Playlist Generation in a Neural Network

by
Sebastian Rodriguez

Intro to Parallel Distributed Processes
85 – 419 Final Project
11 May 2019

TABLE OF CONTENTS

Abstract

The consumption of music, and the specificity which it is listened to, have been in high-demand as a result of today's era of ubiquitous technology. The creation of personalized playlists has largely become the product of neural networks that analyze the behavior of music listeners, as well as a combination of other methods (due to resource limits, this paper will exclude these other methods). The goal of this research is to analyze *user-behavior* to generate a personalized playlist that meets a user's expectations. This paper utilizes LENS to create a *feed-forward (FFD)* and *recurrent (REC)* neural networks that independently generate playlist(s) for an individual user, using one song as the input criterion.

Introduction

For my final project in Introduction to Parallel Distributed Processing I am interested in how music providers like Spotify, Apple Music, and Pandora create a subset of songs, dubbed a playlist, that somehow captures the subjective preferences of a human listener. The goal is to capture the intuition of human-made playlists using computer-made models. By breaking down each song into numerical metrics, such as beats-per-minute (BPM) and energy, each song has a unique 'fingerprint' in the form of seven song features. This data pre-processing was done using a website created with the Spotify API, sortyourmusic.playlistmachinery.com, that takes a set of songs, and creates a matrix of

corresponding values for each song (figure 1). The raw values of each feature has mean values ranging from approximately -7 for the loudness feature, to a mean of 150 for BPM.

Track Name	BPM	Energy	Dance	Loud	Valence	Acoustic	Pop
Gravity – Feat Flyboy	120	56	63	-8	15	14	54
Midnight City	105	71	53	-7	32	2	69
Parallel Jalebi	133	70	65	-15	51	69	51
Gold	113	41	60	-9	41	62	71
Jubel – Orginal Mix	125	48	70	-7	18	19	61
Say My Name – JW Remix	115	69	80	-4	11	0	44
Fantasy – Felix Jaehn Remix	123	78	64	-7	25	1	51
Your Soul	94	68	49	-8	8	0	65
Cinema – Skrillex Remix	145	98	64	-4	43	1	61
Get Free	88	66	61	-7	77	10	56

Figure 1. Data from Sortyourmusic.playlistmachinery.com (RAW values). (1)

Using these raw values could hamper the learning of the network, so I decided to standardize the values of each feature. The values were standardized to values from -2 to 2 for 95% of values by using the mean and standard deviation of each feature to create a z-score for each value (figure 3). The list of z-scores for a song are then used in LENS and fed into each separate network as inputs. The target outputs for both networks consist of 3-5 songs chosen based on my opinion of which songs go together —using meta-data, these target values can more accurately reflect the preferences of a user; importantly, for this paper, it was impractical to curate such information, and so the aforementioned approach was used instead. This lack of meta-data may impact the finding in this paper (this will addressed further in the Discussion section).

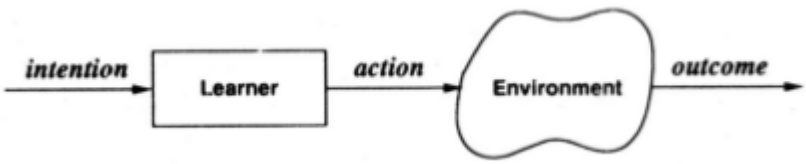


Figure 2: Visual Depiction of Conceptual Feed-Forward Network. (2)

In the first simulation, a feed-forward network (figure 2) was used in an attempt to activate songs based on one song as input. This network is run using LENS, and is constructed as a network with 7 input units, 7 hidden units (context units), and 490 outputs units. In this network, each output unit corresponds to a song. The 7 context units take the 7 input features in for each song and create its own similarity distribution, which then is used to activate target songs. For each song, four to five target sons are given. The targets for each song is, crucially, a subjective process. The targets are taken from a user's preferences (in this paper, my own) to activate other songs with overlapping similarities. The network itself is trained on a set of 490 songs using weight decay to help prevent over-fitting the target set. In the format of a feed-forward network, the internal (forward) model of the world generates action(s) randomly to predict outcome. In this case, the prediction is which songs should be activated, and the optimal outcome is the target songs for each input song. The network uses the difference between the predicted and actual outcomes as the "error signal," or the magnitude of error the predicted outcome has achieved. Feed-Forward Network are, generally speaking, trained quickly, using only half a second to train 10 epochs in the network. This is an FFD networks greatest advantage; however, the depth of its learning is limited for the same reasons it is efficient. An FFD Network has communication between layers, but not within each layer. Meaning FFD networks send values from the input to the output through its hidden layers, but units with each layer do *not* share activations with each other. As a result, too many hidden units can easily cause an FFD network to over-fit to the data and fail at generalizing the learned trends. Advantageously, the FFD network can easily handle a large set of songs as output, as this FFD network has 490 output units, while operating at effectively the same speed as an output layer with 30 songs.

