Kusman, Matthew Richard

matthew.kusman@wsu.edu

Sound Emotion Recognition

Through Ml and ravdess data

Contents

[Introduction 2](#_Toc90251769)

[Real World Application 2](#_Toc90251770)

[Questions Sought to Answer 2](#_Toc90251771)

[Personal Motivation 2](#_Toc90251772)

[Challenges Faced 3](#_Toc90251773)

[Results 3](#_Toc90251774)

[The Task 4](#_Toc90251775)

[An ML Approach to Sound Emotion Recognition 4](#_Toc90251776)

[Training the Model 4](#_Toc90251777)

[Predicting 5](#_Toc90251778)

[Output 5](#_Toc90251779)

[Questions 5](#_Toc90251780)

[Challenges 5](#_Toc90251781)

[Technical Approach 7](#_Toc90251782)

[Algorithmic Approach 7](#_Toc90251783)

[Addressing Challenges 7](#_Toc90251784)

[Pseudo-Code 8](#_Toc90251785)

[Evaluation 9](#_Toc90251786)

[Dataset 9](#_Toc90251787)

[Metrics 9](#_Toc90251788)

[Results and Discussion 10](#_Toc90251789)

[What Worked and What Did Not 11](#_Toc90251790)

[Lessons Learned 12](#_Toc90251791)

[What Did I Learn? 12](#_Toc90251792)

[Improvements Going Forward 12](#_Toc90251793)

# Introduction

The topic of this project is Sound Emotion Recognition (SER), the practice of extracting features from audio to determine emotion carried behind the delivery of the audio. SER is part of a broader Emotion Recognition objective, which seeks to accomplish the same goal through several additional features such as facial expression and body language.

## Real World Application

SER was one of the first machine learning feats and is present in various industries. Emotion is still very unknown and applying emotion to a computational approach is even more complex. However, with the use of machine learning, SER can be achieved through supervised learning.

SER can be seen around the world in various applications such as:

* Human-like AI systems
* Chat bots
* Security systems
* Psychology applications
* Driver fatigue monitoring
* Remote elderly health monitoring
* Voice to text
* Virtual assistants

SER exists in several other applications around the world. While its usage is common, ML SER is still not as strong and capable as most humans. The process for accurately employing Sound Emotion Recognition needs continual improvement to be on the same level as the human brain in determining others’ emotion.

## Questions Sought to Answer

In creating this project, the main questions I sought to answer were the features of speech and how emotion could be derived from them, and the accuracy of a simple algorithm and machine learning process. In my time carrying out the project, I certainly gained insight into answers for these questions, though there is certainly more for me to learn. With more exploration, I would like to find ways to make the prediction of emotion slightly more accurate and employ it in a real-world system such as AI.

## Personal Motivation

When seeking various data mining projects, I knew instantly I wanted to explore machine learning more. Until recently, I had no experience with it and was blown away by the applications of it. Upon learning the simplicity of some ML algorithms, my interest grew: “How could answers so complex come from such a simple process”. While there are certainly more complex implementations, my focus has been on more straightforward approaches that can, albeit tediously, be done by hand.

I have also been fascinated by AI and application of ML in AI. While our lives grow more and more dependent on AI, we grow closer to developing realistic human like AI. Because humans and their though process are so complex, emotion is very difficult to understand computationally. When I began reading about SER, I wanted to tackle such a project that could be applied to human-like AI on an elementary level.

## Challenges Faced

When I first decided to take on the project, I was very confused and had no idea where to start. My first challenge was deciding what dataset to use, and how to process it. I very quickly found the Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) dataset. Then, I had to decide my algorithmic approach to extract features from the data. After achieving that, I encountered issues with the sheer size of data and non-standard formats. All these issues were resolved through algorithmic approaches and low complexity implementations.

## Results

After finishing the project and analyzing the results, I learned just how difficult it can be to determine emotion through speech. I discovered the less emotions available to choose from, the higher the accuracy. In general, my implementation achieved a ~50% accuracy rate for determining emotion given four options.

