Detecting Emotion in Human Speech

Alex Mordkovich, Kelly Veit, Daniel Zilber
{amordkov, kjveit, dzilber}@stanford.edu

December 16th, 2011

1 Introduction

Detection of emotion in speech can be applied in a variety of situations to allocate limited human resources to clients with the highest levels of distress or need, such as in automated call centers or in a nursing home. While a human may not be available to assist everyone, automated emotion detection can be used to "triage" a customer and, if they are growing angry or impatient, transfer them to speak with a human. It may also be useful for helping autistic children and adults learn to recognize more subtle social cues, or help people refine their speaking skills so that they have better control over the message they send. Other applications include improving personal assistants, such as Siri, to help convey emotion in text-based communication. The applications for emotion detection in speech are varied and practical; thus, we undertook the task of building a machine-learning system to detect emotion in utterances of human speech.

2 Features

We used the scripting capabilities of the freely-available Praat [4] software to process the audio data and extract various statistics, including a standard "Voice Report". The voice report includes statistics for pitch, pulses, voicing, jitter, and harmonicity. We use the median, mean, standard deviation, maximum, and minimum of the above characteristics as some of our features. Pulse data includes the number of pulses, number and mean of the periods, and the standard deviation of the periods. Voicing looks at unvoiced frames and calculates the number and percentage of those. Local shimmer is the average absolute difference between the amplitudes of consecutive periods, divided by the average amplitude. There are five different measurements of jitter taken, all a variant of local jitter. Local jitter is the average absolute difference between consecutive periods, divided by the average period.

A common set of features used in voice processing algorithms is the set of Mel-frequency cepstral coefficients (MFCCs). Cepstral features are those represented on a nonlinear "spectrum of a spectrum", i.e. derived by taking the Fourier transform of the logarithm of a spectrum. These features are then represented on the Mel scale, which is designed to approximate the response of human hearing by emphasizing frequencies to which humans are sensitive. Using Praat, we extract 12 commonly-used MFCCs that model the waveform, and use statistics on these coefficients as additional feature.

3 Data

For our training, cross-validation, and testing, we used the Emotional Prosody Speech and Transcripts obtained from the Linguistic Data Consortium [3]. This data consists of recordings of professional actors reciting dates and numbers with various emotional intonations. The semantic content of the utterances is intended to be emotionally neutral, as a form of psychological control in the samples. Each of the 15 audio recordings (∼200MB in size each) is annotated with a transcript file describing the time ranges of the utterances and the corresponding emotion label on each utterance. A sample transcript line looks like this:

```
1325.91 1327.94 A: elation,Seventy-one
```

This line indicates that the utterance from 96.36 seconds into the recording until 97.44 sec-

onds into the recording is an elated utterance of the phrase "two thousand six". The audio recordings and transcript data is preprocessed in a custom four-stage pipeline to generate data file in which each utterance is a data sample represented as a single line. This resulting data file is loaded into Matlab as a stylized design matrix.

90% of all the samples are used for training and cross-validation. The remaining 10% are set aside as test data that is unseen until we are ready to test our learning algorithms' hypotheses.

4 PCA

To get a sense of the training data – and to determine whether there are correlations among the features – we ran Principal Component Analysis on it. Note that since the various features are on different scales (some are percentages, some are Hertz, some are counts, etc.), from here on we work with the zscores (normalized data) rather than the raw values.

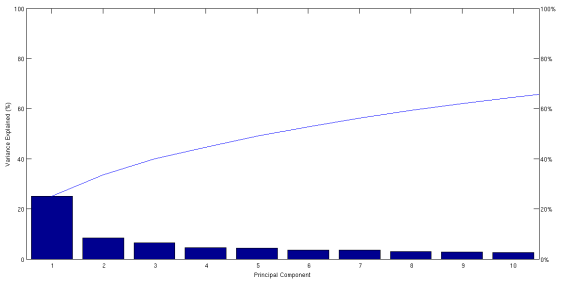

Figure 1: Explaining the Variance

In Figure 1, we see that the first two principal components explain only about 30% of the variance. In general, we see that the variance is distributed thinly across most of the features.

We also noted that the first two principal components gave the largest (absolute) weight to the features describing the median pitch and the mean of the 8th MFCC, respectively, indicating that these are potentially the more "useful" features out of the lot.

We can also project the (normalized) data onto the first two principal components. In Figure 2, we see that almost all the MFCCs are correlated positively with the second principal component, whereas some MFCC's correlate negatively and some correlate positively with the first principal component. Also we do not see any clear clustering of data points.

