Training a Playlist Curator Based on User Taste

Amel Awadelkarim (ameloa)[1], Kevin Coelho (katc)[1]

[1]Stanford University

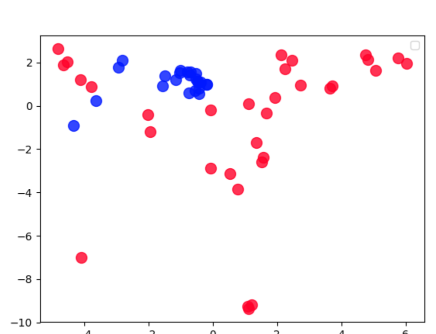
Objectives

The objective of this work is to:

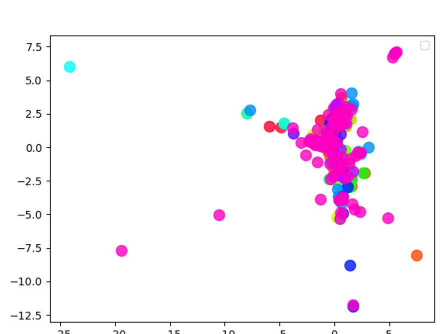
- Classify songs into given playlists
- Explore features like track metadata, artists and genres from Spotify's API
- Explore various ML algorithms in this domain.
- Train and test a model using Spotify's curated playlists and real user data

Introduction

Playlist curation is time consuming and difficult with few tools available for avid music listeners. Spotify's playlist recommendations rely on similarity (collaborative filtering), but these recommendations lack the novelty and creativity of an individual creator's taste. We hope to produce a tool trained at a smaller scale that will capture more interesting traits of each user.
Problem Statement. Given a group of K unfinished playlists, and a set of unclassified songs, S, can we sort song $s \in S$ into the best playlist $k \in K$?



(a) Jacob's playlists



(b) Myles' playlists

Figure: User playlists, mapped to two dimensions via PCA.

Users have a variety of strategies for creating playlists; many users put similar sounding songs together, some make playlists solely from one artist, others generate heterogeneous mixes spanning multiple genres, eras, and soundscapes. Our problem increases in difficulty as playlist moods become less concrete and/or separable (see above figure).

Methods

We started with data we perceived to be most separable, a "toy set" of **Spotify-curated playlists** such as "Spread the Gospel" and "Celtic Punk". We wanted to test in an ideal setting, then generalize a successful model to **real user data**. The best performing algorithms so far have been **SVM**(with RBF Kernel) and **Neural Network**(with 1 hidden layer).

Datasets

Our "toy set" composed 13 of Spotify's own pre-curated playlists (1044 tracks). For our "user set", we hand selected 116 of our friends playlists which broadly represented unique moods/styles (5721 tracks). For each track, we gathered audio features, genres, and artists from Spotify's public API. Audio features include things like "danceability", "energy", and "tempo". For categorical data like genre and artist, we started with one-hot vector representations and plan to try other methods like **node2vec** and **word2vec**.

Perceptron & SVMs

We compared Perceptron with Polynomial, Sigmoid, and RBF kerneled SVMs, with and without the preprocessing step of rescaling the features, on the toy dataset. We performed k-fold cross validation with $k = 5$ and obtained the following results:

	Scaled	Unscaled
RBF SVM	$\boxed{0.77(\pm 0.05)}$	0.25 (± 0.04)
Sig SVM	0.74 (± 0.07)	0.09 (± 0.04)
Poly SVM	0.17 (± 0.02)	0.48 (± 0.05)
Perceptron	0.76 (± 0.05)	0.13 (± 0.10)

Table: Test Accuracy: # correct/total # samples. Trained and tested with audio feature data + one-hot genres.

