Autogroove: Music for Sharks

Alex Bertrand [abert13@stanford.edu] & Jordan Friedland [jordanf2@stanford.edu]

Stanford University

Introduction & Motivation

Services that provide music recommendations based on seed songs have become increasingly popular, most recently with Spotify Radio. Unfortunately, most of these services are either untenably dependent on human labelling (Pandora) or simply not of adequate quality (Pandora and Spotify). We propose to come up with a more nuanced way of generating playlists and music recommendations that have more in common than simply 'genre.'

To that end, we have constructed a convolutional autoencoder that takes as input a spectrogram of a short song clip and, in the process of reconstructing the spectrogram as the output, generates a lower-dimensional encoding of the spectrogram that can be used as input to a simple clustering algorithm. Using these encodings, we can extract relationship between songs based on features that the autoencoder network extracts during training.

Data

Our data comes from the UC Irvine Machine Learning Repository's Free Music Archive [1]. Because we chose to construct a convolutional autoencoder, we computed the spectrogram of 10-second samples of songs and cropped them into 256x256 pixel 'images' (see example below).

The spectrograms were normalized by the total energy of the song sample and then rescaled to the range [0, 1]. Because the vast majority of the pixels had a value of zero and the remaining were concentrated near zero, we preprocessed the spectrograms by multiplying them by a large number and then computing the hyperbolic tangent. While this is a highly nonlinear transformation, we found that this sped up training and gave us a better intuitive picture of how well our autoencoder was working because the transformation accentuated features in the spectrogram that were previously indistinguishable by us and by the network.

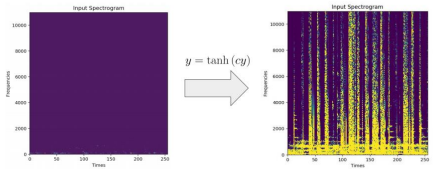

Figure 1: Preprocessing of spectrogram to enhance features.

References & Acknowledgements

[1] UC Irvine Machine Learning Repository. https://archive.ics.uci.edu/ml/index.php.
[2] V. Turchenko, E. Chalmers, and A. Luczak, "A Deep Convolutional Auto-Encoder with Pooling - Unpooling Layers in Caffe," Jan. 2017.
[3]B. J. Frey and D. Dueck, "Clustering by Passing Messages Between Data Points," Science Reports, vol. 315, no. 5814, pp. 972–976, Feb. 2007.
[4] Ng, Andrew. "Sparse Autoencoders." CS294A Lecture Notes.

Thank you to Prof. Andrew Ng, Kian Katanforoosh, our project TA Zahra Koochak, and the rest of the CS 230 teaching team for all of the course material and for a fantastic quarter of deep learning.

Model

The relative lack of data labelling music based on 'groove', 'feel,' or 'danceability' necessitates the use of unsupervised learning. For this purpose we have chosen to use a convolutional autoencoder to extract the encoding of a song in a lower-dimensional space which we hope will correlate to how that song, in general, 'feels'. From there we use clustering algorithms for music grouping.

We minimized a simple mean-squared-error loss function in order to train the autoencoder. Here, *N* is the number of training examples, *h* is the height of the 'image' in pixels, and *w* is the width of the 'image' in pixels.

$$\mathcal{L}(y, \hat{y}) = \frac{1}{Nhw} \sum_{i=1}^{N} \sum_{j=1}^{h} \sum_{k=1}^{w} \left(y^i_{jk} - \hat{y}^i_{jk} \right)^2$$

Our model architecture was relatively standard for a convolutional autoencoder [2][4], consisting of six convolutional layers comprising the encoder, followed by two fully connected layers, and then six convolutional layers comprising the decoder. We then carried out a clustering algorithm called Affinity Propagation [3], an unsupervised clustering technique which does not require a user-chosen number of clusters.

Selected Results

As a proof of concept that our autoencoder architecture was at least somewhat appropriate, we trained for 100 epochs on a training dataset of size 10 examples. From there, we proceeded to train on a dataset of 60,000 training examples for 21 epochs.

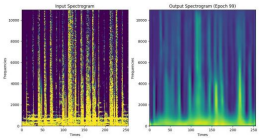

Figure 4: Autoencoder input and output after training on 10-example dataset for 100 epochs.

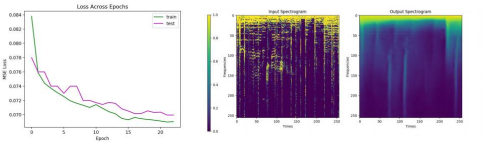

Figure 5: Results of training on 60,000 training examples for 21 epochs. (left) Training and dev loss over 21 epochs. (right) Comparison of input spectrogram and reconstructed spectrogram after 21 epochs of training on 60,000 training examples.

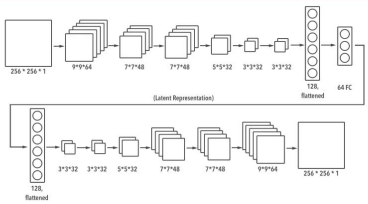

Figure 2: Convolutional autoencoder network model. The latent representation is a 64-dimensional object that encodes the 256x256 pixel input spectrogram.

$$r(i,k) \leftarrow s(i,k) - \max_{k' \neq k} \{ a(i,k') + s(i,k') \}$$

$$a(i,k) \leftarrow \min \left(0, r(k,k) + \sum_{i' \notin \{i,k\}} \max(0, r(i',k)) \right)$$

$$a(k,k) \leftarrow \sum_{i' \neq k} \max(0, r(i',k))$$

Figure 3: Governing equations for affinity propagation. The responsibility matrix *r* characterizes how well a given example serves as a cluster center for other data points. The availability matrix *a* characterizes the probability with which a data point would choose another data point as its cluster center.

Discussion

One of the challenges associated with building an autoencoder is tuning the architecture and deciding upon a value of the loss below which we can deem the autoencoder 'functional.' Training the network so that it can *fully* reconstruct any spectrogram from a significantly lower-dimensional object is very difficult and would require more training than we had the capability to do.

Additionally, evaluating the results of the affinity propagation clustering algorithm proved to be a somewhat subjective process, as the network was not trained long enough to be able to reproduce spectrograms with the accuracy desired. Thus, we could not come to very profound conclusions about the nature of the clustering.

Future Directions

Optimizing the architecture of the autoencoder was relatively difficult and we are not sure whether the architecture we have chosen is indeed appropriate. Given another six months to devote to this project, we would continue to heavily optimize the architecture of the autoencoder and then train for a much longer time on a larger dataset to generate a much more robust encoding system. In addition, we would perform more sophisticated statistical analysis of the input data to determine trends and features of spectrograms that we might want the autoencoder to pick up on.