In the second simulation, a Recurrent Network (Figure 3) is used in an attempt to prevent the over-fitting of the targets. A Recurrent Network is a network that has cross-talk within layers—one of the lacking elements of a feedforward network. On the other hand, training for an REC network is slower than that of a FFD network by a large margin—for a 490 output REC network, training 10 epochs took 40 seconds (approx.), which is 80-times slower than 10 epochs in its counterpart. In order for this to happen efficiently, the network is made of 7 inputs units (of z-scores), 20 hidden units, and 100 output units (390 less output units). The Recurrent network learns much more slowly than the prior simulation, but its learning has more breadth than an FFD network. Although the input sends information to the output units through its hidden layer, each unit within each layer *does* affect all other units within its respective layer. Importantly, the similarities between outputs are shared between each other, preventing any one song from becoming either too active or inactive. This cross-talk will hopefully allow the network to generalize and create playlists reflecting more subjectivity than the Feed-Forward network. On the other hand, the smaller output set limits the extent the network can learn, as less examples are examined in the network. This limitation may prevent the capture of trends within the data.

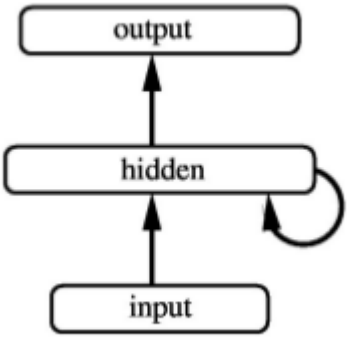


Figure 3. Recurrent Network, Depiction. (3)

Simulations & Methods

Note: The simulations do not use human subjects/participants in any capacity, excluding the author.

Input Data

In the experiment, a network (feed-forward or recurrent) was given an input of one song, decomposed into the form of seven features: Beats-per-Minute (BPM), Energy, Dance, Loud, Valence, Acoustic, and Pop (4). Each feature is given raw values, calculated from the sound profile of each song. Beats-per-minute is also known as the tempo of the song. Energy refers to the energy of a song; further, the higher the value, the more energetic the song is. The Dance feature relates to how easy a song is to dance to—the higher the value, the easier it is to dance to a song. The Loud feature refers to the sheer sound of the song, and greater the value, the louder the song tends to be. The Valence feature refers to the mood of a particular song—the greater the value, the more positive the mood of the song is. The Acoustic feature is, somewhat obviously, refers to the acoustic nature of a song, and the greater the value the more acoustic a particular song is. The Pop Feature is also known as the popularity feature. The Popularity feature grows in value in direct relation to how popular a song is (that is, how many people listen to/have listened to this song).

Track Name	BPM	Energy	Dance	Loud	Valence	Acoustic	Pop
Gravity – Feat Flyboy	120	56	63	-8	15	14	54
Midnight City	105	71	53	-7	32	2	69
Parallel Jalebi	133	70	65	-15	51	69	51
Gold	113	41	60	-9	41	62	71
Jubel – Orginal Mix	125	48	70	-7	18	19	61
Say My Name – JW Remix	115	69	80	-4	11	0	44
Fantasy – Felix Jaehn Remix	123	78	64	-7	25	1	51
Your Soul	94	68	49	-8	8	0	65
Cinema – Skrillex Remix	145	98	64	-4	43	1	61
Get Free	88	66	61	-7	77	10	56

Figure 1 [REPRINTED FOR CLARITY]. Data from Sortyourmusic.playlistmachinery.com (RAW values). (1)

As briefly covered in previous sections, the values for each feature range greatly in value, and have less statistical relevance in their raw states. To modify the data sets so that the values are related to each other, I decided to standardize the data, using the following formula:

$$x_{new} = \frac{x - \mu}{\sigma}$$

Figure 4. Z-Score formula. (5)

The above formula takes in the raw value, mean and standard deviation of its respective feature. In this syntax from figure 4, 'X' is the raw value (for example, a single Dance value), 'μ' is the mean of the category (for example, the mean of all Dance values), and 'σ' is the standard deviation of the category (furthering the prior example, the standard deviation of all Dance values) (5). After using this process to standardize all values within the data set, the values look as follows:

Track Name	BPM	Energy	Dance	Loud	Valence	Acoustic	Pop
Gravity – Feat Flyboy	-0.006	-0.491	-0.238	0.012	-1.304	-0.396	-0.150
Midnight City	-0.536	0.506	-1.098	0.409	-0.673	-0.942	0.828
Parallel Jalebi	0.453	0.440	-0.066	-2.770	0.033	2.105	-0.345
Gold	-0.253	-1.488	-0.496	-0.386	-0.339	1.787	0.959
Jubel – Orginal Mix	0.170	-1.023	0.364	0.409	-1.193	-0.169	0.307
Say My Name – JW Remix	-0.183	0.373	1.225	1.601	-1.453	-1.033	-0.801
Fantasy – Felix Jaehn Remix	0.100	0.972	-0.152	0.409	-0.933	-0.987	-0.345
Your Soul	-0.924	0.307	-1.442	0.012	-1.564	-1.033	0.567
Cinema – Skrillex Remix	0.876	2.301	-0.152	1.601	-0.264	-0.987	0.307
Get Free	-1.136	0.174	-0.410	0.409	0.998	-0.578	-0.019

Figure 4. Created by Author, using data collected from (1).

