Analyzing Vocal Patterns to Determine Emotion

Maisy Wieman, Andy Sun

1. Introduction

The human voice is very versatile and carries a multitude of emotions. Emotion in speech carries extra insight about human actions. Through further analysis, we can better understand the motives of people, whether they are unhappy customers or cheering fans. Humans are easily able to determine the emotion of a speaker, but the field of emotion recognition through machine learning is an open research area.

We begin our study of emotion in speech by detecting one emotion. Specifically, we investigate the classification of anger in speech samples. In our analysis of emotion, we start by delineating the data used. We transition to discussing our methodology, and through this analysis, we investigate the best algorithms to select features that are relevant to predicting emotion. We also consider multiple machine learning models with which to classify emotion.

2. Data

The Stanford University Linguistics Department has an Emotional Prosody and Speech Corpus that contains many wav files of professional actors reading dates and numbers in the following emotions: happiness, anger, sadness, despair, and neutral. Upon receiving appropriate access permissions from the Stanford Corpus TA, the data files can be accessed from an online repository. Each file stores a short audio clip of a professional actor/actress reading a neutral word or phrase in a specific emotion. We also recorded our own audio clips. These extra recordings were intended to reduce variance, but the quality of the recordings (as we are not professional voice actors) might have introduced extra error to the data set. Approximately ½ of the 690 examples exhibited anger.

3. Methodology

We want to determine whether or not an audio file contains angry emotions. At a high level, each audio file from the repository is processed and used to build a feature vector with corresponding labels that are either angry or not angry. After extracting features, we implement feature selection algorithms to determine the most relevant results, and apply various models to the features with the highest scores. To determine the testing and training error, we use k-fold cross validation. We use k=7 bins of 98

examples each, for a total of 592 training examples and 98 testing examples.

3.1 Feature Extraction

The time-domain representation of sound is very complex, and in its original form, it does not provide very good insight into key characteristics of the signal. Because of this characteristic of sound signals, we map this time domain representation into more telling features. The most straightforward technique involves determining the average energy of the signal. This metric, along with total energy in the signal, indicates the "volume" of the speaker. Duration also offers insights into emotion, as do statistics like the maximum, minimum, range, mean, and standard deviation of both the signal and spectrum. These may indicate fluctuations in the volume or pitch that can be useful in determining emotion. For both the signal and spectrum, we also derive skewness, the measure of departure of horizontal symmetry in the signal, and kurtosis, the measure of height and sharpness of central peak, relative to a standard bell curve.

We also process the signal in the frequency domain through the (Fast) Fourier Transform. We use windowed samples to get accurate representations of the frequency content of the signal at different points in time. By taking the square value of the signal at each window sample, we can derive the power spectrum. We use the values of the power spectrum as features, but we also find the frequencies that have the greatest power. We obtain the three largest frequency peaks for each window and add those to the feature vector. In addition, we find the maximum and minimum frequencies with substantial power for each time frame, and use these values to determine the frequency range for each frame. The auditory spectrum can be derived by mapping the power spectrum to an auditory frequency axis by combining the Fast Fourier Transform bins into equally spaced intervals.

The Mel-frequency Cepstrum captures characteristics of the frequency of the signal represented on the Mel scale, which closely aligns to the nonlinear nature of human hearing. By extension, the Mel-frequency Cepstrum Coefficients (MFCC) represent the "spectrum of the spectrum." MFCC's can be derived by mapping the powers of the frequency spectrum onto the mel scale, and then by taking the log of these

powers, followed by the discrete cosine transform. MFCC's are commonly used as features in many speech recognition applications.

Changes in pitch over time are measured on both a coarse time scale and a fine time scale. For coarse measurement, the signal is divided into 3 parts (beginning, middle, and end), and the part of the signal with the highest average pitch is used to determine whether the pitch rises or falls over time. For fine measurement, the dominant frequency of each windowed sample is compared to the dominant frequencies of the windowed samples immediately preceding and following. This difference is recorded in the feature vector.