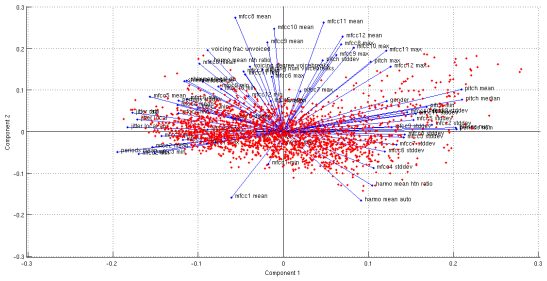

Figure 2: Principal Component Space

5 Supervised K-means

We chose to use K-Means clustering for our first classification algorithm due to its simplicity and applicability to a multilabel problem. However, because the training data set is labeled, we implemented a supervised version of K-means in which the starting centroids are calculated by grouping the data points by label and then calculating the mean for each label.

Without any further restrictions, this algorithm obtained an error of approximately 85% when attempting to classify between 14 emotions, when trained on the complete training data set. This implies that the data is not clearly separated into clusters in the given feature space. This is also a consequence of using 14 input labels in the training set. To get around this problem, additional/different features need to be carefully selected or a different metric should be used to predict emotions correctly. We explored these options to improve the K-Means predictions by implementing custom feature selection, label grouping, and fitting normal distributions akin to Gaussian descriminant analysis.

5.1 Feature Selection

Feature selection was implemented in two ways. The first way was to find the combination of 1-3 features that minimizes training error. The second approach was to use a forward or backward search heuristic. Neither approach improved results significantly. These search algorithms for new features proved to be slow. Below are some

Test Error	Train Error	Feat. 1	Feat. 2	Feat.3
86%	29%	7	11	-
93%	66%	14	19	20
97%	53%	13	18	22
92%	43%	5	7	18
88%	19%	1	2	-
90%	53%	13	18	-
89%	29%	1	2	-

anxiety	2	1	2	2	1
boredom	2	2	3	2	1
cold anger	2	1	2	3	1
contempt	3	2	2	2	1
despair	2	2	3	2	3
disgust	3	1	3	3	1
elation	1	1	3	1	1
happy	1	2	3	3	1
hot anger	1	1	3	1	1
interest	1	3	2	3	1
neutral	2	3	2	2	2
panic	3	1	3	2	2
pride	2	1	2	1	1
sadness	3	2	3	2	1
shame	2	1	2	3	1
% Error	28.6	26.2	23.8	19.1	14.3

of the results of the first approach on small sample sets containing 14 emotions. Small data sets were used to limit run time.

5.2 Label Grouping

To reduce the generalization error, the emotions were categorized into several small groups instead of 14 distinct labels, as shown in the table above. After running hundreds of randomized labelings, the best groupings were recorded with error being significantly decreased as expected, in the range of 25-30% test error. There were some trivial relabelings in which most of the emotions were grouped together, recreating the enviroment of previous research in which one or two emotions are classified against all the remaining emotions; these labelings naturally offered the lowest error at around 20 to 15%. When checking each grouping, feature selection was also applied but set to decrease test error rather than training error, making the results optmistic.

The grouping corresponding to 19.05% error was reevaluated on different samples, resulting in 23.8% and 32.5% testing error. Using two clusters yielded a 15% error, while four yielded a 40% error.

5.3 Gaussian Fitting

In addition to the label grouping and feature selection attempts to reduce error, one last approach we tried was to use gaussian distributions for each label. In theory, this would help classify some of the outliers that were closer to "incorrect" centroids when using the L2 norm. The means were calculated the same way as centroids but now a covariance matrix was calculated for the features of samples with a specified label. To avoid singular matrices, the data was normalized using zscore. The results were on par with the regular K-Means algorithm.

6 SVM

We experimented with the SVM implementations in Liblinear, LibSVM, and the MATLAB-builtin SVMClassify for our classification tasks. The linear, polynomial, and RBF kernels were used with LibSVM. All data was normalized before processing. The results of these methods are shown in the Results Matrix Tables 1 and 2. Backward feature search was used to reduce error, as speed was not a limiting factor. As seen in the result matrix, this reduced error by approximately 1% on average.

While others [6][7] have used SVM to classify emotions to distinguished between two or three emotions that known a priori, classifying a wider spectrum of emotions is a more pragmatic endeavor. The application of an algorithm that can choose only between two known emotions is quite limited. Thus, we decided to train and test our algorithms on utterances ranging the full spectrum of the 14 available emotions; the largest number of emotion labels that previous studies have worked with is only 8. The best generalization error we achieved for exact emotion classification is 43.67%.