These values now range from -2 to 2, depending on the mean and standard deviation of each feature. Importantly, the values that had greatly different magnitudes, such as the BPM and Loud features, will now have equal influence on the network during training and analysis. After standardization, the values for each song's feature now reflect how many standard deviations the value is from the mean. This step

is crucial to the network, otherwise values with great magnitude would overly effect the network, and, for instance, beats-per-minute may have become the dominant factor in each song. Importantly, I opted to standardize the data instead of normalizing the data because normalization loses information about the outliers in a dataset, but also tranforms all values to range from [0,1] (6). For the uses of each value, their relation to other values in each category was more important than all values having related values —this is because values from different features are used to influence the network, but are not strongly interacting with each other directly. Normalization could be more useful in other situations of analyzing this same dataset, but is less relevant for the purpose of this research.

Simulation 1: Feed-Forward (FFD) Network

In this simulation, a Feed-Forward Network (FFD Network) takes in one input in the form of seven song features, and output the most active songs in the output (of 490 output units), through one seven-unit hidden layer. Each input is tasked with activating the four target outputs. The expectation was the network would activate the target units, and the target units would therein activate their targets, creating a playlist.

Methods 1:

The dataset was standardized (as described above) and input into a network in the form of the 7 standardized values for a song. These seven features are fed into a hidden layer of seven units as well, and fed into 490 output units. Each training example has the following structure:

```
name: "405_California"
I: -0.5270 0.6492 -0.9184 0.4510 -1.2977 1.4898 -0.3721
t: 208 210 114
;
name: "406_Freddie_Freeloader_-_Studio_Sequence_2"
I: 0.5257 -2.2146 -0.7898 -2.9379 0.6865 1.2014 -0.4735
t: 256 357 81 233
;
name: "407_Smash_Into_You"
I: -0.6289 -0.5590 -0.4041 0.0744 -1.8514 -0.4969 0.0335
t: 413 7 282 487
;
```

Figure 5. Songs 405, 406, and 407 from training set (in LENS syntax).

The Input values ("I:") are in the order of {BPM, Energy, Dance, Loud, Valence, Acoustic, Pop}, and the input for song "California" is {-0.5270, 0.6492, -0.9184, 0.4510, -1.2977, 1.4898, -0.3721}, and the output is songs (numbered in the training set) {208, 210, 114}, which correspond to songs: "The Recipe (Black Hippy Remix) – Bonus Track," "Sex, Love, & Money (Album Version) – Edited," and "If There is Someone Lovelier Than You." Each training example has this same structure. The FFD Network takes in the 7 values, and in the output activates most strongly the three aforementioned songs, in the case of the song "California." Further, the network (figure 6), after 0 training epochs, looks as follows (for the song "Tokyo Files").

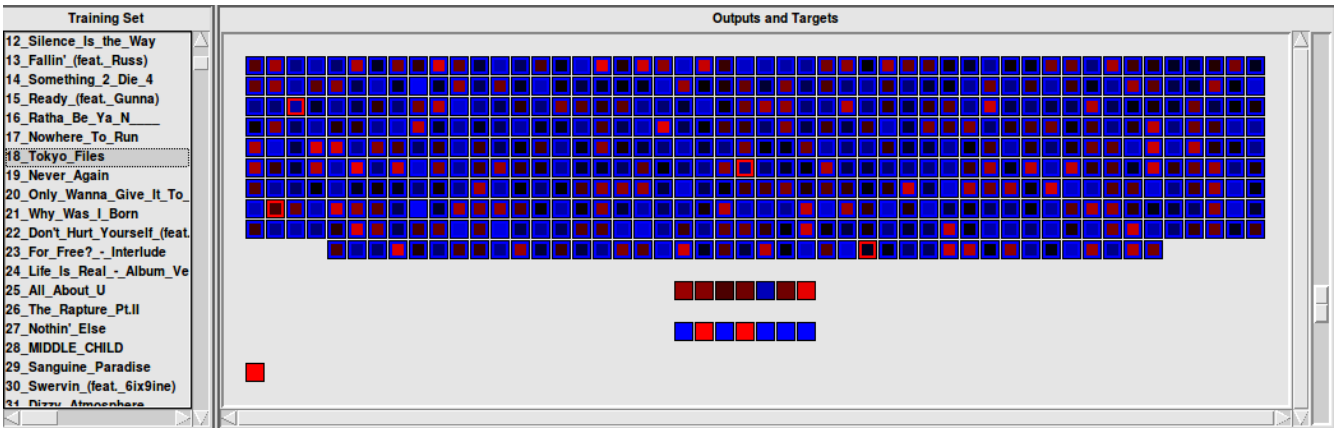


Figure 6. From top to bottom: output units, hidden lay, input, bias (input song).

