of unique artist, the "artist repertoire" which is similar to the notion of "vocabulary" in the case of the original word2vec model. Table 1: provides the size of the artist repertoire for each of the 16 subsets.

Step 3 - Creating Input/output pairs: The context window used for the skipgram model was set to 5 after trying different window sizes (2 and 3) which lead to accurate predictions. A window of 5 means that for each artist (**target artist**) in a user playlist sequence we will use the 5 artists played before and the five artists played after the target artist as input data for the model. These 10 artists represents the input data with and are each paired with the target artist. The input and output pairs are therefore defined as the pair of target artist and artist within the window of size five of the target artist.

In the following paragraph, I use the data of one user to explain the word2vec sikpgram neural network architecture that I used for this project.

Skipgram Neural Network Architecture

Input layer: The input layer presents the artists within the window of size 5 for each target artist in a user playlist sequence. The artist repertoire is transformed into a vector using a one hot encoding with each position in the vector representing a unique artist in the repertoire. A given artist is represented by a 1 in the artist position and zeros in all other elements of the vector. This one hot encoding of the artists in the window of size 5 of a given target artist represents an input for the neural network:

Since we build 16 models in parallel using 16 subsets of the data, the size of the one hot encoding vector varies by subset depending on the size of the "artist repertoire".

Hidden Layer: The word2vec skipgram model has a very simple neural network architecture as it only has one hidden layer known as the embedding layer. Moreover, there is no activation function between input layer and hidden layer. For this project, I set the number of neurons in the hidden layers first to 32 and then to 48. My original intent was to try up to 120 neurons however training the model was significantly slowed down with more than 48 neurons which led me to only report the results using 48 neurons for all 16 subsets of the data.

The output layer: the output layer is a vector of the same size as the input layer. However, instead of one hot encoding a softmax function is applied as the activation function between the hidden layer and output layer. This yields values between 0 and 1 for each of the elements of the D dimensional vector (. The values in each entry of the vector can be interpreted as the probability that the artists corresponding to that position in the vector is the target artist (that is the artist to be most likely played in a playlist including the input artist)

The Figure1 below provides an illustration of the word2vec model architecture I used to train the weights for each of the 16 data subsets.

Optimization Function: The goal in the skipgram model is to maximize the likelihood of the data representation. The formula of the likelihood function is provided below. Once the output layer is generated using the softmax function, the difference between the vector of probability obtained and the true target artist one hot encoding vector is computed.

Loss function: The difference $(y_i - \hat{y_i})$ represents the error of the model after each pair of input and output is feed to the neural network. The formula of the loss function is provided below.

Model search method: I used gradient ascent to update the weights W_1 and W_2 of the hidden and output layers respectively. In order to obtain the gradient of the weights, I used the method of backpropagation which computes the derivatives likelihood function with respect to the parameters W_1 and W_2 using the chain rule. I provided the equations involved in both forward pass and backpropagation for the neural network.

Training process

Training the neural network can be summarized as follows: I run the algorithm for a total of 250 epochs with a learning rate $\epsilon = 0.001$.

Forward Pass: x $\in \mathbb{R}^{1\times D_k}$, $W_1 \in \mathbb{R}^{D_k\times 48}$, $W_2 \in \mathbb{R}^{48\times D_k}$, $\hat{y}_i \in \mathbb{R}^{1\times D_k}$

Where the index D_k represents the number of unique artists in the kth subset of the data, i the ith input/ ouput pair and j the jth artists in the repertoire.

