Hit Song Identifier Based on Song Features

Meggie Hodgson, Serena Khanna, Ethan Peters, and Duncan Stuart

Introduction

Will this song be successful?

Popularity makes or breaks an artist's career. Song writers are constantly competing to cater to customers, but only a fraction of them are able to top the charts. The only indication of a song's success is based on industry knowledge of trends and perhaps small focus groups that listen to the music before release. What if a trained artificial neural network (ANN) can determine the probability of a new song's success. This would allow songwriters to test and optimize a songs before releasing it to the public, thus maximizing its potential to top the charts and generate more profit.

Four different ANNs were trained and tested on 200 songs in 5 different genres: pop, rock, country, electronic, and indie to classify songs as either successful or unsuccessful.

Data Collection

Input features were defined and provided by Spotify. 12 input features were used being: Danceability, Energy, Key, Loudness, Mode, Speechiness, Instrumentalness, Liveness, Valence, Temp, Duration, and Time Signature. Popularity, calculated in Spotify based on number of plays over time and recently, was used to define success as a popularity score threshold. Song selection was based on spotify's song recommendation algorithm when prompted with a genre.

Growing Feed-Forward Network

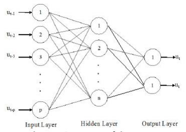


Fig. 1 GFFN Architecture

As a baseline, each genre is applied to a 3-layer feed-forward network with using error back propagation learning. Input layer size is chosen as 12 as per the number of input features in the data, and output layer size is chosen as two (one for popular and one for not popular). Hidden layer size is initialized to zero and the network is trained until total error reaches a minimum. Hidden layer size is increased and the process is repeated until an increase in the featurization of the network ceases to appreciably the proportion of true positives to data size. For each genre, this converged around a hidden layer size of 6, which was used unanimously to generate the results of the network for comparison purposes.

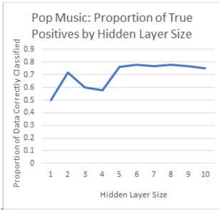


Fig. 2 Sample Hidden Layer Growth and its effect on accuracy

NeuroEvolution of Augmenting Topologies

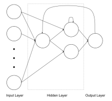


Fig. 8 NEAT Example Architecture

A strategy to determining the most effective structure/topology of an ANN (artificial neural net) is making the ANN build itself. One of these methods is NEAT (Neural Evolution of Augmented Topologies (4)). NEAT uses a biological model to assign individuals of a population to several species. These species/individuals then "compete" to become the fittest based on how they perform on the dataset. Instead of using conventional feed-forward weight optimization strategies such as back propagation, NEAT uses mutation and crossover between individuals. NEAT is unique compared to other neural evolution methods in its use of genetic encoding, tracking connections through historical markings, the protection of new connections through speciation, and minimizing dimensionality through incremental growth from a minimal structure. Figure 7 shows the average number of nodes (top-right) increases while the error (bottom-right) is fluctuating but ultimately decreasing. NEAT's evolution can provide a number of types of connections including: feed-forward and backward, residual, and recurrent. (Figure 8)

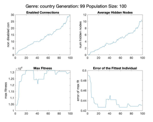


Fig. 7 Analysis of Evolution over generations

Probabilistic Neural Network

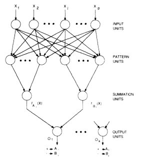


Fig. 4 PNN Architecture (2)

Probabilistic Neural Networks (PNNs) operate within 4 layers for class prediction using the parent probability distribution function. PNNs have an input layer, pattern layer, summation layer and output layer (figure 4). In this implementation, each song, or datapoint, has 12 input features requiring 12 input layer nodes. The pattern layer has n number of nodes where n is the number of training data points as each song has a unique feature pattern (n = 199, 200 dependent on genre). Each pattern node implements a Gaussian function to map a peak probability in a 12-dimensional function. Every peak per class is summed in the summation/category layer. A visual representation of a 2D probability of summated peaks is shown in figure 5 with different smoothing factors, sigma values, applied. The output layer then takes the class of the highest probability (highest peak) at the vector location of a test data point and returns that class. This architecture does not need a training step due to the mapping of pattern and class peaks, thus allowing for a very fast run time that will always converge onto the optimal class.