I saw varying results with different parameters to my classifier and different sizes of data.

# The Task

## An ML Approach to Sound Emotion Recognition

Using machine learning to tackle SER involved 2 main steps, as most supervised learning approaches do: Train a model and use that model to predict some data.

### Training the Model

This step is most of the process. It involves reading the data, formatting the data, and extracting features that will guide the algorithm, and using some model to fit the data.

The input data I used was the RAVDESS dataset, open to public use and containing 24.8 gigabytes of sound/video of 24 different actors speaking with various emotions. The videos were then rated by over 200 people to determine the emotion behind each. The dataset includes both speech and song, of which I focused on speech. I also used the .wav format of the audio to avoid unnecessary file size of the videos. Smartlabratory.org’s description of the data is:

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) contains 7356 files (total size: 24.8 GB). The database contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

To format the data, I used the SoundFile Python library, and read each .wav file as a soundfile object. This had its own challenges. Because the speech’s emotion is labeled in its filename, I had to use the glob library to determine the correct emotion from the filename. Then, I ran into an issue of file format: some files were stereo audio, but most were mono. I decided to disregard stereo-channel files and not include them in my model. Because there were only 6 in the entire dataset, I determined the change was negligible and therefore acceptable. Once properly formatted, I extracted the key features and stored them for use. Extracted and formatted input data looked like this:

mfcc chroma mel

-5.10892273e+02, 7.36717529e+01, -7.93800545e+00

7.76466656e+00, 5.89791632e+00, 1.25717802e+01,

-8.39814472e+00, -9.70091045e-01,-2.68479657e+00,

-8.09254742e+00, -9.16224480e+00, -2.60333943e+00,

-6.15299456e-02, -7.17633581e+00, -5.77479076e+00

The process of extracting the features was my biggest challenge. After some research, I learned that three main features can be used to describe audio: the Mel-frequency cepstral coefficients, power spectrogram chromogram, and mel-scaled spectrogram (mfcc, chroma, and mel respectively). The Python library librosa has tools capable of extracting each of these, which I then stored in a numpy array.

Once all the data if read, formatted, and extracted, I used a Multi-layer Perceptron classifier (MLPC) from sklearn.neural\_network to fit the data. I found that the MLPC to be the most appropriate as it is an easy-to-use neural network classifier that accurately classifies data. Once fit, the model can predict the labels of data given the features.

### Predicting

Because of the MLPC’s ease of use, predicting was extremely simple. The model allows you to input the features through the predict function, which returns the predicted emotion labels. I scored the predicted labels using sklearn.metrics’ accuracy\_score function.

### Output

The program outputs two main data points: the overall accuracy of the prediction, that is what percentage of emotions it successfully predicted, and the accuracy for each emotion included. I chose to output this data because I was particularly interested in which emotions were the hardest to predict. I ran the script with different model parameters: four alpha values, over four different learning rates each for a total of 12 runs of the script. The output looks like the following:

Learning Rate: adaptive

Alpha: 0.001

Emotion accuracies:

neutral 0.00%

calm 31.48%

happy 64.10%

sad 52.08%

angry 78.18%

fearful 73.91%

disgust 40.43%

surprised 34.21%

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## Questions

The questions I sought to answer were:

1. How accurately can a neural network perceptron classifier detect emotion in short speech?
2. Which learning rates more accurately/consistently learn how to detect emotion?
3. Which alpha values more accurately/consistently learn how to detect emotion?
4. Which emotions consistently score more/less accurately on the model?

I feel my approach adequately seeks answers to these questions, but also opens the door for more questions, such as WHY the answer is what I found. More questions I have now are things such as how we can more accurately detect emotion and are there additional features we should look for in speech to detect emotion.