$$z_1 = xW_1$$

$$h = z_1$$

$$z_2 = hW_2$$

$$\hat{y} = \frac{\exp\,(z_2(j))}{\sum_{J=1}^{D} \exp\,(z_2(j))}$$

The log likelihood function is:

$$L_i = \sum_{j=1}^{D} y_i(j) \log\, \hat{y}_i(j)$$

Backpropagation :

$$\frac{dL_i}{dz_2} = y_i - \hat{y}_i$$

$$\frac{dL_i}{dW_2} = h_i^T(y_i - \hat{y}_i)$$

$$\frac{dL_i}{dh} = W_2(y_i - \hat{y}_i)$$

$$\frac{dL_i}{dW1} = x_i^T W_2(y_i - \hat{y}_i)$$

$$\frac{dL}{dW_1} = \frac{1}{N_i}\sum_{i=1}^{N_i}\frac{dL_i}{dW_1}$$

$$\frac{dL}{dW_2} = \frac{1}{N_i}\sum_{i=1}^{N_i}\frac{dL_i}{dW_2}$$

Gradient ascent update:

We use gradient ascent to update the weights of the neural network.

$$W_1 = W_1 + \epsilon\frac{1}{N_i}\sum_{i=1}^{N_i}\frac{dL_i}{dW_1}$$

$$W_2 = W_2 + \epsilon\frac{1}{N_i}\sum_{i=1}^{N_i}\frac{dL_i}{dW_2}$$

Figure 2 showcases the decrease in the loss function after 100 epochs for one of the subsets as an example.

Prediction Process:

After the weights W_1 and W_2 are learned, I used them in order to predict the 30th song of each user. The following schema is what I used to generate the prediction results.

- For each user in the subset data, averaged the one hot encoding vector representation of all 29 artists the user played.
- I then used that new vector as input vector and completed one forward pass in the neural network with learned weights W_1 and W_2.
- The output is a vector with the same dimension as the number of artists in the repertoire and each entry represents the probability that the artists at the corresponding position would be the most likely to be played along in a playlist which contains the 29 artist used as input.
- I then selected the artist with the highest 10 corresponding probabilities as my 10 best predictions.

Potential improvements that can be made:

A recurrent neural network (RNN) would be a better model for this prediction problem. In part two of the project, implement a (LSTM) model.

I also think that I can improve the current model by running more epochs and increasing the number of hidden neurons.

Finally, to get very good prediction accuracy, more data should be provided along with the playlist sequence. For instance, information about the genre each artist fall into if provided could be used as side information and included in training a neural network for prediction purpose.

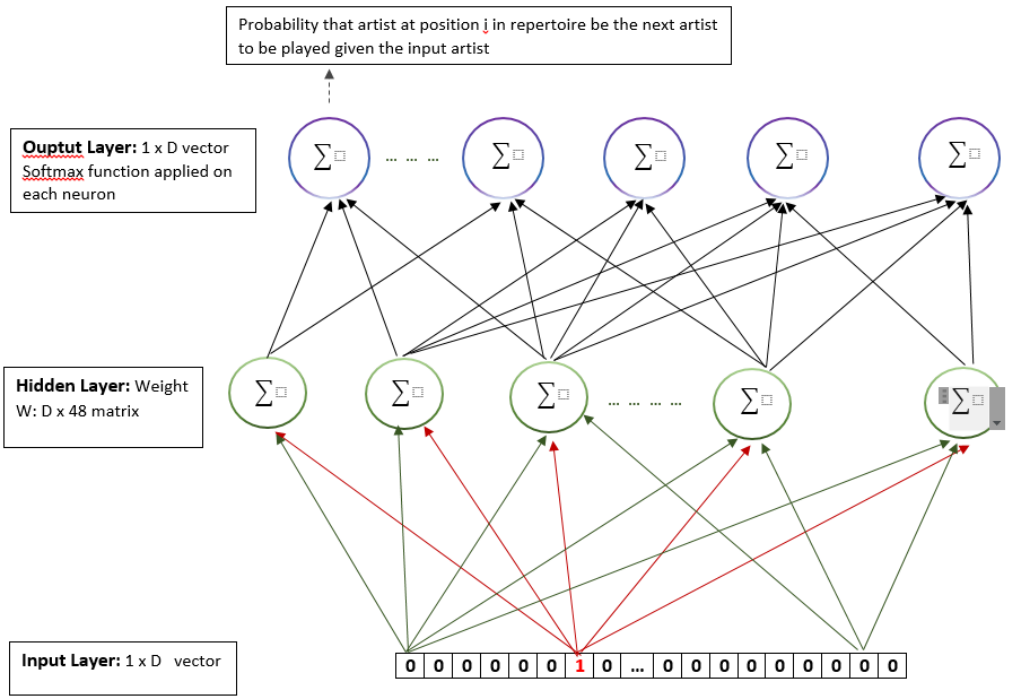

Figure 1: Wor2Vec Neural Network Architecture (fully connected Network)

