Hit Song Identifier Based on Spotify Song Features

|  |  |
| --- | --- |
| Meggie Hodgson | 10144700 |
| Serena Khanna | 10145609 |
| Ethan Peters | 20021583 |
| Duncan Stuart | 20016104 |

CONTENTS

|  |  |
| --- | --- |
| Abstract……………………………………………………………………………………. | 3 |
| Motivation…………………………………………………………………………………. | 3 |
| Related Works…...………...………………………………………………………………. | 3 |
| Duncan…………………………….………………………………...…...…………... | 3 |
| Ethan……………...………….………………………………………...…………….. | 4 |
| Meggie…..…………………….………………………….……………...………….... | 4 |
| Serena……………..………….………………………….……………...……………. | 5 |
| Dataset…………..…………………………………………….………………………….... | 6 |
| Methods……..………………………………………………….………………………….. | 7 |
| Growing Feed-Forward Backpropagation Network……….…..……………………... | 7 |
| NeuroEvolution of Augmenting Topologies……...………………………………….. | 8 |
| Probabilistic Neural Network……..……….…………………………………………. | 10 |
| Extreme Learning Machine………………….…..…………………………………… | 13 |
| Summary and Future Works…………..…….…..…………………….…………………… | 15 |
| References……………..……………………………………………….………………….. | 17 |

ABSTRACT

Four distinct neural network architectures are presented with song feature data to build a predictor for song popularity. Spotify is used to derive features from songs, as well as a metric for deciding on popularity. Outcomes for the four architectures suggest that identifying unpopular songs is significantly more achievable than identifying popular songs, which is in agreement with previous literature. One of the four investigated architectures improves upon the state of the art.

MOTIVATION  
  
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. We investigate 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, giving songwriters more tools to pitch their songs to labels, and maximizing their potential to top the charts and generate higher profits.

RELATED WORKS

Duncan

Existing literature surrounding the application of neural networks to audio and song classification problems is relatively scarce. Lee. H. et al. say as much in their investigation into convolutional networks and their applicability to audio classification. Using a unsupervised Convolutional deep belief and restricted Boltzmann machines, their method was able to improve upon the state of the art for classifying song genre and artist, if by small margins and with a maximum accuracy of 80%. They demonstrate advancements to be made in neural-network based audio classification and “hope that [their] approach will inspire more research on automatically learning deep feature hierarchies for audio data.”

In likeness, Lee D. et al. apply a Feed-Forward Deep Neural Network to identify select for audio features in sound clips. The network was designed with two hidden layers, at 3000 neurons large, and applied supervised learning on raw audio data clips retrieved from freesound.org. Accuracy for this network with a 90/10 train/test ratio on a dataset of 1655 sound samples reached a maximum accuracy of 72.2% with size 3000 hidden layers. This prompted investigation into the applicability of deep feedforward networks to extracted audio features as opposed to the raw audio itself.

Ethan

In 2009, Monterola, et al. applied a feed-forward neural network to predict potential hit songs

resulting in an overall accuracy of 81%. Monterola, et al. used 56 features (pitch, tempo, etc.)

from 380 Original Pilipino Music (OPM) songs released from 2004 to 2006. Their method

proved to be better than the state of the art predictors at the time such as LDA and CART.

Mestyán, M. et al. provide an alternative method of predicting the success of a piece of

entertainment. Instead of using features like our process Mestyán, M. et al. use the activity level of sites such as wikipedia. This showed to be quite accurate which invites a new method to the prediction of success we could implement into our input data.

Meggie

The hit song prediction problem has been attempted before by Yang et al. in the paper, Revisiting the problem of audio-based hit song prediction using convolutional neural networks. However their methodology was quite different. Beyond implementing a Convolutional Neural Network, in this paper, the authors only used five song features with a 1000 song data set. Their best implementation resulted in 0.3 recall (Yang et al., 2017). Despite having a deep learning, promising implementation their work was likely limited by their small amount of features as well as not grouping datasets by genre.

The paper, Predicting the popularity of instagram posts for a lifestyle magazine using deep learning, explores the ability to predict the success of content on Instagram (De et al., 2017). This problem is comparable to the hit song prediction problem at hand for several reasons. Both problems revolve around creative content produced to be consumed by the masses. Both require significant engagement with their demographics to be profitable. Thus, popularity is measured by audience interaction in both problems. While in this report song popularity is measured by number of total and recent plays, De et al. measures instagram content popularity by number of likes. Additionally, De et al. used 83 feature inputs with parallel parameters to ours such as length of caption (parallel to song duration). However, the raw input of the images themselves are considered one feature, which is the anticipated future of the hit song predictor problem. Using a Deep Neural Network, with 2000 epoch, and and 83200 data points De is able to classify based on popularity at 88% accuracy. These results show promise for applying a similar approach to music and identifying hit songs.

Serena

The problem of predicting media popularity has been heavily researched. A study by Hoiles *et al.* looked at predicting the popularity of YouTube videos using data consisting of 6 million videos spread over 25 thousand channels. This study used 54 features for each video relating to the video's Thumbnail, Title, Keywords, and auxiliary channel information of the user that uploaded the video. They also took into consideration the sensitivity of each feature, some having more importance than others in the popularity of the video. They used a large dataset of YouTube videos with varying popularity, and aimed to predict their view count. They then tried out 12 different neural networks to predict YouTube video popularity. They found that the ELM had the least mean square error out of all of the networks. They then looked at the impact of changing certain features on the view count and found that a title change and thumbnail change helped significantly (Hoiles et al., 2017). This can provide a tool to YouTubers looking to increase their popularity and view count on the platform. If their video is predicted as having a low view count, they can optimize their video based on key features in order to improve it. This is similar to what our study wants to accomplish. However, instead of YouTube videos, we are looking at songs. We look at 12 key features of songs that effect it’s popularity, which could be used to help an artist before making their song public.

