Applications of Machine Learning Models on Audio Features to Detect Emotion

Michael Villegas
Department of Electrical and Computer Engineering
University of Florida
Gainesville, USA
villegasmichael@ufl.edu

Jonathan Lau
Department of Mechanical and Aerospace Engineering
University of Florida
Gainesville, USA
JonathanLau@ufl.edu

Andrew Meyer
Department of Electrical and Computer Engineering
University of Florida
Gainesville, USA
ameyer1@ufl.edu

Vishvak Seenichamy
Department of Computer & Information Science & Engineering
University of Florida
Gainesville, USA
vishvak.seenicha@ufl.edu

***Abstract*— Studies on speech have proved to have a large demand in today's industries. Voice recognition has been demonstrated in household devices, like Amazon Alexa, and smartphone applications, like Siri. Machine Learning algorithms have shown to be at the forefront of speech recognition due to their ability to effectively analyze audio files and their intrinsic properties. Features like Mel-Frequency Cepstral Coefficients and technologies like Fast Fourier Transforms have allowed for digital signal processing techniques to show their capabilities. This paper will primarily focus on how machine learning can be used to detect emotion in audio files. This paper compares the k-Nearest Neighbor algorithm, support vector machines, and a multilayer perceptron to detect a range of emotions in audio signals.**

I. Introduction

Machine learning algorithms have proven their importance in speech recognition and are currently being used in applications such as voice commands. In particular, this paper takes interest in emotion recognition from audio files. Emotion recognition is an area that has shown growing interest [1, 2] and is being implemented in new technologies. This paper aims to use machine learning algorithms, Fourier Transforms, and Neural Networks to differentiate different audio properties and use them to classify an audio sample with an emotion.

II. Data collection

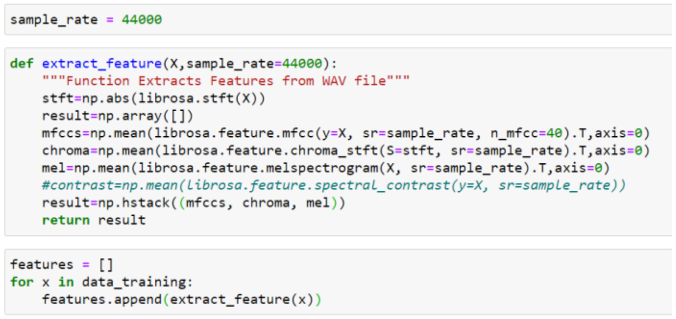
We have attained a data set from the 38 students enrolled in EEE4773. Each individual recorded two different statements, with 8 different tones. Each tone represented an emotion label. Each individual recorded 5 trials of each statement and each emotion label, giving a total of 80 recordings per student. Each recording file was 2 seconds long and recorded at 44kHz. The emotion labels are: (1) neutral, (2) calm, (3) happy, (4) sad, (5) angry, (6) fearful, (7) disgust, (8) surprise. The speech statements read were as follows: (1)" Kids are talking by the door", and (2)" Dogs are sitting by the door." These samples were later split with 70% of the samples used to train our model and the remaining 30% of the samples used to test the performance of our model.

III. Implementation

This paper will use k-Nearest Neighbor Algorithm (k-NN), Support Vector Machine (SVM), Multilayer Perceptron (MLP), and Convolutional Neural Network (CNN) to classify our data.

A. Feature Extraction

A filter was run through the audio files before model implementation. Previous studies have shown that Mel-Frequency Cepstrum and Zero-Cross Rates are able to successfully extract distinguishing features from audio files [3, 4]. These methods work by running a Fourier Transform through audio and extracting important frequencies from each audio file. After implementing and testing the Zero-Cross Rates feature, the accuracy score of the model decreased so Zero-Cross Rates was removed from the final model implementation. A Short Time Fourier Transform (STFT) was also used on the audio file due to increased accuracy score after testing and due to observations in previous studies [5]. Further, Chroma and MEL have also demonstrated success in extracting distinguishing features from audio files [6]. Chroma analyzes the audio file and categorizes various pitches for comparison. MEL feature extraction scales the frequency of the audio signal to be better processed. By using these extracted features, a model can be implemented to differentiate speech and emotion from a waveform.



```
sample_rate = 44000
```

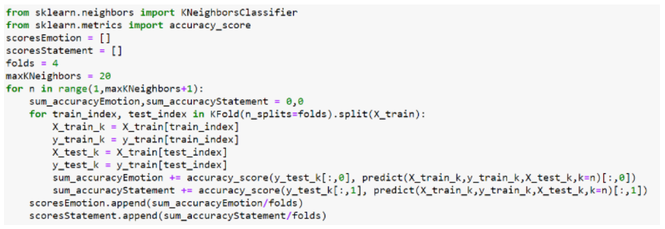
```
def extract_feature(X,sample_rate=44000):
    """Function Extracts Features from WAV file"""
    stft=np.abs(librosa.stft(X))
    result=np.array([])
    mfccs=np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T,axis=0)
    chroma=np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
    mel=np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
    #contrast=np.mean(librosa.feature.spectral_contrast(y=X, sr=sample_rate))
    result=np.hstack((mfccs, chroma, mel))
    return result
```

```
features = []
for x in data_training:
    features.append(extract_feature(x))
```

Fig. 1. Implementation of Mel-Frequency Cepstrum, Mel Spectrogram, and Chromagram [7].