Note: D represents the number of artists in the artists repertoire. This value will vary for each subset of the data. Refer to Table 1 for the exact value of D for each subset

Figure 1

Subset #	1	2	3	4	5	6	7	8
Number of users in subset	60	60	60	60	60	60	60	60
Number of unique artists in repertoire	829	893	813	883	811	850	652	793
Subset #	9	10	11	12	13	14	15	16
Number of users in subset	60	60	60	60	60	60	60	72
Number of unique artists in repertoire	880	893	899	880	903	896	882	1069

Table 1: Number of unique artists in subsets

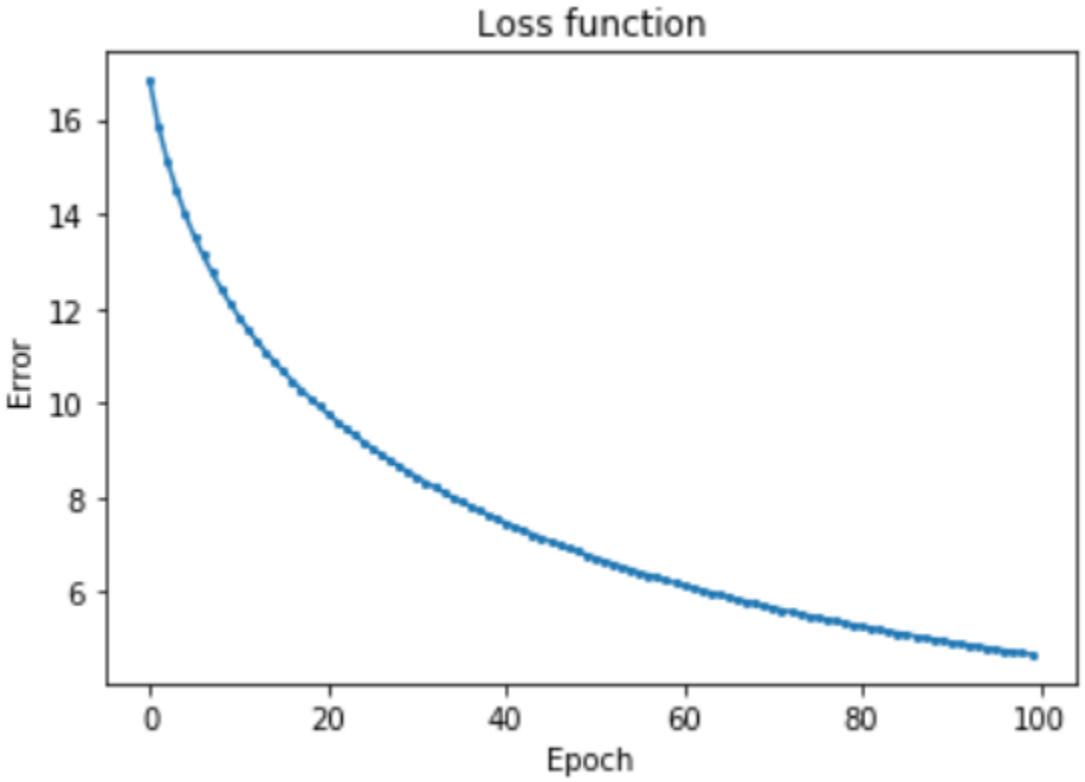

Figure 2: Neural Network loss function after 100 epochs