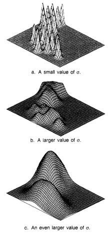
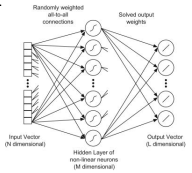


Fig. 5 Class Peaks (2)

Extreme Learning Machine

The Extreme Learning Machine (ELM) Neural Network randomly chooses hidden nodes and analytically determines output weights for feedforward networks with a single hidden layer. This type of neural network is unique from standard feed-forward networks as it contains a large number of hidden nodes whose parameters are randomly generated independently from the training samples. An ELM network contains an input layer, a wide hidden layer, and an output layer (figure 6).

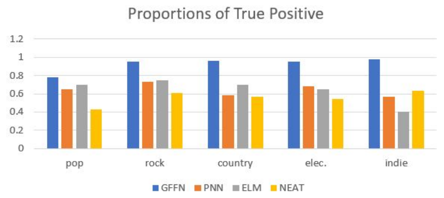
In an ELM, the weights from the input to hidden layer are randomly generated. The output weights are computed using the dot product of the input and input-to-hidden weights, and applying a sigmoid activation function. A least square error regression formula is then applied using this input-to-hidden matrix, as well as the training classes. This works to minimize the least square error between the predicted and actual class during training. Then, to test on the remaining data, the dot product of the input-to-hidden matrix and output weights is computed. This approach offers an extremely fast training algorithm, resulting in a very low runtime.

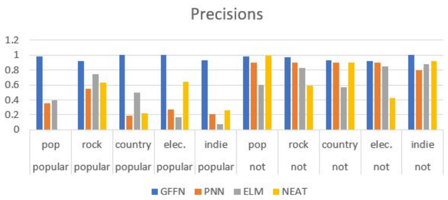


Randomly weighted all-to-all connections
Solved output weights
Input Vector (N dimensional)
Hidden Layer of non-linear neurons (M dimensional)
Output Vector (L dimensional)

Fig. 6 ELM Architecture (3)

Results





Conclusions

What we've learned

The neural networks consistently classify unsuccessful songs correctly at a higher rate than successful songs. This relationship is demonstrated by the precision and recall of each class. This is consistent with the results of previous papers on the subject (4). The relatively higher precision of unsuccessful classification causes the average precision and recall to be dependent on the randomly selected songs for the test set and so the ratio of unsuccessful to successful data points.

This phenomenon of easily identifying unsuccessful songs relative to successful songs implies it is easy to identify when a song will not appeal to audiences, but it is difficult to identify when a certain combination of features will yield success. This observation is consistent with the concept that it is relatively simple to identify when a song's pattern signature is not cohesive and will not be successful. However, a cohesive song pattern does not guarantee success.

Next steps

It is clear from the results that more variables are required to accurately predict success. In such an artistic medium, it may be difficult to quantify the variables indicative of success, especially cross-genre. By using the songs themselves as raw inputs, the guess work of which features are relevant can be eliminated.

Limitations

There are some limitations to the data that affect the accuracy of the neural network implementations for classifying hit songs. Our dataset is small as it uses 1000 songs out of the available 10 million songs on Spotify. There is also survivor bias against the low popularity ranked songs as unpopular songs are less likely to be recommended. The songs in the dataset had an average popularity of 60 out of 100. Furthermore, our dataset does contain some duplicate songs. The experiment also has inductive bias as a limited number of parameters (12) is used which may not be be enough information to capture what makes a song successful within its genre.

References

1. Stanley, Kenneth O., and Risto Miikkulainen. "Evolving Neural Networks through Augmenting Topologies." Evolutionary Computation, vol. 10, no. 2, 2002, pp. 99–127., doi:10.1162/106365602320169811.
2. Specht, D. F. (1990). Probabilistic neural networks. *Neural Networks,3*(1), 109-118. doi:10.1016/0893-6080(90)90049-q
3. Tissera, M. D., & McDonnell, M. D. (2016). Deep extreme learning machines: supervised autoencoding architecture for classification. Neurocomputing, 174, 42-49.
4. Nasreldin, F. (2018). Song Popularity Predictor https://towardsdatascience.com/song-popularity-predictor-1ef69735e380