3.2 Feature Selection

After we processed the original sound signal to extract features, the high variance of our algorithm revealed that we needed to filter the many features to determine which contribute most to the classifier. Our input speech signals were windowed, with approximately 72 windows per audio sample, and each of these windowed samples provided a total of 577 features. In total, we extracted 41.558 features. This large number of features (much larger than the number of examples) resulted in a very high variance. Clearly, we needed to extract the most important features. Because of the large number of features, we used heuristics to score each feature, rather than implement a brute force forward or backward search. We used three methods: mutual information, principal component analysis, binary classification tree.

3.2.1 Mutual Information: We discretized the continuous feature values into 20 bins. These bins spanned an equal range of values, but did not necessarily contain the same number of examples. After discretizing, we applied the formula for mutual information given by:

$$MI(x_i, y) = \sum_{x_i \in \{0, 19\}} \sum_{y \in \{0, 1\}} p(x_i, y) \log \frac{p(x_i, y)}{p(x_i)p(y)}$$

The results with the highest values indicate the most relevant features; we found that the most relevant features were the kurtosis and skewness (in the frequency domain), the locations of the frequency peaks, and several power spectrum values.

<u>3.2.2 PCA</u>: We used Principal Component Analysis to transform the features into a new space represented by a smaller number of dimensions. To prepare the feature matrix for PCA, we normalized it so that each

feature had zero mean and a variance of one. We used the method described by Song et al. to determine feature selection, given by the following formula:

$$score_j = \sum_{p=1}^{m} |V_{pj}|$$

Where V_{pj} represents the jth entry of the pth eigenvector of the covariance matrix of PCA. This sums the contribution of each feature to the eigenvectors corresponding to the m largest eigenvalues. The resulting score was used to determine which features were most heavily used in the final, lower dimension PCA representation, and thus gives a good indication of which features are most relevant. We also created a "weighted" PCA representation, in which we weighted each eigenvector's contribution by the corresponding eigenvalue, to reflect the relative "importance" of each eigenvector.

$$score_{j_weighted} = \sum_{p=1}^{m} \lambda_p |V_{pj}|$$

These two methods produced relatively similar results, which, surprisingly, indicated that the most relevant features were solely power spectrum and auditory spectrum values.

3.2.3 Decision Tree: We applied a third method, Binary Classification Tree, to choose the best features. Matlab's Binary Classification Tree tool box internally uses the input data to form predictive models using the optimal combination of possible binary splits. Binary splits at nodes are made based on Gini's Diversity Index (gdi), which is a measure of impurity and is given by:

$$gdi = 1 - \sum_{i} p^2(i)$$

where the sum is over the classes i at the node, and p(i) is the observed fraction of classes with class i that reach the node. A node with just one class (a pure node) has a Gini index 0; otherwise the Gini index is positive. By looking at the predictive tree model Matlab then creates, we can select the most relevant features by observing the most important nodes and slits used in the predictive model. The most relevant features were mostly power spectrum and auditory spectrum values; additionally, some weight was placed on both the coarse and fine frequency changes, and the skewness of the signal.

From the aforementioned feature selection heuristics, we have determined which features are likely most relevant to emotion detection. The results of the feature selection calculations are shown in Table 1.

Heuristic	Most Relevant Features			
Mutual	Kurtosis			
Information	Skewness (in the frequency domain)			
	Locations of the frequency peaks			
	Power spectrum values			
PCA	Power spectrum values			
unweighted	Auditory spectrum values			
PCA	Power spectrum values			
weighted	Auditory spectrum values			
Binary	Power spectrum values			
Decision	Auditory spectrum values			
Tree	Coarse and fine frequency changes			
	Skewness of the signal			

Table 1: Features with the highest scores for various feature selection heuristics

4. Models

In addition to finding the most relevant features, we also sought to find the most relevant model. Towards that end, we implemented the following classification models: Support Vector Machine, Generalized Linear Model (specifically, logistic regression), and Binary Decision Tree. For each model, we ran the simulation multiple times, with each simulation using a different number of the most relevant features (the top feature, the top two features, etc). We repeated this for each feature selection method.