## Challenges

As touched on earlier, I faced a few challenges in this project:

1. What data can be useful for this project
2. Finding an accurate and consistent algorithm for the project
3. Once data was found, formatting that to a standard such that it is useful
4. Extracting features from the data
5. Choosing a learning model to accurately fit and predict the data

Each challenge did indeed have a solution, most had several. I believe my approach adequately and efficiently addressed each solution.

# Technical Approach

## Algorithmic Approach

As mentioned earlier in this report, my algorithm included two main steps: train the model, and predict the test data.

Training the model involved the following steps:

1. Load the data
   1. For every file in a given directory following the format specified (done using glob.glob()), extract features from the audio file
      1. Read the file with soundfile
      2. Use librosa to extract the MFCC
      3. Use librosa to extract the chroma
      4. Use librosa to extract mel frequency
      5. Store each in a tuple, return the tuple
   2. Append the tuple returned to x and y respectively (features and labels)
2. Split the data between train and test x and y using train\_test\_split() from sci-kit.learn
3. Initialize an MLPC with specified parameters
4. Fit the MLPC using the train x and y

Then I did the predicting simply by:

1. Calling the predict() method on the model with the test x
2. Scoring the prediction with accuracy\_score(test y)
3. For each emotion, count the % of times it correctly predicted it
4. Output to file

My script included various helper and debugging tools as well, such as file size read stats, timing stats, and file read issues.

## Addressing Challenges

The challenges I needed to address were:

1. What data can be useful for this project
2. Finding an accurate and consistent algorithm for the project
3. Once data was found, formatting that to a standard such that it is useful
4. Extracting features from the data
5. Choosing a learning model to accurately fit and predict the data

I addressed them in the following manner:

1. I researched voice data, and did not find many. The most notable and sizable I discovered was the RAVDESS dataset, which I ended up exploring and using
2. Again, after research, I learned neural networks were generally the most appropriate for human areas of machine learning, such as voice recognition and emotion detection. I was familiar with sci-kit.learn, so I used the MLPC.
3. I knew of the librosa and soundfile libraries from a previous project, and found they were adequate for the project
4. This was by far the hardest challenge. I know nothing of speech features, so it involved a lot of research. I ended up finding a similar project done which noted the 3 features: the Mel-frequency cepstral coefficients, power spectrogram chromogram, and mel-scaled spectrogram. I found I could extract each using librosa.
5. As mentioned in challenge 2, I used sci-kit.learn’s MLPC, which is capable of fitting and predicting with my data and features.

## Pseudo-Code

x, y = load\_data()

data = test train split(x,y)

model = MLPC(varying parameters for each iteration)

model.fit()

predicted = model.predict(test)

accuracy = accuracy\_score(predicted, test y)

for each emotion:

emotion accuracy = emotion correct / emotion wrong

output to file

load data:

pattern = '/Actor\_\*/\*.wav'

for each file in destination with pattern:

extract features from (file)

x,y append(features, label)

return x,y

extract feature(audio file):

mfcc = mean mfcc in file

chroma = mean chroma in file

mel = mean melspectogram in file

return mfcc, chroma, mel

# Evaluation

## Dataset

As mentioned earlier in the report, I used the RAVDESS dataset for my project. It is described as:

The Ryerson Audio-Visual Database of Emotional Speech and Song (RAVDESS) contains 7356 files (total size: 24.8 GB). The database contains 24 professional actors (12 female, 12 male), vocalizing two lexically-matched statements in a neutral North American accent. Speech includes calm, happy, sad, angry, fearful, surprise, and disgust expressions, and song contains calm, happy, sad, angry, and fearful emotions. Each expression is produced at two levels of emotional intensity (normal, strong), with an additional neutral expression.

My main challenge with this data was the size. Originally I was using the full 24 gigabytes, which I then learned included the video. Since my project does not need video features, I switched to only audio. Then, I cut the song files, as my main goal was to analyze speech. This brought my dataset size down to ~500 MB, which I felt was adequate for proper analysis.