B. K-NN

After filtering our audio file, k-NN was implemented on our data. The number of neighbors was iteratively tested and compared, to find the best result for data classification.



```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
scoresEmotion = []
scoresStatement = []
folds = 4
maxKNeighbors = 20
for n in range(1,maxKNeighbors+1):
    sum_accuracyEmotion,sum_accuracyStatement = 0,0
    for train_index, test_index in KFold(n_splits=folds).split(X_train):
        X_train_k = X_train[train_index]
        y_train_k = y_train[train_index]
        X_test_k = X_train[test_index]
        y_test_k = y_train[test_index]
        sum_accuracyEmotion += accuracy_score(y_test_k[:,0], predict(X_train_k,y_train_k,X_test_k,k=n)[:,0])
        sum_accuracyStatement += accuracy_score(y_test_k[:,1], predict(X_train_k,y_train_k,X_test_k,k=n)[:,1])
    scoresEmotion.append(sum_accuracyEmotion/folds)
    scoresStatement.append(sum_accuracyStatement/folds)
```

Fig. 2. Implementation of K-NN.

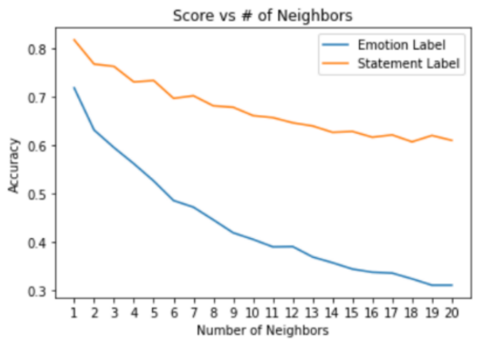
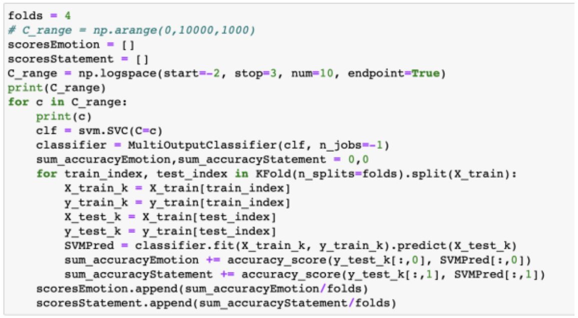


Fig. 3. Finding k for k-NN

From Fig. 3, k = 1 was found to be the best value to maximize the accuracy of the classifier.

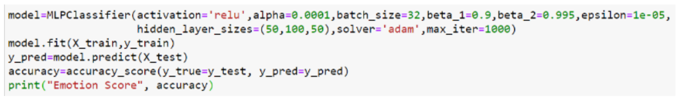
C. SVM and MLP

Support Vector Machines and Multi-layer Perceptron were used to analyze our filtered data.



```
folds = 4
# C_range = np.arange(0,10000,1000)
scoresEmotion = []
scoresStatement = []
C_range = np.logspace(start=-2, stop=3, num=10, endpoint=True)
print(C_range)
for c in C_range:
    print(c)
    clf = svm.SVC(C=c)
    classifier = MultiOutputClassifier(clf, n_jobs=-1)
    sum_accuracyEmotion,sum_accuracyStatement = 0,0
    for train_index, test_index in KFold(n_splits=folds).split(X_train):
        X_train_k = X_train[train_index]
        y_train_k = y_train[train_index]
        X_test_k = X_train[test_index]
        y_test_k = y_train[test_index]
        SVMPred = classifier.fit(X_train_k, y_train_k).predict(X_test_k)
        sum_accuracyEmotion += accuracy_score(y_test_k[:,0], SVMPred[:,0])
        sum_accuracyStatement += accuracy_score(y_test_k[:,1], SVMPred[:,1])
    scoresEmotion.append(sum_accuracyEmotion/folds)
    scoresStatement.append(sum_accuracyStatement/folds)
```

Fig. 4. Implementation of SVM.



```
model=MLPClassifier(activation='relu',alpha=0.0001,batch_size=32,beta_1=0.9,beta_2=0.995,epsilon=1e-05,
                    hidden_layer_sizes=(50,100,50),solver='adam',max_iter=1000)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Emotion Score", accuracy)
```

Fig. 5. Implementation of MLP.