One study by Pham *et al.* that looks at predicting song popularity used a dataset of 2717 songs and 45 features. They chose to classify popular songs at being in the top 25% of song hotness, so they set their threshold to be 0.623, since this was the 75th percentile for their dataset. They put the data through six different neural networks and were able to achieve precision values ranging from 0.359 to 0.610 (Pham et al., 2016). However, their dataset was unbalanced as 75% of their dataset was classified as unpopular while 25% was classified as popular. I’m curious to see if our implementation, using a more balanced dataset and having a strict cutoff for song popularity classification can produce better results. It will also be interesting to see if alternative machine learning methods to this paper can improve on the accuracy for classifying popular songs.

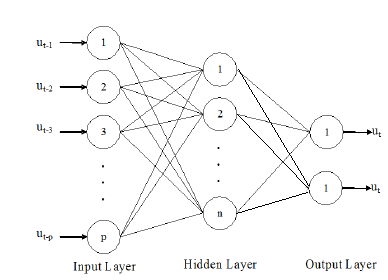
DATASET

The dataset generated consists of 500 songs, 100 songs from each of the 5 genres: Pop, Rock, Country, Indie, and Electronic. Input features were defined and provided by Spotify using algorithms supplied by The Echo Nest. 12 input features are used: 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; songs over 75 popularity were considered ‘popular’, and under 75 were not. Song selection was based on spotify’s song recommendation algorithm when prompted with a genre.

Spotify was chosen as a source for songs and featurizations for two primary reasons. Song features from spotify are publically accessible and easy to interface with using the Spotify web API, which is important; this allows professional, amateur, or aspiring musicians to easily generalize the process to within their genre, or upload and classify their own songs. As well, with over 200 million monthly active users in Q4 2018, and featurization by The Echo Nest, confidence in the accuracy of the popularity metrics and input values for Spotify songs is high.

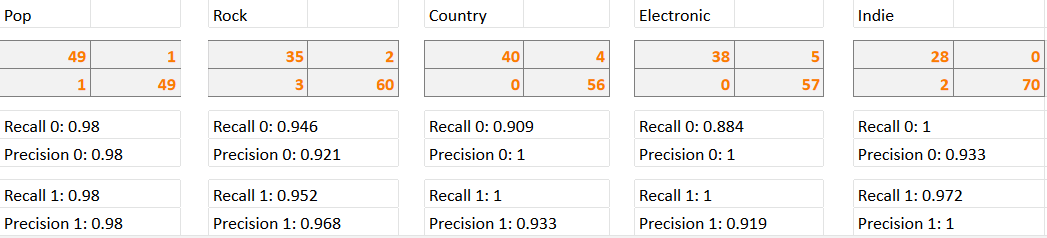
METHODS

Growing Feed-Forward Backpropagation Network - Duncan



To establish a baseline, the dataset is presented to a traditional Feed-Forward Network using Backpropagation learning, sigmoid activation function, and a sum-of-squares error metric. An input layer of 12 is chosen to match the number of input features, and an output layer size of 2 is chosen, one for a ‘popular” classification and one for a “unpopular” classification. Hidden layer size was initialized at 1 and grown while an increase in featurization continued to noticeably improve the accuracy of the network after it had converged. This resulted in a hidden layer size of 6 nodes, which was then applied to all five of the genre datasets. Training and testing sets were randomly selected based on a 80/20 ratio, and one network was trained per genre to generate the classification metrics.

Confusion matrices and precision/recall values for this network are as follows:

