A Machine Learning Approach to Emotion Classification

Luke Rowe
lukerowe@uvic.ca

Muhammad Ali
m.asim.a.seng@gmail.com

Ryley Woodland
wryley3845@gmail.com

1 Introduction

In an age where human and AI interaction is becoming increasingly more prevalent, the ability to accurately detect and classify human emotion is crucial. Although the problem of classifying emotion has been closely studied, few have attempted to classify emotion given only audio information. The goal of this work is to build an emotion classification system that can be used to classify the underlying emotion present in audio recordings from the recently published RAVDESS dataset. Namely, our goal is to achieve a 70% test accuracy with our best model. The justification for a seemingly low benchmark is that we believe the problem of classifying emotion is inherently subjective, since how one perceives emotion varies from person to person. The emotion labels we use are happy, sad, angry, fearful, disgusted, and surprised; these 6 emotions are commonly referred to as the 6 universal emotions. [1] Specifically, we use Linear SVM as the baseline model for the system, and we extend the Linear SVM model to include an RBF Kernel.

We hypothesize that the SVM with RBF Kernel will outperform the Linear SVM model in test performance. We believe that this is likely due to the kernel trick being able to map our feature data into an infinite dimensional space where it can find a better separating line for our data. Due to the close resemblance of audio files between different classes, we hypothesize that the data will not be linearly separable, so we will need to map our data into a higher dimensional space to find accurate separating lines. We also hypothesize that the problem of classifying emotion is a "deeper" problem than what can be modeled with these SVM classifiers. We believe this is because the SVM models will not learn the "subtle" features that can be used to accurately discern one emotion from the other.

2 Dataset

The RAVDESS dataset [2] is used to build the emotion classification system. This dataset consists of 7356 annotated audio and video samples recorded by 24 actors which are labelled as 1 of 8 emotions: neutral, calm, happy, sad, angry, fearful, disgusted or surprised. The samples have 2 levels of emotional intensity (weak or strong) and the phrases are either spoken or sung. We use only the 1888 speech and song audio samples that are labelled as one of the 6 universal emotions as described above. i.e.) we do not use the calm and neutral samples.

3 Data Preparation

3.1 Data Trimming

For each audio wav sample, we convert the wav file into a time series at a sampling rate of 16000 Hz using the Librosa library [3]. We first trim each audio sample to the same length. From inspection, we find that most of the speech clips contain about 2s of non-silent audio (from 1-3s in the audio clip), whereas most of the song clips contain about 3s of non-silent audio (from 1-4s in the audio clip), as seen in Figure 1. Originally, we trimmed each audio sample to 2.5s (0.75-3.25s. for speech, and 1.25-3.75s for song) but this led to fairly severe information loss with the song samples, as the ends of the samples were not being fully captured. As a second approach, we trim each audio sample to 3s (0.5-3.5s for the speech samples, and 1-4s for the song samples), with the trade-off that a higher proportion of each time series is "zeroed-out". With the ladder approach, we witness an improved test performance.

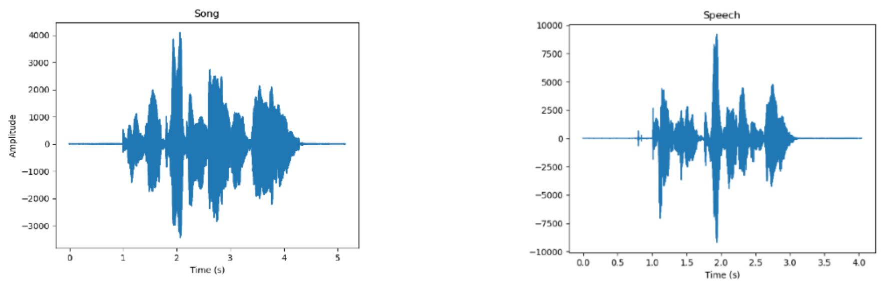


Figure 1: Time series visualization for a song and speech sample.

3.2 Feature Extraction: MFCCs

For feature extraction, the Mel-Frequency Cepstrum Coefficients (MFCCs) are extracted from each trimmed time series representation. MFCCs are a common feature representation used for audio classification purposes, and are used in other audio classification tasks, namely in [4] and [5]. The MFCCs provide a data representation that is sensitive in a way similar to how human perceive sound; they also have the added benefit of reducing the feature space from 3*16000 = 48000 to 3877 dimensions. As explained in [6], we first split the time series $s(n)$ into a series of overlapping frames $s_j(n)$ and convert each frame into the frequency domain using the Discrete Fourier Transform (DFT) over a hamming window $h(n)$:

$$S_j(k) = \sum_{n=1}^{N} s_j(n)h(n)e^{-2\pi ikn/N} \qquad k = 1,2,\dots,K$$

where $s_j(n)$ represents the j^{th} frame of the audio signal, N is the number of data samples per frame, and K is the size of the DFT. We pass the periodogram estimate of this frequency representation through a human-perceptual Mel-filterbank. We then take the logarithm of the filterbank energies, since humans hear loudness on a more logarithmic scale. Finally, the logarithm filterbank energies are passed through a Discrete Cosine Transform (DCT) to decorrelate the frequencies in each mel-filterbank, and the first 13 coefficients of the DCT are the "Mel-Frequency Cepstrum Coefficients" extracted from each frame of the sample. A heat map of the MFCC features are shown in Figure 3. We use the python_speech_features library [7] to implement the MFCCs.