The parameters for SVM and MLP were determined by iterating through various different parameters for each classification method. For MLP, GridSearchCV from scikit learn was test the various hyperparameters.

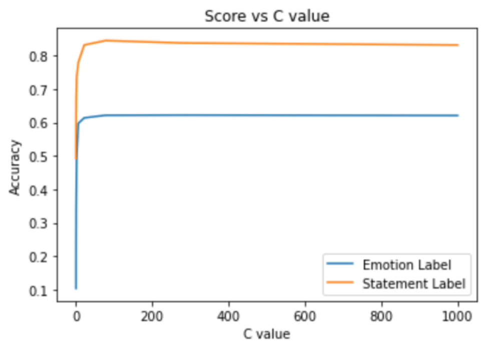
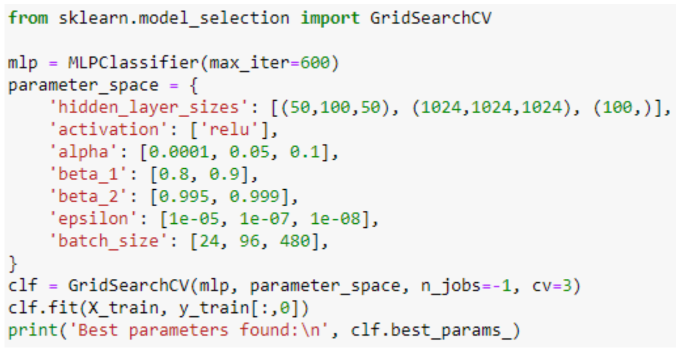


Fig. 6. Finding parameters for SVM.



```
from sklearn.model_selection import GridSearchCV

mlp = MLPClassifier(max_iter=600)
parameter_space = {
    'hidden_layer_sizes': [(50,100,50), (1024,1024,1024), (100,)],
    'activation': ['relu'],
    'alpha': [0.0001, 0.05, 0.1],
    'beta_1': [0.8, 0.9],
    'beta_2': [0.995, 0.999],
    'epsilon': [1e-05, 1e-07, 1e-08],
    'batch_size': [24, 96, 480],
}
clf = GridSearchCV(mlp, parameter_space, n_jobs=-1, cv=3)
clf.fit(X_train, y_train[:,0])
print('Best parameters found:\n', clf.best_params_)
```

Fig. 7. Finding parameters for MLP via GridSearchCV.

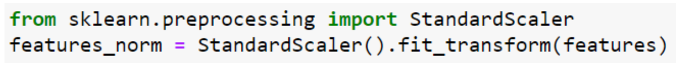
D. CNN

A convolutional neural network (CNN) was implemented to analyze the filtered data. Various parameters were tested iteratively.

IV. Experiments

A. K-NN

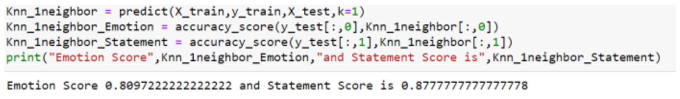
K-NN was used with both normalized and unnormalized data. We found that normalizing the data led to a higher accuracy score. Standard scaling from scikit learn was used to scale the data. The following results were obtained:



```
from sklearn.preprocessing import StandardScaler
features_norm = StandardScaler().fit_transform(features)
```

Fig. 8. An accuracy score of 81% was achieved with data normalization.

Better results were achieved for k-NN when the data was standardized. Per the scikit learn documentation, applying standard scaler to data removes the mean and scales it to unit variance. This standardization allows the individual features to approximate better like a Gaussian. This helps k-NN better classify the emotion, improving its performance. Further, the best result was obtained for k-NN when the number of neighbors was set to one, as shown previously in Fig. 3.



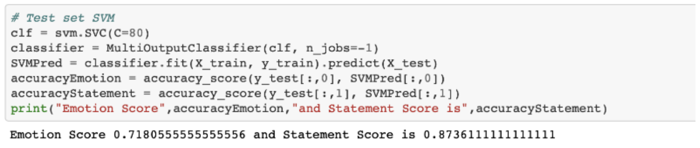
```
Knn_1neighbor = predict(X_train,y_train,X_test,k=1)
Knn_1neighbor_Emotion = accuracy_score(y_test[:,0],Knn_1neighbor[:,0])
Knn_1neighbor_Statement = accuracy_score(y_test[:,1],Knn_1neighbor[:,1])
print("Emotion Score",Knn_1neighbor_Emotion,"and Statement Score is",Knn_1neighbor_Statement)
```

```
Emotion Score 0.8097222222222222 and Statement Score is 0.8777777777777778
```

Fig. 9. K-NN received an accuracy score of 81.0%

B. SVM

The results for SVM were not as promising as those from k-NN. SVM received a proper detection rate of 71.8%, approximately 10% less than k-NN. Our k-NN classifier performed much better at predicting the statement (0 or 1) and the label than SVM. Though even at this moment, our results show promise as our predictions are almost six times better than random (12.5%).