In order to train this network, a weight-decay must be added so that the network does not over-fit, and as a result lack in generalization. The weight decay value used is 0.025, such that activation for other output units do not all remain too active, but some of the more related output units remain active. The network trains too quickly under Steepest Momentum, and so the first 200 epochs are trained using Doug's Momentum, and then 10 epochs at a time with Steepest Momentum. The first 200 epochs of Doug's Momentum activates *all* output units to approximately .417 (figure 7).

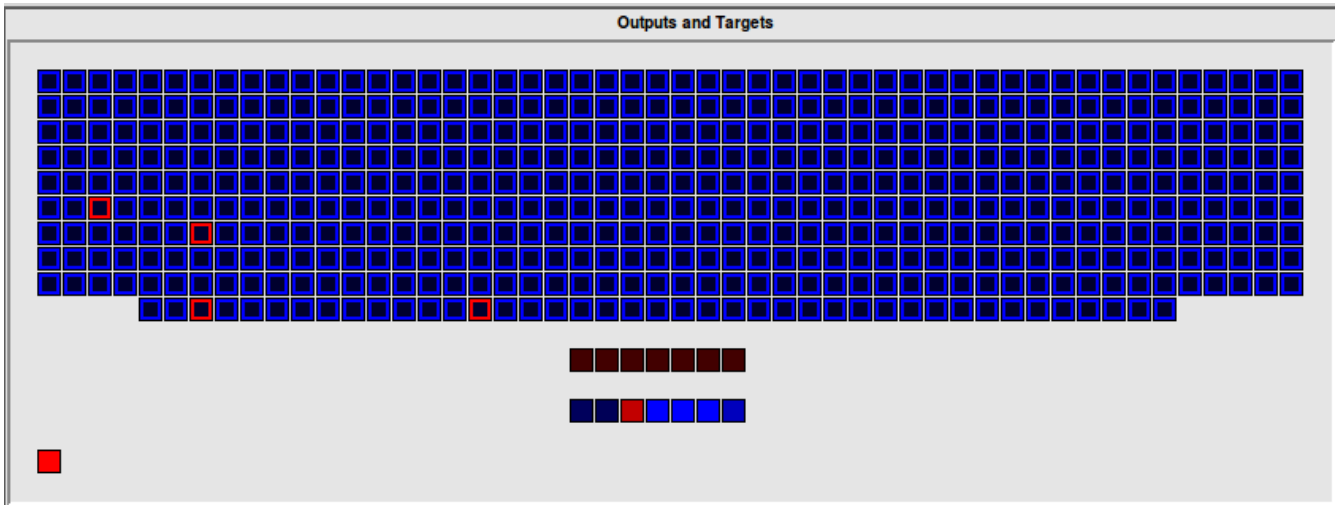


Figure 7. After 200 epochs of training with 0.025 weight decay, and Doug's Momentum.

Thereafter, the activations of all output, except a few, drop to 0. The active outputs that are not targets change with every 10 epochs, and are the same for every track (Figure 8). These activations seem almost arbitary, and change every 10 epochs. As increasingly more epochs are run, all output units ,except for the targets for each input, begin to equalize and drop to 0.

Figure 8. Activations for “Habitat” after 240 epochs (40 under Steepest Momentum).