Another challenge was the inconsistency of mono and stereo audio. I found ~6 files that were stereo audio, which could not be run in my algorithm the same way as mono audio files. Since the amount was so few, I decided to remove them from analysis. My reasoning for this is as follows: Including them would mean merging the audio channels, which made the audio slightly less realistic. I figured it was more appropriate to remove the few amount of files.

## Metrics

The main metric I used was overall accuracy percentage, based off the model’s prediction. This was simple to calculate, as I was supervising the learning of the model and could simply calculate the number of mistakes in the prediction.

I am confident this is a good measure of evaluation as the main goal of the project was to determine how accurately I could predict emotion in human speech, and success/failure rate directly corresponds to that objective.

In the real world, for my application I believe this is appropriate. However, this is only because it is a low stakes objective. Higher objective learning tasks with more critical roles may not have the freedom of failure, and thus a different metric would be more appropriate for evaluating.

# Results and Discussion

The output from the project presents a few main data points: the accuracy of different alpha values and learning rates averaged out and on different emotions individually. Graph 1 displays the average accuracy of each of the three learning rates used: adaptive, constant, and invsacaling; all compared to the average of the three:

**Graph 1**

As you can see, there is a small amount of correlation between alpha value and accuracy, while invscaling slightly outperforms the other two learning rate methods with an alpha value of .01 and .1.

Breaking each piece of output down into emotions is even more interesting: some emotions are a 0% detection rate, possibly due to order of the emotions or order of the data. At the same time, some consistently land higher percentages around 80%. Graphs 2, 3, and 4 represent the emotion accuracy breakdown over different alpha values of the three learning rates used:

**Graph 2**

**Graph 3**

**Graph 4**

Each graph depicts extremely varied data, however some trends can be seen. For example, the neutral emotion consistently is not accurately read. This is perhaps because the weights of all the other emotions tend to be significantly overbearing over an absence of emotion weight. Other emotions like happy and angry tend to be more accurate, perhaps because they are seen as strong emotions, and thus easier to detect.

Overall, it appears that with an alpha value of 0.0001, accuracies were the highest, and the invscaling learning rate produced more accurate than the other options. While emotions seemed to be all over the place, a few had strong accuracies such as happy and angry, while some were harder to detect such as neutral.

## What Worked and What Did Not

The data gathered from this experiment is hard to strictly label as “Does not work” versus “Does work” as all accuracies were generally lower than expected, averaging around 49% in total. Even still, can you consider a prediction that occasionally misses full emotions or assigned emotions when none are present reliable? In my project, stakes were low and the cost of inaccurate results is effectively zero. However, in applications such as physiological areas or courtroom software, results could improperly diagnose individuals or lead them to a life in prison.

# Lessons Learned

## What Did I Learn?

After this project and analyzing the results, I have learned a great deal about human speech and the subtleties of our voice. I was shocked to see the computational representation of voice and use it as input for a neural network learner. At the same time, I learned how inaccurate results could be, even with large datasets. There is no doubt that with an even larger dataset, the results would be more accurate, though I certainly question how much.

## Improvements Going Forward

It goes to say, I would have liked to conduct this project with a larger dataset, and perhaps a more advanced understanding of neural networks to tweak even more parameters in the model and perhaps reach more accurate results.

I would also like to conduct further research into aspects of speech and find more features to analyze and use as input to the model.

Combining these two improvements could produce far more accurate results and possible allow for applications in higher stakes areas, or just everyday life.

As for more technical improvements, adjusting the epsilon, processing certain features, and amplifying their differences, and using various models would be interesting to try.

# Acknowledgments

A great thanks goes out to all resources I used during research and implementation of this project.

**Scikit-learn:** <https://scikit-learn.org/stable/>

**SMART Lab: RAVDESS:** <https://smartlaboratory.org/ravdess/>

**Data-Flair:** <https://data-flair.training/>

**SoundFile:** <https://pysoundfile.readthedocs.io/en/latest/>

**Librosa:** https://librosa.org/doc/latest/index.html