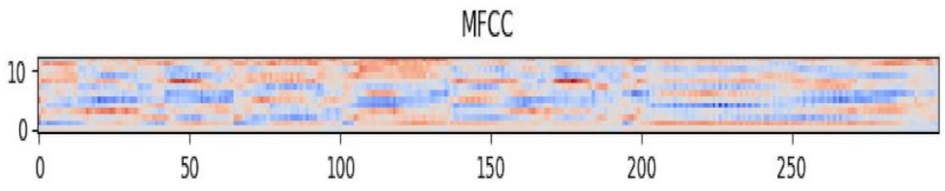


Figure 2: An MFCC heat map where cooler regions correspond to higher energies; the 299 overlapping frames signal are stacked along the x-axis, and the 13 MFCC coefficients for each frame are stacked along the y-axis.

3.3 Data Preprocessing

We create a randomized balanced train and test set with an 80/20 train/test split. Once we obtain the MFCC features for each audio sample, we flatten the MFCC features of each sample into a Dx1 feature vector. We then create a feature matrix of size N x D, where N is the number of samples, and D is the number of MFCC features for one sample. We normalize our MFCC features along each of the D dimensions. The SVM with RBF classifier includes a PCA reduction to 200 dimensions, so that the *training set size to feature dimension ratio* is more in proportion. i.e.) from about 1:2 to 10:1.

4 Methods

4.1 Linear Support Vector Machine

The baseline model is a Linear SVM classifier. We extend the binary SVM algorithm learned in class to a multiclass, one-versus-all soft-margin classifier. We also include an L2 regularization term to mitigate the effects of overfitting, with $\alpha = 0.0001$. That is, as suggested in [8], we find:

$$\min_{W,b} \frac{1}{2}\sum_{k=1}^{6} \|w_k\|^2 + \sum_{i=1}^{N} \max\{0, \max_{j \neq y_i}\left(w_j^T x_i + b_j\right) - \left(w_{y_i}^T x_i + b_{y_i}\right) + 1\}$$

where w_k are the coefficients for the decision function of class k, and x_i is the i^{th} training sample with label y_i. For one-versus-all classification, a binary classifier is learned for each of the 6 classes to discriminate between the other 5 classes. We initialize the coefficient matrix W and intercept matrix b to small uniformly random numbers in the range $[-0.001, 0.001]$. We set aside 10% of the training set for validation to ensure that our model was not overfitting; concretely, our model terminates training when the validation score has not improved by at least tol=1e-5 for 5 consecutive epochs. We used a learning rate of 1e-4, and we implemented this using scikit-learn. [9]

4.2 Support Vector Machine with Radial Basis Function Kernel (SVM-RBF)

For our next approach, we train a multiclass one-versus-one SVM classifier with a Radial Basis Function (RBF) Kernel. In a one-versus-one scheme, 6*(6-1) / 2 = 15 binary SVM-RBF classifiers are constructed. For a sample at prediction time, the class with the most positive predictions from the classifiers is the class labelling assigned to that sample. The RBF kernel projects our data onto an infinite dimensional space, where the classifier can find a more complex decision boundary between the classes. The RBF kernel represents similarity as a decaying function of the distance between two vectors x and x' as follows:

$$K(x,x') = e^{-\gamma\|x-x'\|^2}$$

where the variable γ is a hyperparameter. Empirically, we find that $\gamma = 0.0003$ and penalty parameter $C = 10$ produced the best results. This is implemented using scikit-learn [9].

5 Results and Discussion

We average our results over 10 runs of each algorithm and have listed our train and test accuracy results in Table 1 below. The confusion matrix in Figure 3 details our test results with the SVM-RBF classifier. Our results show that the MFCC extracted features, the use of the kernel trick, and the emotional intensity of our samples have a significant impact on our classification accuracies. At test time, we find that MFCC features significantly outperform the raw time series features. For the Linear SVM model we have an average test accuracy of 54.5% and we are unable to achieve a train accuracy of 100%, which suggests that the data is not linearly separable. Therefore, we use the kernel trick with an RBF Kernel to map our MFCC data onto a linearly separable space, from which we can use SVM. The SVM-RBF model show a training accuracy of 100% and an improved test accuracy of 76.2%. Furthermore, our results show that the emotional intensity of a sample have a significant impact on test accuracy; namely, with the SVM-RBF model at test time, samples labelled as having "strong" emotional intensity are correctly classified 12% more often than samples labelled as having "weak" emotional intensity. The results in Table 1 also suggest that the model is overfitting quite significantly; no matter how we tuned our model, the overfitting was unavoidable. We believe this is due to the model not learning the deeper and more subtle features that can be used to better discern one emotion from the other.