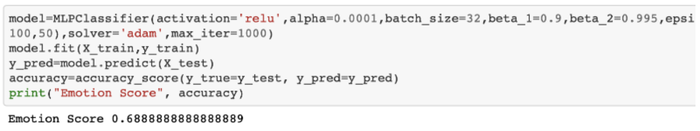
```
# Test set SVM
clf = svm.SVC(C=80)
classifier = MultiOutputClassifier(clf, n_jobs=-1)
SVMPred = classifier.fit(X_train, y_train).predict(X_test)
accuracyEmotion = accuracy_score(y_test[:,0], SVMPred[:,0])
accuracyStatement = accuracy_score(y_test[:,1], SVMPred[:,1])
print("Emotion Score",accuracyEmotion,"and Statement Score is",accuracyStatement)

Emotion Score 0.7180555555555556 and Statement Score is 0.8736111111111111
```

Fig. 10. SVM was able to predict emotion at 71.8% accuracy.

C. MLP

The Multilayer Perceptron was able to achieve an accuracy score of 68.9%, which is worse than k-NN, SVM, and CNN, but still 5.5 times better than random.



```
model=MLPClassifier(activation='relu',alpha=0.0001,batch_size=32,beta_1=0.9,beta_2=0.995,epsi
100,50),solver='adam',max_iter=1000)
model.fit(X_train,y_train)
y_pred=model.predict(X_test)
accuracy=accuracy_score(y_true=y_test, y_pred=y_pred)
print("Emotion Score", accuracy)

Emotion Score 0.6888888888888889
```

Fig. 11. MLP received an accuracy score of 68.9%.

D. CNN

The convolutional neural network was implemented using a sequential model in TensorFlow. The best results were obtained by flattening the data to shape (180,), and using three hidden layers with 1024 neurons in the first two layers, and 9 neurons in the third layer. This CNN model received a score of 74.8%, which is worse than k-NN but better than SVM and MLP.

V. Conclusion

We will be using k-NN to classify emotion. K-NN achieved an accuracy score of 80.9**%**. Our results using k-NN proved to be very promising, although being shy of 90% accuracy. Although the expressed emotions in the audio samples in this study were often difficult to distinguish.by the human ear, our algorithm was able to perform at a reliable accuracy. In the samples, the emotions calm and neutral are nearly identical and not distinguishable by the human ear. Our feature selection- Stft, Mfccs, Chroma, and Mel- allowed us to achieve a high accuracy score, but further discoveries in audio feature extraction can allow for a higher accuracy score.

Acknowledgments

We thank the UF Research Computing team for providing access to the HiPerGator supercomputer, and we thank Dr. Catia Silva for establishing the project and providing insightful discussions.

References

[1] Tarnowski, Paweł & Kołodziej, Marcin & Majkowski, Andrzej & Rak, Remigiusz. (2017). Emotion recognition using facial expressions. Procedia Computer Science. 108. 1175-1184. 10.1016/j.procs.2017.05.025.

[2] Thanapattheerakul, Thanyathorn & Mao, Katherine & Amoranto, Jacqueline & Chan, Jonathan. (2018). Emotion in a Century: A Review of Emotion Recognition. 1-8. 10.1145/3291280.3291788.

[3] Sreeram, Lalitha & Geyasruti, D. & Narayanan, Ramachandran & M, Shravani. (2015). Emotion Detection Using MFCC and Cepstrum Features. Procedia Computer Science. 70. 29-35. 10.1016/j.procs.2015.10.020.

[4] Wang, Kunxia & Li, L. & Yang, J. & Ren, Fuji. (2013). Speech emotion recognition using a novel feature set. Journal of Computational Information Systems. 9. 6097-6104. 10.12733/jcisP0241.

[5] W. Lim, D. Jang and T. Lee, "Speech emotion recognition using convolutional and Recurrent Neural Networks," 2016 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA), Jeju, 2016, pp. 1-4, doi: 10.1109/APSIPA.2016.7820699.

[6] U. Garg, S. Agarwal, S. Gupta, R. Dutt and D. Singh, "Prediction of Emotions from the Audio Speech Signals using MFCC, MEL and Chroma," 2020 12th International Conference on Computational Intelligence and Communication Networks (CICN), Bhimtal, India, 2020, pp. 87-91, doi: 10.1109/CICN49253.2020.9242635.

[7] Jay, Rocco. Audio to Emotion. https://github.com/RoccoJay/Audio_to_Emotion