RBF kerneled SVM reliably performed the best with preprocessed data. Tuning the penalty parameter on the error term, we obtain an accuracy of $\boxed{0.80 \pm 0.05}$ and the following precision, recall, f-score, and support results:

	A	B	C	D
"Swagger"	0.65	0.62	0.64	76
"Spread the Gospel"	1.00	0.93	0.96	40
"'90's Baby Makers"	0.74	0.79	0.76	47
"Tender"	0.75	0.36	0.49	33
"Have a Great Day!"	0.69	0.83	0.75	100
"Dance Rising"	0.85	0.94	0.89	99
"Sad Vibe"	0.71	0.83	0.77	42
"Afternoon Acoustic"	0.79	0.80	0.80	76
"Kitchen Swagger"	0.49	0.43	0.46	75
"All The Feels"	0.85	0.88	0.86	65
"Jazz Vibes"	0.92	0.92	0.92	118
"Celtic Punk"	1.00	0.94	0.97	49
"Country by the ..."	0.95	0.84	0.89	49

Table: A: Precision, B: Recall, C: F1-score, D: Support

NN Results

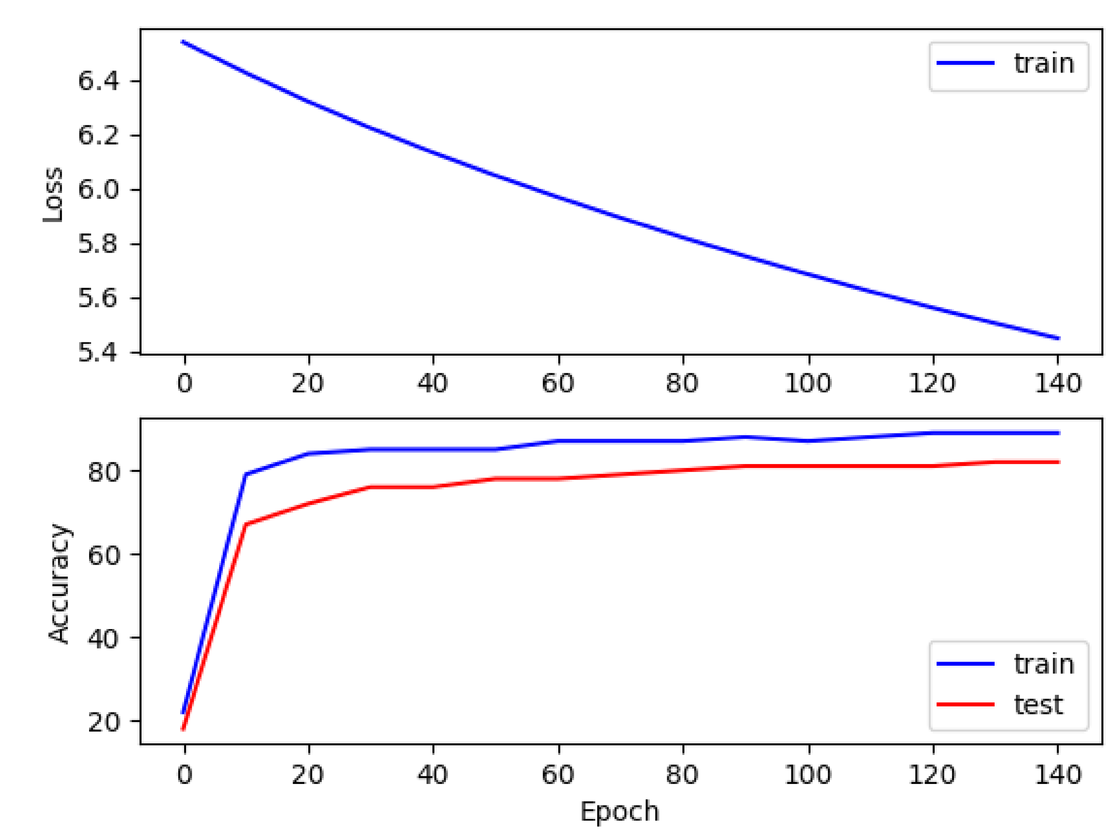


Figure: Results of training neural network with one hidden layer (sigmoid activation) and LogSoftmax output layer, minimizing NLL loss with L_2 regularization.