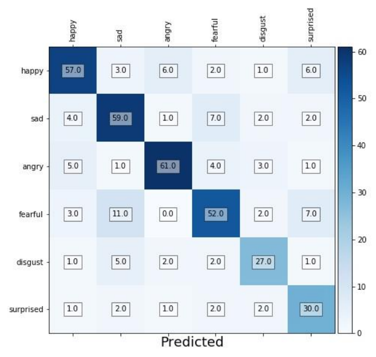
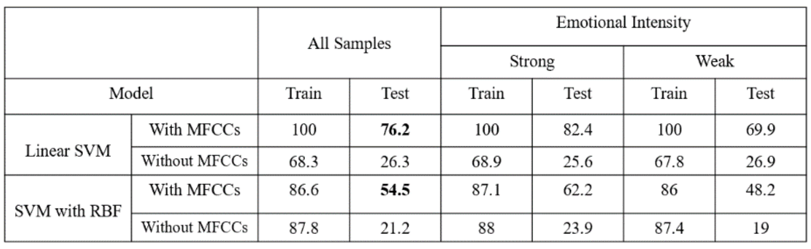


Figure 3: Confusion matrix of test results for SVM with RBF Model



		All Samples		Emotional Intensity			
				Strong		Weak	
Model		Train	Test	Train	Test	Train	Test
Linear SVM	With MFCCs	100	**76.2**	100	82.4	100	69.9
	Without MFCCs	68.3	26.3	68.9	25.6	67.8	26.9
SVM with RBF	With MFCCs	86.6	**54.5**	87.1	62.2	86	48.2
	Without MFCCs	87.8	21.2	88	23.9	87.4	19

Table 1: Train/ Test Accuracy Results

6 Conclusions and Future Work

Our hypothesis that the SVM with RBF Kernel model would outperform the Linear SVM model is confirmed by our results in Table 1. We well-exceeded our initial goal to achieve a 70% test accuracy with our best model, in that with our SVM-RBF model, we were able to achieve a 76.2% test performance. The subtleties between audio segments with different underlying emotions are extremely hard to distinguish, and so we have concluded that to accurately discern one emotion from the other, it would require a "deeper" method than what can be modeled with these simple machine learning algorithms. Another note about the misclassifications is that the weak-intensity samples are misclassified at a significantly higher rate than strong-intensity samples, which also indicates the need for a method to distinguish the more subtle components of emotion. With more time, a Convolutional Neural Network (CNN) would be constructed to learn the subtleties that truly discern one emotion from the other, which would hopefully lead to an improved test performance.

References

[1] P. Ekman, “An argument for basic emotions,” in *Cogn Emot.* 6, 169-200. 1992.

[2] S. Livingstone, “The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS): A dynamic, multimodal set of facial and vocal expressions in North American English”, *journals.plos.org,* May 16, 2018. [Online] Available: https://doi.org/10.1371/journal.pone.0196391. [Accessed: Mar. 9, 2019]

[3] B. McFee , C. Raffel, D. Liang, D. Ellis, M. McVicar, E. Battenbergk, O. Nieto, “librosa: Audio and Music Signal Analysis in Python”, in *PROC. OF THE 14th PYTHON IN SCIENCE CONF.*, 2015.

[4] G. Tzanetakis and P. Cook, “Musical Genre Classification of Audio Signals” in *IEEE Transactions on Speech and Audio Processing.* July 2002.

[5] B. Uzkent, B. Barkana and H. Cevikalp, “Non-speech environmental sound classification using SVMs with a new set of features”, in *International journal of innovative computing, information & control: IJICIC,* 8(5), May, 2012.

[6] Practical Cryptography, “Mel Frequency Cepstral Coefficient (MFCC) Tutorial,” *practicalcryptography.com.* [Online]. Available: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/. [Accessed: Mar. 7, 2019].

[7] J. Lyons, “Welcome to python_speech_features’s documentation!” *python-speech-features.readthedocs.io*, 2013. [Online]. Available: https://python-speech-features.readthedocs.io/en/latest/. [Accessed: Mar. 10, 2019].

[8] P. Felzenszwalh, “CS142: Machine Learning Lecture 11,” 2017. [Online] Available: http://cs.brown.edu/people/pfelzens/engn2520/CS1420_Lecture_11.pdf. [Accessed: Mar. 15, 2019].

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. “Scikit-learn: Machine learning in Python,” in *Journal of Machine learning Research,* 12:2025-2830, 2011.