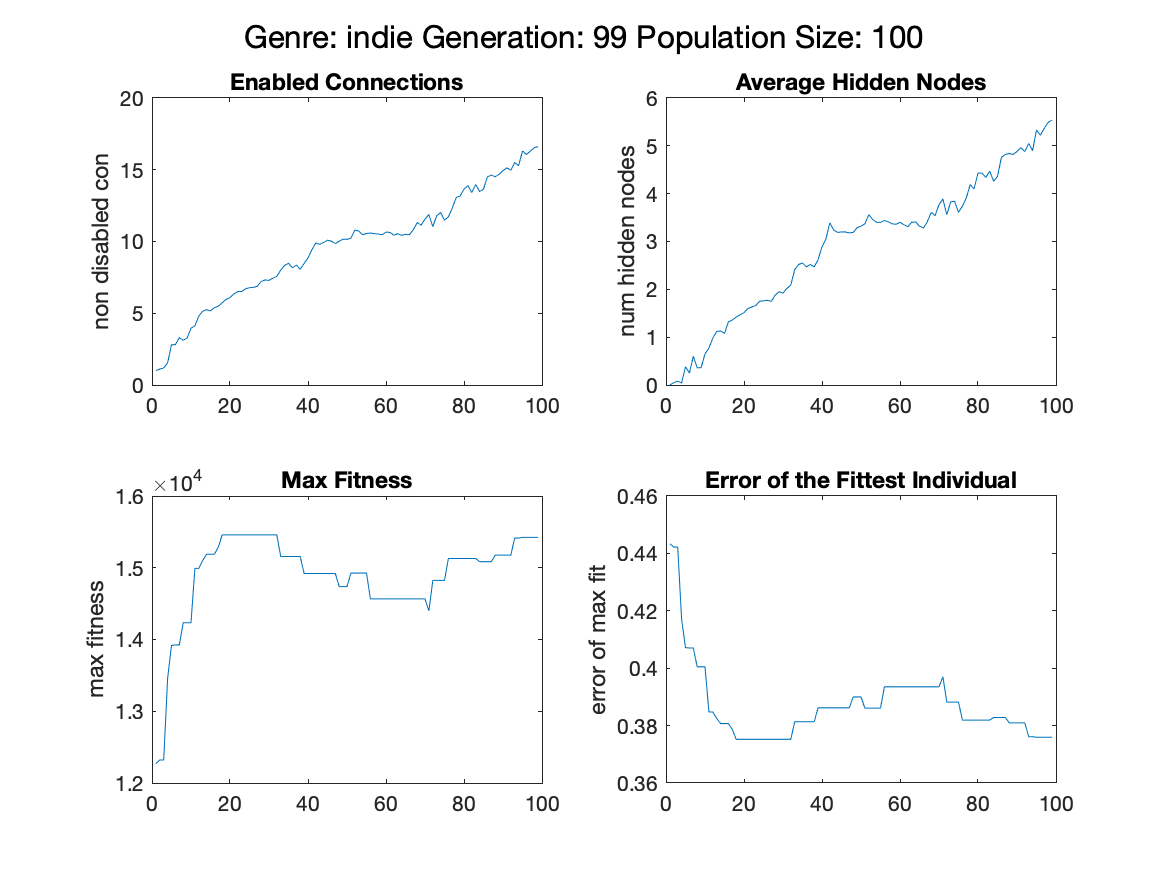
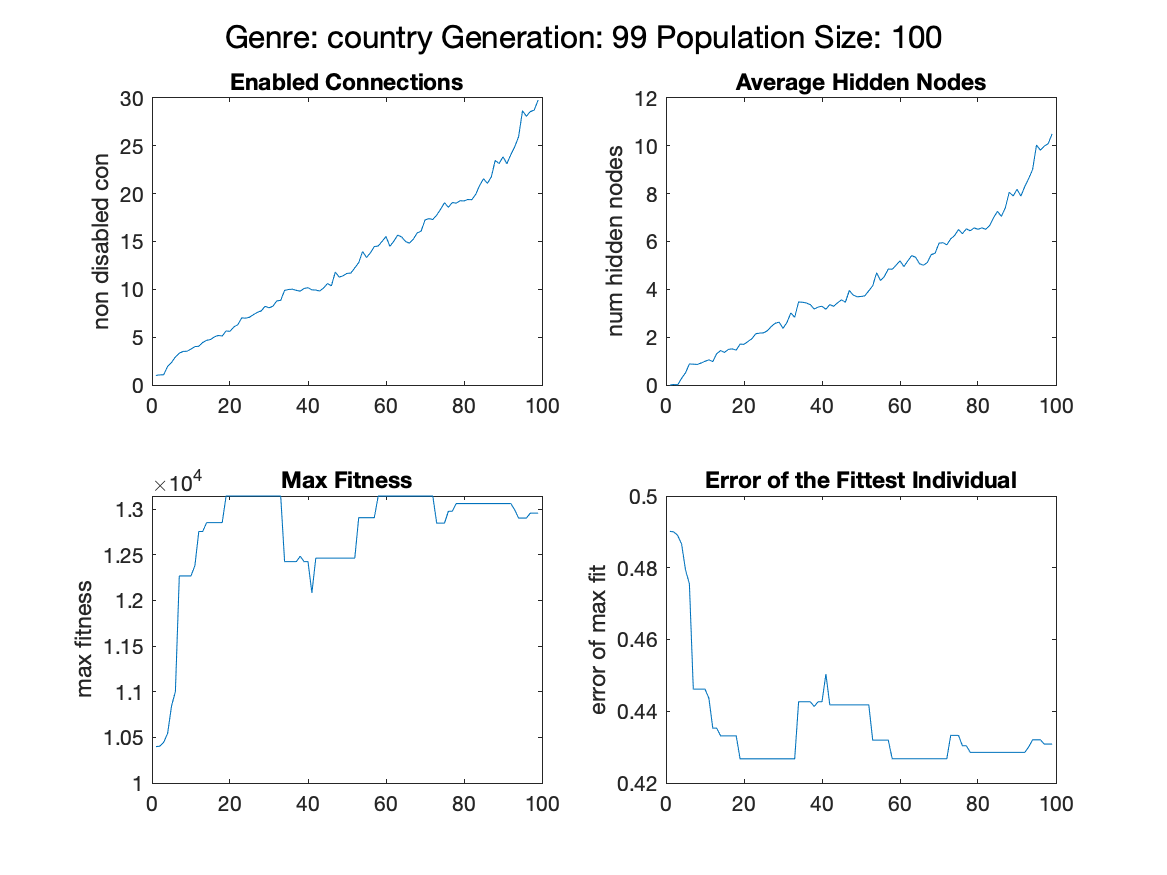


NeuroEvolution of Augmenting Topologies - Ethan

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 . 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. The motivation behind this is humans will save time in trying out different models because Neat will build itself. 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. The figure above shows the overall data flow and structure of the populations. Each genre is assigned a population which is then split to species containing individuals and each individual has its own neural net.