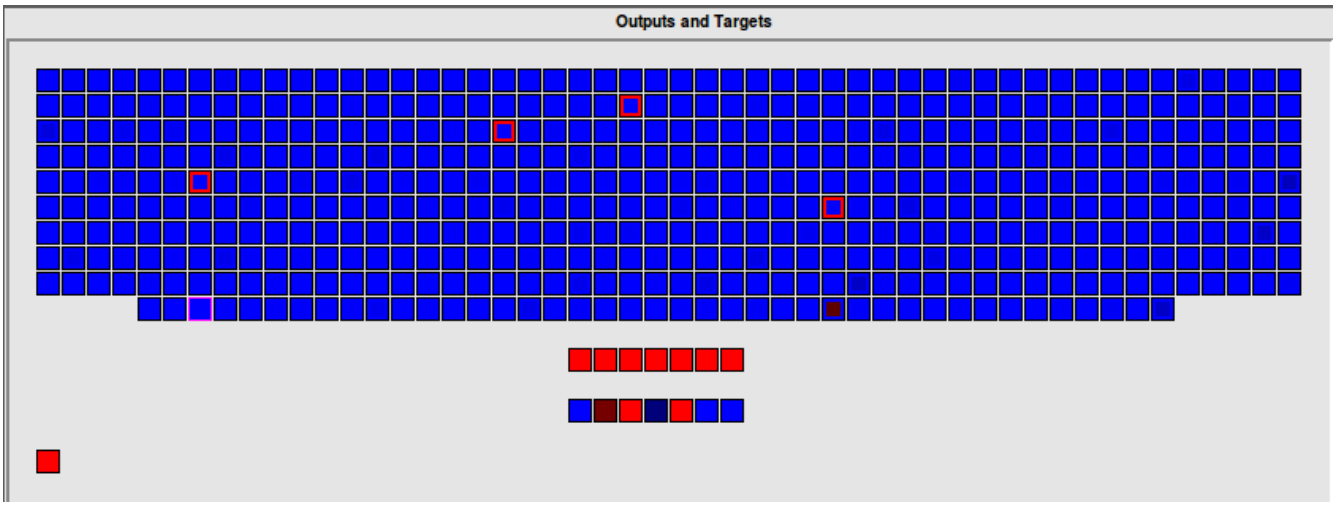
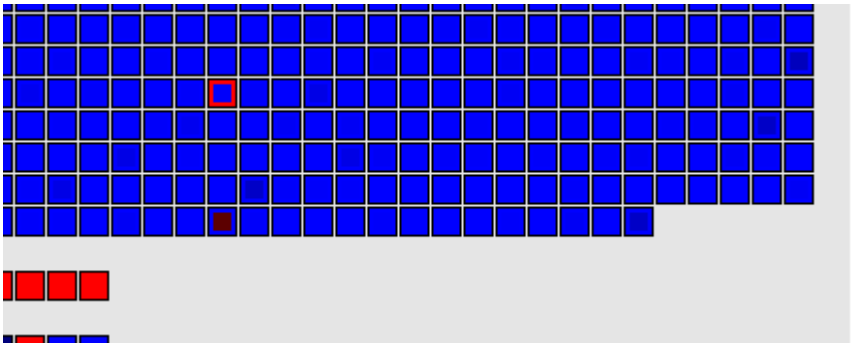


Figure 8.1 Close-up of aformentioned songs from figure 8.



Interestingly, the songs most active in all instances after 240 epochs are Songs #477, 432, 490, 348, 249 with respective activations of .682, .114, .116, .113, .139. All of these activations are more strongly active than any song’s respective targets. After another 10 epochs, the most active songs (uniformly active for all input instances) change again (figure 10).

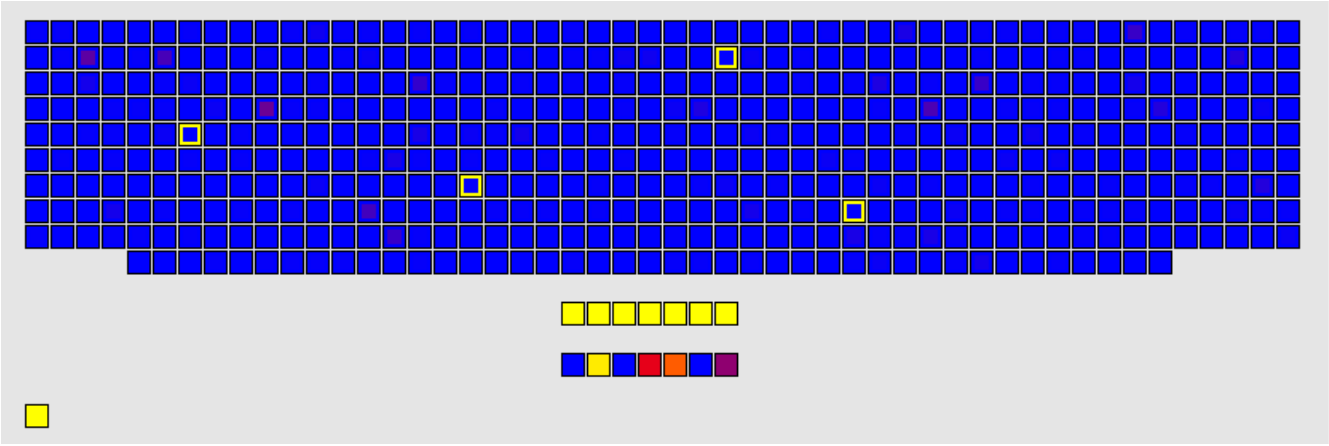


Figure 10. Activations after 240 epochs, 40 in Steepest Momentum. (Palette changed for visibility).