We explored many architectures and ultimately found that a neural network, minimizing NLL loss, with one hidden layer of neurons, and a LogSoftmax output layer performs best on the toy set.

# Hidden layers	Train	Test
2 (identity + sigmoid)	91.0%	77.0%
1 (sigmoid)	89.0%	$\boxed{82.0\%}$

Table: Train and test accuracy on the toy dataset for various architectures (activation functions in parentheses).

	Train	Test
Jacob	100%	90%
Kevin	94%	78%
Miz	62%	40%
Myles	64%	15%

Table: Train and test accuracy on various users using final architecture.

Conclusion & Future Work

- The NN achieved highest accuracy. This may be because SVMs are less suited to understanding relationships between classes
- The NN struggled on the user set. May be improved with:
  - larger labeled datasets (bigger user playlists)
  - better playlist selection (choosing more separable)
  - different algorithm, such as decision trees
  - better feature selection
- Features we hope to explore:
  - node2vec representation of Spotify's "related artists" and our own collection of related genres (each artist and genre is a node on a "related graph")
  - user-generated tags (via a companion app)

We have gathered related artists data and computed 128-dimensional node2vec artist embeddings. See below for a two-dimensional visualization of the clustering via PCA. The representation successfully captures artist similarity.

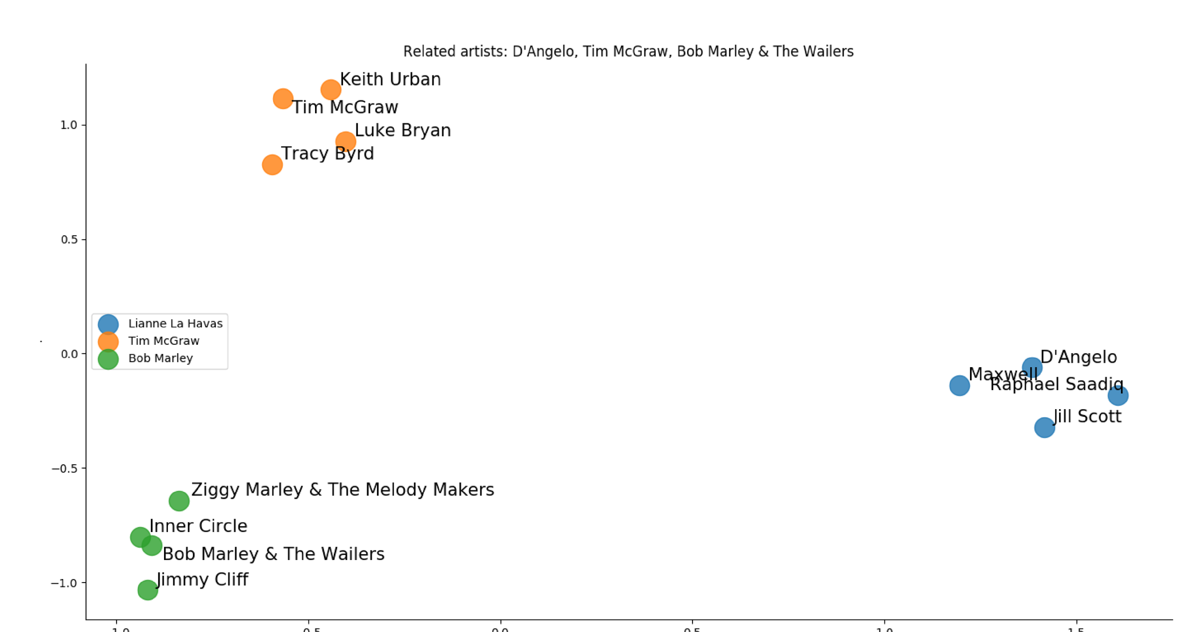


Figure: 2D node2vec data on related artists.