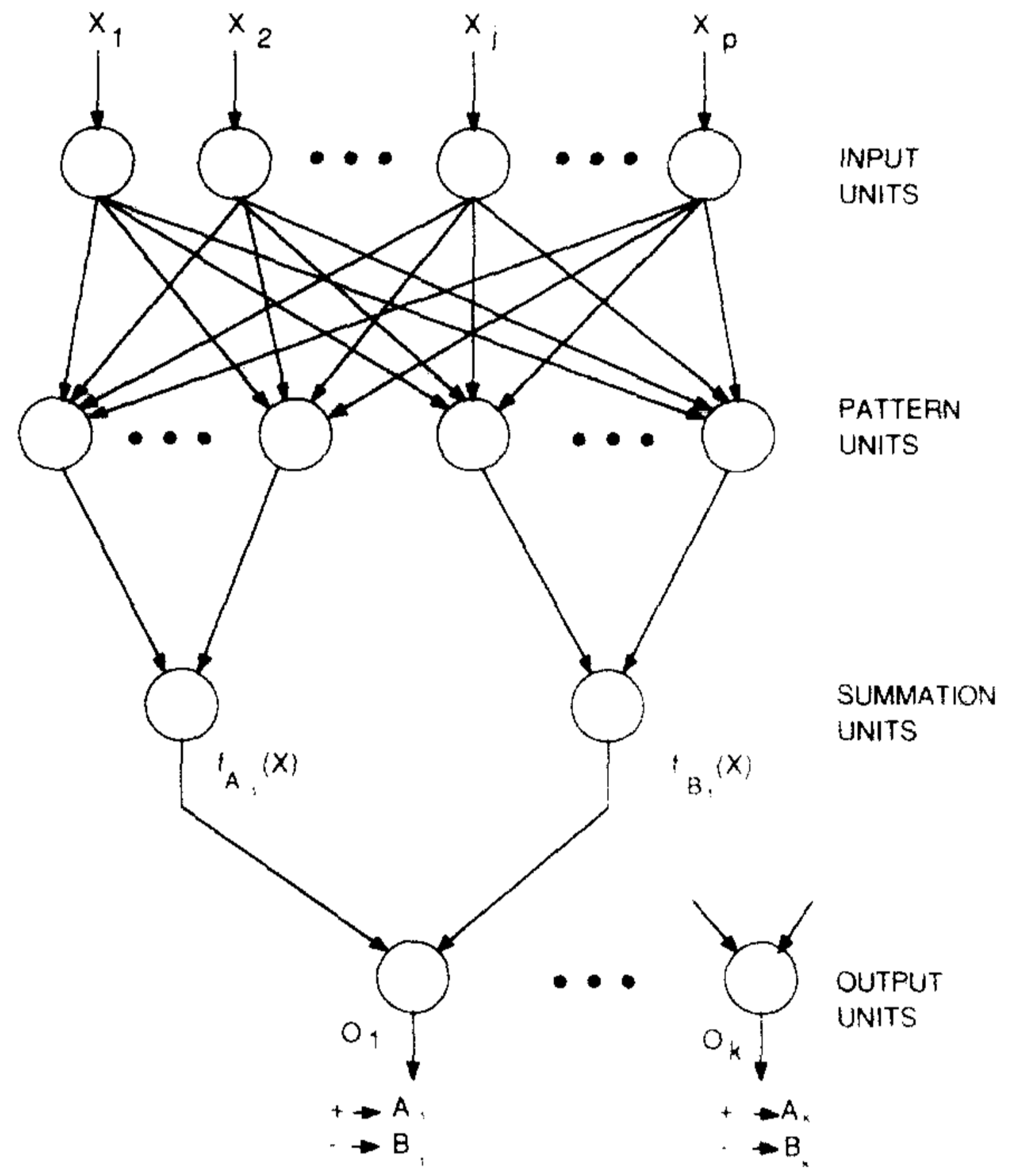
NEAT was not successful or efficient in classifying a song’s success. This was due to several reasons including: poor or no converge of error, linear increase of nodes, a large amount of input nodes, poor effect of crossover and mutation in relation to minimizing error, slow execution with the increase of speciation, etc. It can be shown in the figures below that neat did not return acceptable results for any genre. These graphs were generated on a population size of 100 and max generation of 100. These poor results were most likely cause by an incorrect/sub-optimal configuration of parameters.



Confusion matrices:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | rock | | pop | | country | | electronic | | indie | |
|  | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 1 (success) | 65 | 38 | 0 | 113 | 22 | 76 | 66 | 36 | 23 | 65 |
| 0 (failure) | 40 | 57 | 1 | 86 | 10 | 92 | 56 | 41 | 9 | 102 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **rock**  **Recall:**  -Success: 6.190476e-01  -Failure: 6.000000e-01  **Precision:**  -Success: 6.310680e-01  -Failure: 5.876289e-01 | **pop**  **Recall:**  -Success: 0  -Failure: 4.321608e-01  **Precision:**  -Success: 0  -Failure: 9.885057e-01 | **country**  **Recall:**  -Success: 6.875000e-01  -Failure: 5.476190e-01  **Precision:**  -Success: 2.244898e-01  -Failure: 9.019608e-01 | **electronic**  **Recall:**  -Success: 5.409836e-01  -Failure: 5.324675e-01  **Precision:**  -Success: 6.470588e-01  -Failure: 4.226804e-01 | **indie**  **Recall:**  -Success: 7.187500e-01  -Failure: 6.107784e-01  **Precision:**  -Success: 2.613636e-01  -Failure: 9.189189e-01 |