Support Vector Machines: We used LIBSVM's oneclass model with Gaussian kernel to map training vectors to a higher dimensional subspace, and find the separating hyperplane with the largest margin in this new space.

Generalized Linear Model: We used a generalized linear model using a binomial distribution and the logit link function form of a GLM (logistic regression) to iteratively determine a weight with which to implement binary classification for each example. The weight is used in the sigmoidal hypothesis function.

Binary Classification Tree: We used a binary classification tree formed by taking in the most relevant features, previously determined by mutual information or PCA, and creating a predictive model based on binary splits and tree nodes that were dominated by one class.

5. Results and Discussion

Applying the features selected with the various algorithms revealed that, with the right model, anger in speech samples can be predicted with relative accuracy. The results are summarized in Table 2.

Heuristic	Error Type	SVM	GLM	Dec. Tree
Mutual	Train	15%		2%
Information	Test	28%		6%
PCA	Train	$\rightarrow 0 \%$	15 - 20%	4%
unweighted	Test	31%	35%	8%
PCA	Train	→ 0%	15 - 20%	2%
weighted	Test	32%	35%	8%

Table 2: Performance of models with various feature selection algorithms. Numbers given represent the average value as the number of features increases. The GLM error with features derived from Mutual Information has been removed from the table because the error increased with the number of features.

4.1 SVM

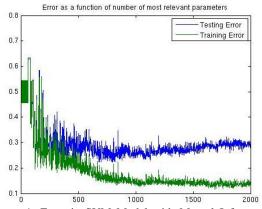


Figure 1: Error in SVM Model with Mutual Information feature selection

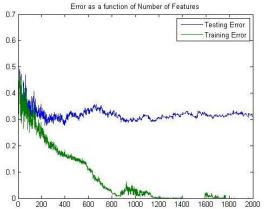


Figure 2: Error in SVM Model with unweighted PCA feature selection

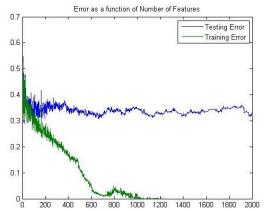


Figure 3: Error in SVM Model with weighted PCA feature selection

For the SVM model, we found that the features selected by mutual information performed better, asymptotically approaching an error of 28% (as opposed to 32% with PCA-selected features). The model exhibits overfitting, and only the first 500 features seem to contribute to the testing accuracy.

4.2 GLM

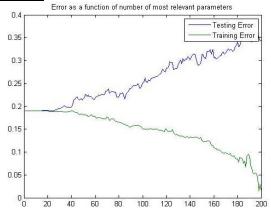


Figure 4: Error in GLM Model with Mutual Information feature selection

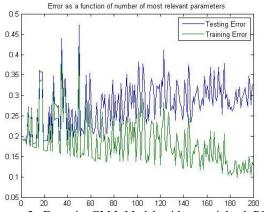


Figure 5: Error in GLM Model with unweighted PCA feature selection

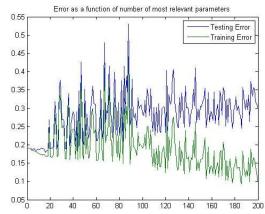


Figure 6: Error in GLM Model with weighted PCA feature selection

For the GLM model, we found that the testing error increases when using more than about 20 samples. We saw the same results when using the features selected by mutual information and the features selected by PCA. This indicates that, for the number of examples that we have, there is a high variance for even a small number of features. The performance of the features selected by PCA is much more erratic than that of Mutual Information because Matlab's GLM model reached the iteration limit for each training attempt, which in turn was caused by the **PCA** similarity in features chosen by (multicollinearity decreases convergence).