In epoch 240, there are 9 active songs: 52_Who_is_Themba, 55_Westside_Bound_3, 159_The_Pap, 115_Chopstix_(with_Travis_Scott), 43_I_Ain't_Mad_At_Cha, 137_California_Love_(remix), 185_All_of_You, 363_It_Never_Entered_My_Mind_Live_at_Pasadena_Civic_Auditorium _Pasadena _CA_-_February_1956, 414_Rise_'N'_Shine. Up to epoch 320, each 10 epochs yields another unique subset of active songs that are not the targets of any input and are present for all inputs. After epoch 320, some inputs have output unit (song) activations specific to a song.

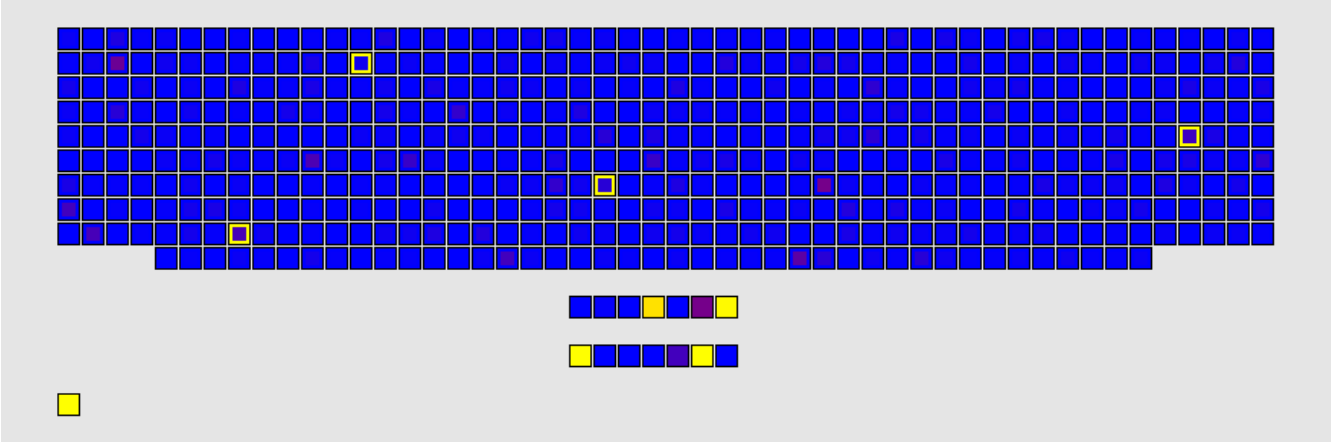


Figure 11. 488_Countdown yields multiple active output songs (and other inputs do not).

This development in the network is noteworthy, because, up until this point, the network has had no unique playlist activations for a single song, but instead was outputting a single playlist for the entire network. The playlist, then, for 488_Countdown are the following songs: 52_Who_is_Themba, 260_Naima_-_Alternate_Version,_Take_2,_False_Start, 264_'Round_Midnight_-_Live_at_the_Newport_Jazz_Festival,_Newport,_RI_-_July_1955, 320_Blue_in_Green_-

_Studio_Sequence, 464_Budo, 331_6_Inch_(feat._The_Weeknd), 476_Slide_(feat._Blueface_&_Lil_Tjay), 299_GOAT.

Results 1

After 350 epochs, where the first 200 were trained using Doug's Momentum, the networks error plateaus and continues at the same error level (y-axis) indefinitely. Importantly, 488_Countdown was not unique in having multiple active songs specific to its network—Countdown's activation were just the most visible. This is a result of the network having such small activation values, so small that the proposed playlists for each song were hardly visible regardless of palette type.

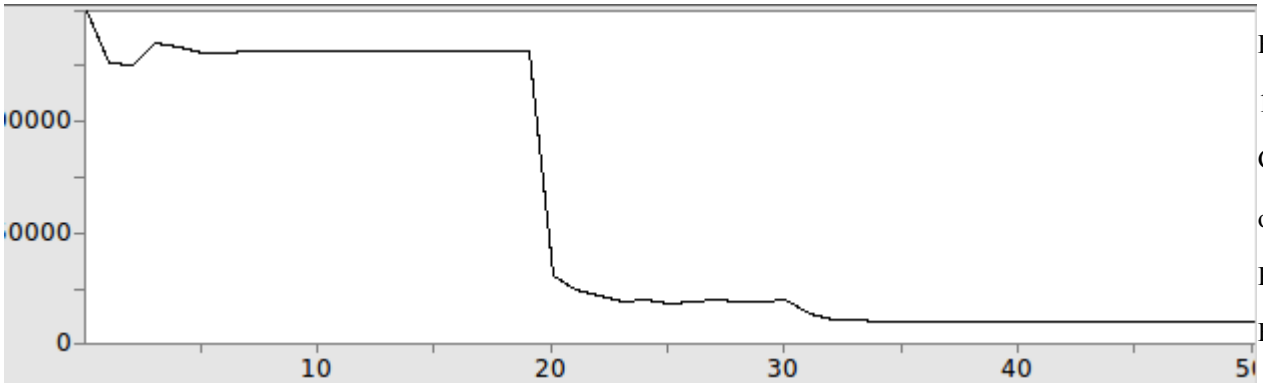


Figure 12. Graph of Error, FFD Network. (X-axis is in multiple of ten. For instance, time period 20 is epoch 200, where Doug's Momentum is switched to Steepest Momentum.