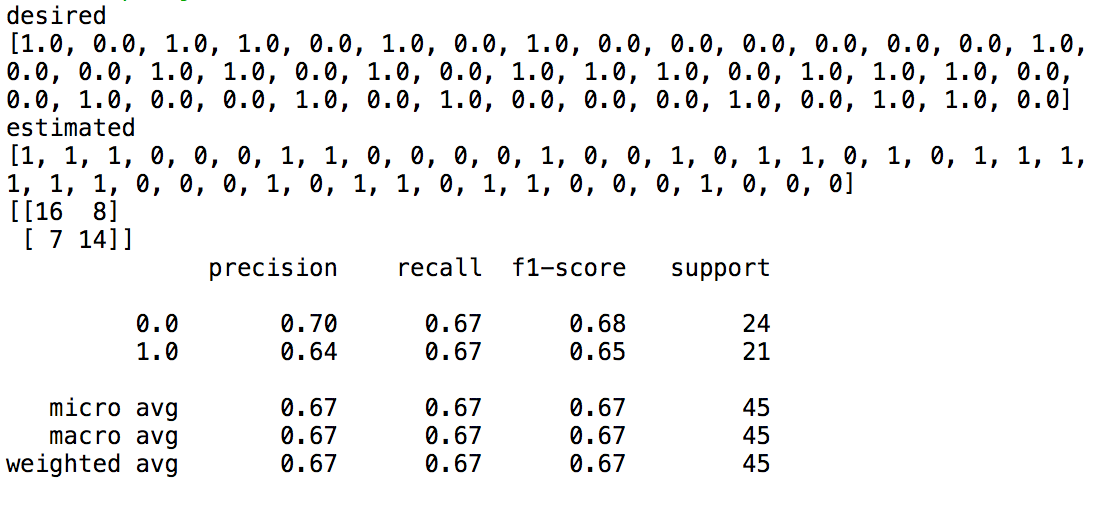
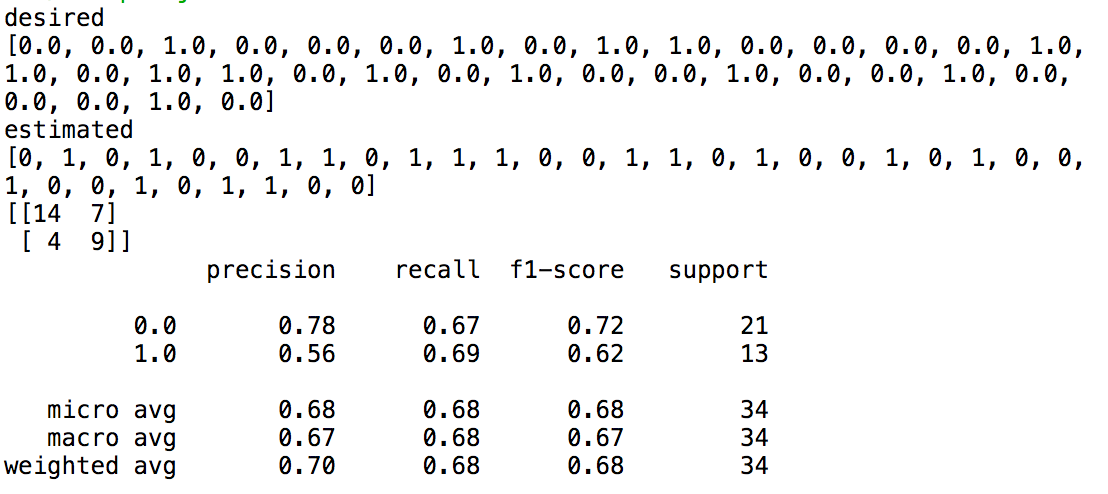
Probabilistic Neural Network - Meggie

Probabilistic Neural Networks (PNNs) are commonly used for classification problems. Rather than train a neural neural network based on vector distances, a PNN probability peaks using training data. This network is non-iterative and supervised as it simply plots each input pattern’s peak and groups them based on associated class. The probability peak are generated in the second layer where there is a node for each pattern or input and is aptly called the pattern layer as seen in the adjacent figure. At each node in the pattern layer the probability peak is calculated using a Gaussian function where 𝜎 is the smoothing factor of the peaks. The smaller the smoothing factor the steeper each peak will be. Based on the class of the training data point, the resulting probability is summated into that class’s entire probability peak. This occurs in the the third layer known as the summation layer. Once fully plotted, test data can be run through the network and plotted. At the location which the input pattern is mapped, the height of competing classes are compared. Whichever, has a greater height, or higher probability will be the class in which the data point is classified and thus the winning class is passed to the output layer.