4.3 Binary Decision Tree

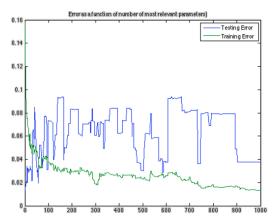


Figure 7: Error in Binary Decision Tree Model with Mutual Information feature selection

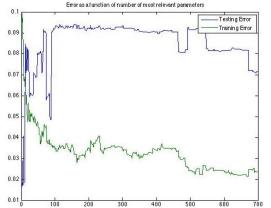


Figure 8: Error in Binary Decision Tree Model with unweighted PCA feature selection

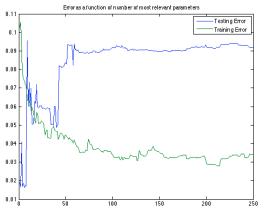


Figure 9: Error in Binary Decision Tree Model with weighted PCA feature selection

For the Binary Classification Tree model, we found that the features selected by mutual information performed better, asymptotically approaching an error of 6% (as opposed to 8% with PCA-selected features). The optimal method used by the Binary Decision Tree to split the examples at each node resulted in a much lower error than for either of the other two models.

The results for both PCA methods of feature selection were quite similar. Weighting by the eigenvalues did not significantly change which features were most relevant, nor did it improve performance. Both feature selection methods of PCA resulted in higher error than selecting features via mutual information.

For all feature selection methods and models, we encountered a high variance. This likely occurred because of the large number of features and the relatively small number of examples. This might explain why performance either plateaus or degrades after adding a certain number of features for all models.

In conclusion, the binary decision tree outperformed the other two models. Matlab's Binary Decision Tree is able to determine the most relevant features as well as predict, but to reduce time, the mutual information of the features can be calculated, and the top results of this calculation can be used by the Binary Decision Tree for prediction.

6. Future Work

In the future, we would seek to expand the number of examples by obtaining a larger set of emotionally labeled, professional voice recordings. This could decrease the variance in our results. In addition, there are many nuances of each model and feature that we could seek to optimize; one of these nuances that we hope to investigate is the performance of Mutual Information in feature selection when the features take on continuous values. In this paper, we discretized the continuous features into an arbitrary number of discrete bins. However, this could pose an interesting problem on its own: how does the number of bins affect the scores of different features, and how can we optimally divide features into discrete sets? We also hope to expand to use a multi-class model in order to distinguish between a more diverse set of emotions. Such models include multi-class svm, neural networks, and multinomial logistic regression.

7. References

- [1] Liberman, Mark, et al. Emotional Prosody Speech and Transcripts LDC2002S28. Web Download. Philadelphia: Linguistic Data Consortium, 2002.
- [2] Ellis, Daniel P. W. "PLP and RASTA (and MFCC, and Inversion) in Matlabusing Melfcc.m and Invmelfcc.m." *PLP and RASTA (and MFCC, and Inversion) in Matlab Using Melfcc.m and Invmelfcc.m.* 1 Jan. 2005. Web.
- <http://labrosa.ee.columbia.edu/matlab/rastamat/>.
- [3] Fengxi Song; Zhongwei Guo; Dayong Mei, "Feature Selection Using Principal Component Analysis," *System Science, Engineering Design and Manufacturing Informatization (ICSEM), 2010 International Conference on*, vol.1, no., pp.27,30, 12-14 Nov. 2010
- [4] Chang, Chih-Chung and Lin, Chih-Jen. LIBSVM: A library for support vector machines. . p. 27:1--27:27 2011
- [5] Ratanamahatana, C. (A. & Gunopulos, D. (2003), 'Feature Selection for the Naive Bayesian Classifier Using Decision Trees.', *Applied Artificial Intelligence* **17** (5-6), 475-487