The training in this network yields playlists for each song, strongly influenced by my subjective preferences. As this was the main goal of this paper, this is an important accomplishment to note; however, the activation strengths for each song are quite low, and could be better represented by another network. With this possibility in mind, a recurrent network may do this better. In terms of the testing sets, there is always a 0% accuracy because of how the results of each input instance is

interpreted. The testing set tries to see if the 3-5 target units are highly active, but the human-interpretation is all the *other* active units that could be added to the playlist, in addition to the targets and the input song.

Simulation 2: Recurrent (REC) Network

In this simulation, a Recurrent Network (REC Network) takes in one input in the form of seven song features, and output the most active songs in the output (of 100 output units), through one seven-unit hidden layer. Each input is tasked with activating the four target outputs. The expectation was the network would activate the target units, and output (target) units would communicate with each other across songs, creating more highly active playlists for interconnected songs. The pre-processing of the datasets for this network are identical to the FFD Network, so refer to the first paragraph under *Methods 1* for pre-LENS setup. The only distinction is that the REC network will be using 390 fewer songs, so the computation time is more efficient.

Methods 2

To train this network, the same approach as the FFD network was used. Specifically, the network is trained with a weight decay of 0.025 for 200 epochs using Doug's Momentum, then trained using Steepest Momentum. Before training (epoch 0), the network looks as follows, for song 3_On_Green_Dolphin_Street:

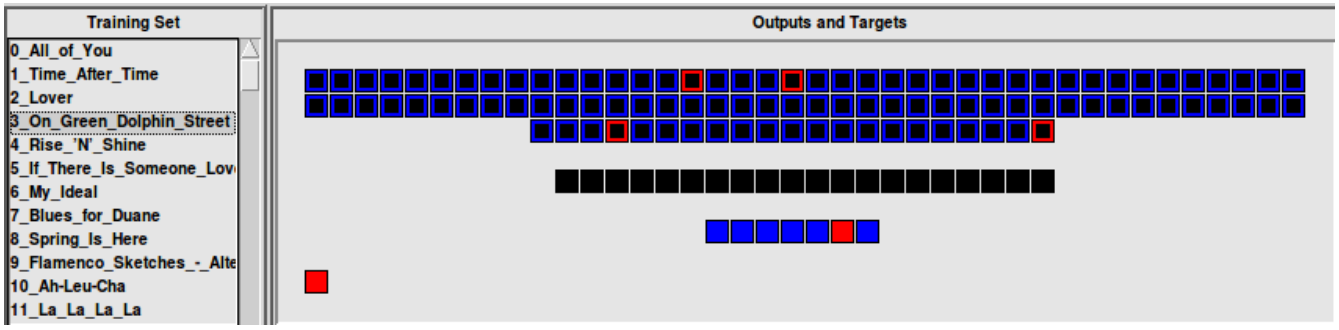


Figure 13. Recurrent Network BEFORE training, "On Green Dolphin Street."

Because the REC learns more slowly, the weight decay = 0.0025 starting at epoch 450. The same steps were taken in the recurrent network so that there are less variables to compare the effectiveness of the two types of networks. Regardless of training methods, however, the REC Network is ineffective at generating song-specific playlists. After 2000 epochs of training while decreasing weight decay until it eventually rests at 0.00025 for the last 500 epochs, there is no unique activations for specific inputs.

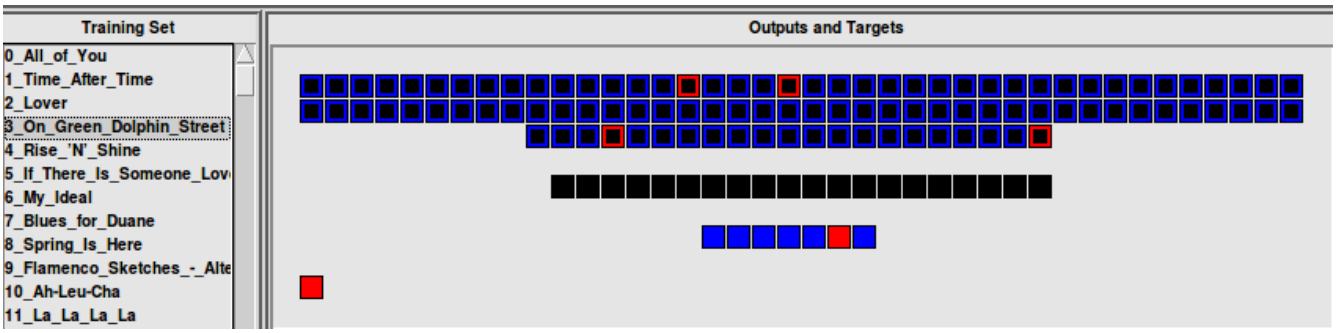


Figure 14. "On Green Dolphin Street," after 2000 training epochs.