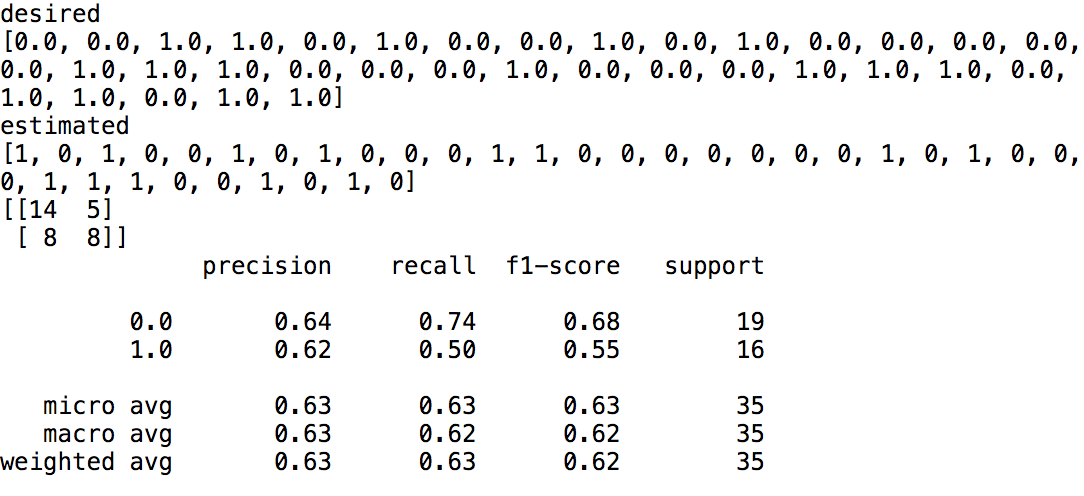
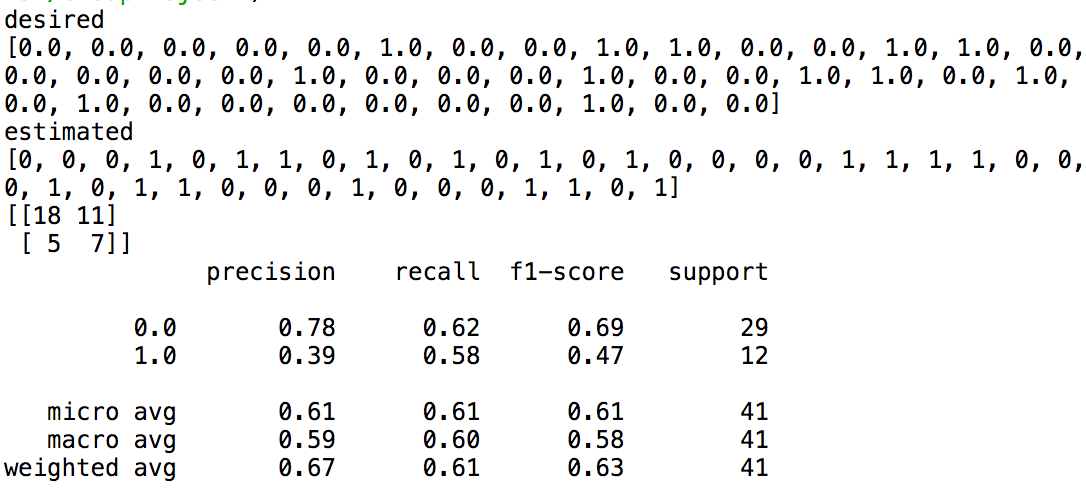
This architecture allows for great efficiency as there is no updating during training time. In the context of classifying successful and unsuccessful songs, efficiency is a high priority as songs, even within a genre, there are millions of songs and thus distinct patterns that can be used to train. Having a large a dataset as possible is favourable with such an artististic medium because so much variation exists within successful songs of one genre and so there are many patterns to cover. This efficiency will become even more advantageous as research moves towards raw songs, rather than extracted features, are fed into artificial neural networks to determine success. With so much, data it is likely iterative data will not be a feasible option, and so a PNN is a great choice when considering the future of song success prediction.

The approach taken with the datasets was to run one genre at a time and randomly splitting the data points into a 80/20 split for training/testing. This allowed testing using different test sets to assess consistency in prediction. The data was fed with a classifier attached for plotting and validating. The threshold of popularity for success was set at 75 or above as this split the most popular genre, pop, approximately evenly. Because the PNN architecture is rather rigid, the only variable subject to change is the smoothing factor. While testing with a range of 𝜎 values showed little effect on the results, a small 𝜎 of 1 was chosen as very many small pockets of similar patterns were expected. This is because within genres there exists songs that overall have similar sounds, but for the most part are very distinct from almost all other songs.

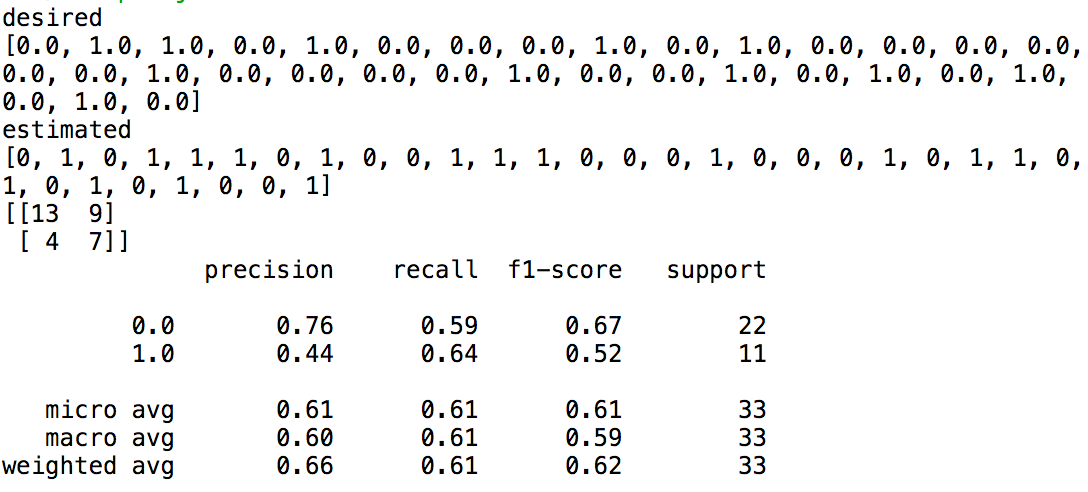
Below are results the network with each genre with randomly generated training/testing splits.

**Pop Rock**

**Country Electronic**

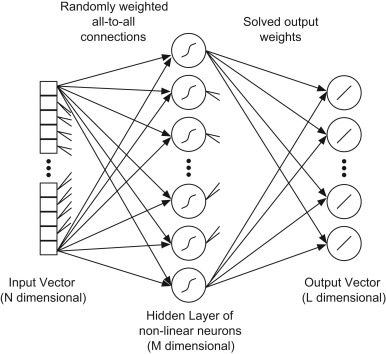


**Indie**

The above results are examples of most accurate runs of each genre. Running the network generated significantly different results each time as the test set ratio of successful to unsuccessful songs varied. Later the importance of this ratio will be discussed. The overall precision and recall of each genre on their best runs are relatively low. However, there is a clear trend of higher precision in class 0, the unsuccessful class, compared to class 1, the successful class. This distinction is likely due to the subject at hand. When listening to music, it is typically easy to tell when a song is not cohesive and thus creates a pattern is not successful. However, many songs that does everything right, or in other words has what appears to be a successful pattern, may still not succeed. In the music industry this is often considered the “it” factor because it is indescribable. From the PNN results, it is clear the network was able to identify an unsuccessful pattern, but did not have enough data to identify which songs have this “it” factor. With more feature inputs, higher precision of the successful class would be expected. Even better results would be expected from the song itself being used as the input. A PNN would be favourable with these next steps due to its non-iterative algorithm as discussed earlier. Being a “one-shot” algorithm is also what limits its ability to improve using the current dataset. 

To further analyze the results, comparison of the genres can be made. The generally more popular genres such as Pop and Rock show the highest average precisions while Indie, the least popular genre having the least successful songs, had the lowest precision. The popular genres today tend to have a formulaic approach to songwriting, where if a song has has a certain sound, it will likely succeed. This may explain why such genres are easier to classify. Contrastingly, indie is by definition unique and non-formulaic. Thus, making it harder to predict what will succeed.

Extreme Learning Machine - Serena



The Extreme Learning Machine (ELM) Neural Network randomly chooses hidden nodes and analytically determines output weights for feedforward networks with a single hidden layer (Huang et al., 2016). This type of neural network is unique from standard feed-forward networks in that it contains a large number of hidden nodes whose parameters are randomly generated independently from the training samples. The input layer requires one node per input feature, which is 12 for this implementation. Then, the ELM contains a wide hidden layer. I found that 100 hidden layer nodes worked most efficiently with the data. Finally, the output layer contains one node per output class. This implementation uses binary classification to classify a song as either being popular or unpopular, resulting in 2 output nodes. Song are classified as popular if they have a popularity ranking of over 75 on the spotify API, otherwise they are classified as unpopular.

In the ELM implementation, each genre was separated and randomly split into 80% training data and 20% testing data. The data was preprocessed using normalization, then put through ELM training. In an ELM, the weights from the input to hidden layers 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, providing the predicting output class as either 0 (unpopular song) or 1 (popular song). The accuracy and precision of the ELM is then calculated to see how effective it is at classifying the popularity of the songs.

The ELM obtained an overall accuracy of 67% (Table 1). This approach offers an extremely fast training algorithm, resulting in a very low runtime of 0.186632 seconds, as expected.

**Table 1. ELM Precision and Accuracy Over Each Genre**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Genre** | Pop | Rock | Country | Electronic | Indie | **Overall** |
| **Unpopular Precision** | 0.64 | 0.83 | 0.83 | 0.87 | 0.91 | **0.82** |
| **Popular Precision** | 0.50 | 0.75 | 0.50 | 0.20 | 0.11 | **0.41** |
| **Accuracy** | 0.60 | 0.80 | 0.70 | 0.70 | 0.55 | **0.67** |

The following are the output results from my ELM code.

**Pop**

Actual:

[1 1 1 0 0 1 1 0 0 1 1 0 1 0 0 0 0 0 0 0]

Predicted:

[0 0 0 0 0 1 0 0 0 1 1 1 0 1 1 0 0 0 0 0]

Accuracy:

0.6

Unpopular Precision (0):

0.6428571428571429

Popular Precision (1):

0.5

**Rock**

Actual:

[1 1 0 1 0 1 1 0 0 0 0 0 1 0 0 1 0 1 0 0]

Predicted:

[1 1 0 1 1 0 1 0 0 0 0 0 1 1 0 1 0 0 0 0]

Accuracy:

0.8

Unpopular Precision (0):

0.8333333333333334

Popular Precision (1):

0.75

**Country**

Actual:

[0 0 0 0 0 1 0 0 0 1 1 1 0 0 0 1 1 0 0 0]

Predicted:

[0 0 0 0 1 1 0 0 1 1 0 1 0 1 0 0 1 1 0 0]

Accuracy:

0.7

Unpopular Precision (0):

0.8333333333333334

Popular Precision (1):

0.5

**Electronic**

Actual:

[0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0]

Predicted:

[0 1 0 1 0 1 0 0 1 0 0 1 0 0 0 0 0 0 0 0]

Accuracy:

0.7

Unpopular Precision (0):

0.8666666666666667

Popular Precision (1):

0.2

**Indie**

Actual:

[0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0]

Predicted:

[0 0 0 1 1 0 1 1 0 1 1 1 0 1 0 1 0 0 0 0]

Accuracy:

0.55

Unpopular Precision (0):

0.9090909090909091

Popular Precision (1):

0.1111111111111111

**Overall Accuracy:**

0.6699999999999999

Execution Time:

0.18663200000000302 seconds

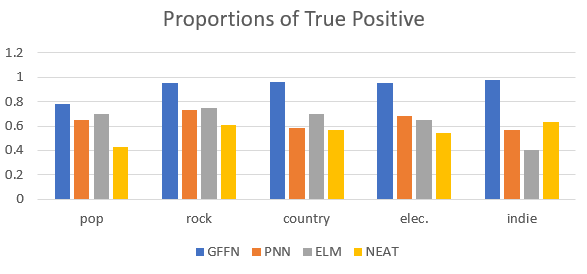
The is the best run I got overall, as the accuracy and precision values varied with each run. This lack of accuracy may be due to not using enough songs or audio features. It is also possible that my two variable parameters, the number of hidden nodes and the steepness of the sigmoidal activation function, were not optimized for my dataset.

Interestingly, certain genres performed better at popularity classification than others. For example, rock had 80% accuracy while indie had 55% accuracy (Table 1). This may be due to the fact that popular rock songs are very similar to each other, relying more on distinct features of the music. Meanwhile, the features of a popular indie song may vary, making it hard to classify using our 12 audio features.

Furthermore, the precision for classifying the unpopular songs was significantly higher than classifying popular songs (Table 1). This may be due to the fact that it is easier to identify a unsuccessful song as features would present as being very non-cohesive. In contrast, there are so many features and how they relate to each other to be considered when identifying popular songs. In order to improve my ELM neural network, more features and a larger dataset of songs should be used. There has also been a lot of research on multilayered ELMs. Perhaps adding more hidden layers will be able to better find complex patterns between features, and better classify the popularity of a song.

SUMMARY AND FUTURE WORK

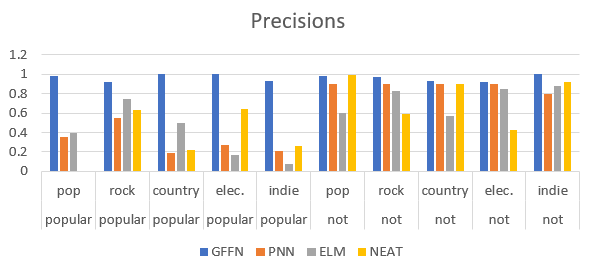
Comparison of Accuracies Between Neural Networks



**Graph 1. Comparison of accuracies for four different neural networks across each genre**

When comparing the accuracies achieved for the four different neural networks, the GFFN performed the best with an overall accuracy of 96%. There also seems to be a general trend in which certain genres are easier to classify (Graph 1). Rock popularity classification performs the best, while classifying indie songs has the least accuracy.

Common Trends Found in all Neural Networks



**Graph 2. Comparison of precision values across each class and genre for four different neural networks**

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 (Graph 2). This is consistent with the results of previous papers on the subject. 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.

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.

REFERENCES

De, S., Maity, A., Goel, V., Shitole, S., & Bhattacharya, A. (2017). Predicting the popularity of instagram posts for a lifestyle magazine using deep learning. *2017 2nd International Conference on Communication Systems, Computing and IT Applications (CSCITA)*. doi:10.1109/cscita.2017.8066548

Hoiles, W., Aprem, A., & Krishnamurthy, V. (2017). Engagement and popularity dynamics of youtube videos and sensitivity to meta-data. IEEE Transactions on Knowledge and Data Engineering, 29(7), 1426-1437.

Pham, J., Kyauk, E., & Park, E. (2016). Predicting song popularity. *Stanford,* 26.

Specht, D. F. (1990). Probabilistic neural networks. *Neural Networks,3*(1), 109-118. doi:10.1016/0893-6080(90)90049-q

Yang, L., Chou, S., Liu, J., Yang, Y., & Chen, Y. (2017). Revisiting the problem of audio-based hit song prediction using convolutional neural networks. *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. doi:10.1109/icassp.2017.7952230

Tissera, M. D., & McDonnell, M. D. (2016). Deep extreme learning machines: supervised autoencoding architecture for classification. Neurocomputing, 174, 42-49.

Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2004). Extreme learning machine: a new learning scheme of feedforward neural networks. Neural networks, 2, 985-990.

Lim, M., Lee, D., Park, H., Park, U., Kim, J. H., Park, J. S., & Jang, G. J. (2015). Audio Event Classification Using Deep Neural Networks. *Phonetics and Speech Sciences*, *7*(4), 27-33.

Lee, H., Pham, P., Largman, Y., & Ng, A. Y. (2009). Unsupervised feature learning for audio classification using convolutional deep belief networks. In *Advances in neural information processing systems* (pp. 1096-1104).

Stanley, K. O., & Miikkulainen, R. (2002). Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation,* *10*(2), 99-127.

Monterola, C., Abundo, C., Tugaff, J., & Venturina, L. E. (2009). Prediction Of Potential Hit Song And Musical Genre Using Artificial Neural Networks. *International Journal of Modern Physics C,* *20*(11), 1697-1718.

Mestyán, M., Yasseri, T., & Kertész, J. (2013). Early Prediction of Movie Box Office Success Based on Wikipedia Activity Big Data. *PLoS ONE,* *8*(8).