Results 2

The cross-talk among outputs have actually worked to eliminate any input specific activations, opposite the hypothesized result of using a recurrent network (figure 14). The network itself does not over-fit data by reducing error too significantly, but rather the communication between output muddles

any noticeable preference generations. Visually, the activation output for "On Green Dolphin Street," seem to remain the same after 2000 training epochs (as is the case for all inputs). Further, the training of this network is slower, and provides almost no insight into creating a playlist from song preferences. Below is the error graph reflecting the decrease in error over 2000 epochs.

Discussion

In all aspects, the Feed-Forward Network had better results, given the way this experiment was constructed. Although the FFD Network is not working optimally, there are key insights that can be gleaned from the activations from each song, and playlists can be created from these generated output activations (figure 11).

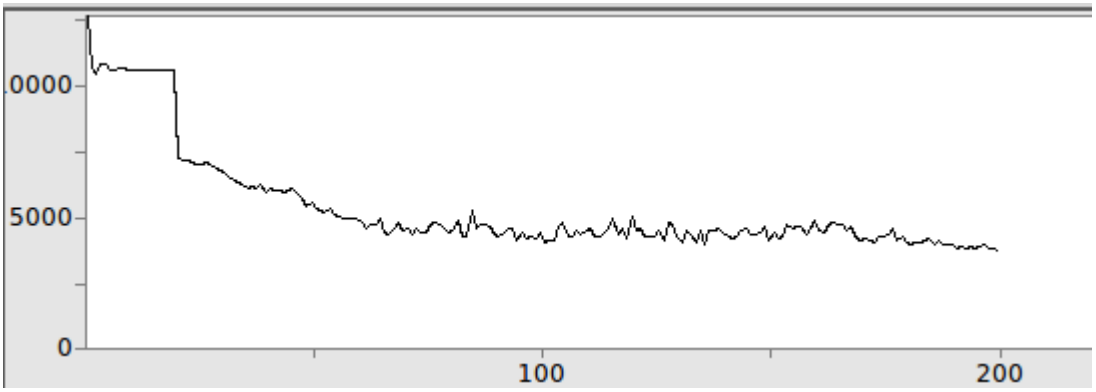


Figure 15. Error Graph, after 2000 epochs. (For clarity, leftmost downward spike is where Doug's Momentum is changed to Steepest Momentum).

The Recurrent Network was disappointing in practice, but this is a result of some of the limitation of my resources. Access to behavioral preferences of other music listeners (but the the same songs) could also have provided key information for either/both of the networks to learn possible songs in a playlist

better. Further, the “other” methods for generating playlists could have proved beneficial; the two major methods are Natural Language Processing and a (more rigorous) Audio Processing.

In terms of potential improvements, The dataset for the FFD could be increased by a couple thousand, and, as aforementioned, include different targets for the same songs, based on other users’ preferences. This could help to provide a user with tastes that are new yet meet the user’s expectations of a playlist. Notably, the clear lack of access to meta-data may have been a leading reason in the lack of clear-cut activation, and may have been an essential element in the Recurrent networks failure to generalize. Importantly, the FFD Network did indeed generate playlists given my preferences, albeit rather obscurely. In order to test the results of the FFD Network, the author would need to create playlists given the information from inputs (like from figure 11), and listen to them to see if their subjective tastes were well-met.

References

1. Lamere, Paul. *Sort Your Music*, Spotify API, 9 Dec. 2018, sortyourmusic.playlistmachinery.com/index.html.

2. Jordan, Michael I, and David E Rumelhart. *Forward Models: Supervised Learning with a Distal Teacher*. 1992.

3. Elman, Jeffrey L. *Finding Structure in Time*. UC San Diego, 1990.

4. Hipolito, Abigail. "Spotify: Analyzing and Predicting Songs." *Medium*, ML Review, 8 Nov. 2017, medium.com/mlreview/spotify-analyzing-and-predicting-songs-58827a0fa42b.

5. Saitta, Sandro. “Standardization vs. Normalization.” *DataMiningBlogcom*, 10 July 2007, www.dataminingblog.com/standardization-vs-normalization/.

6. “What's the Difference between Normalization and Standardization?” *Cross Validated*, 2011, stats.stackexchange.com/questions/10289/whats-the-difference-between-normalization-and-standardization.