Spoken Language Classification
CS 229 – Machine Learning

Julien De Mori   Misrab Faizullah-Khan   Cameron Holt   Shahriyar Pruisken

Autumn 2012

Abstract

We attempt to identify the spoken language of brief audio samples of speech. We employ several machine learning-based algorithms to approach this problem, using the mel-frequency cepstral coefficients of the audio signals as primary features. We discuss the results and examine the advantages and disadvantages of each algorithm.

1 Introduction

There has been much literature related to speech recognition in the machine learning community. The most common problem in this area is to translate spoken words into text. However, this approach is generally restricted to the case where the speaker's language is already known. In many situations, such as automated telephone systems, it would be useful for an algorithm to first recognize the speaker's language so that it can translate the speech from the appropriate language. As there has not been much research on this problem, we examine various machine learning algorithms to identify the spoken language from an audio clip: an SVM, neural networks, and a Gaussian mixture model. Other algorithms investigated include naïve Bayes, softmax regression, k-means, and k-nearest neighbor, however the initial results from these algorithms were not as good as those from our main models. We focused on binary classification among various pairs of languages, although our algorithms can be easily extended to sets of more than two languages.

2 Data Processing

We obtained our training and testing data from VoxForge [9], a website providing a collection of audio recordings of speech in various languages. We wrote code to scrape a large number of 48KHz 16-bit recordings in six languages: English, French, German, Dutch, Italian, and Portuguese.

Most of the clips are between four and seven seconds in length. As the computation time for our algorithms is a large concern, we strip out all but the middle 1.5 seconds from each clip. We then divide the remaining 1.5 seconds of speech into 25ms frames, each overlapping by 10ms, and multiply each frame by the Hamming window in order to smooth discontinuities in the signal. From each of these frames, we extract features known as mel-frequency cepstral coefficients.

The mel-frequency cepstrum is a representation of an audio signal on the mel scale, a nonlinear mapping of frequencies that down-samples higher frequencies to imitate the human ear's ability to process sound. In our implementations, we used the first 13 cepstral coefficients as our primary features, as is common in similar applications. A visualization of these features can be seen in Figure 1.

Besides the cepstral coefficients, we also investigated using delta features. Delta features are the first and second time derivatives of the cepstral coefficients, capturing the change of the cepstral features over time, which we hypothesize will be useful in classifying language, since pace is an

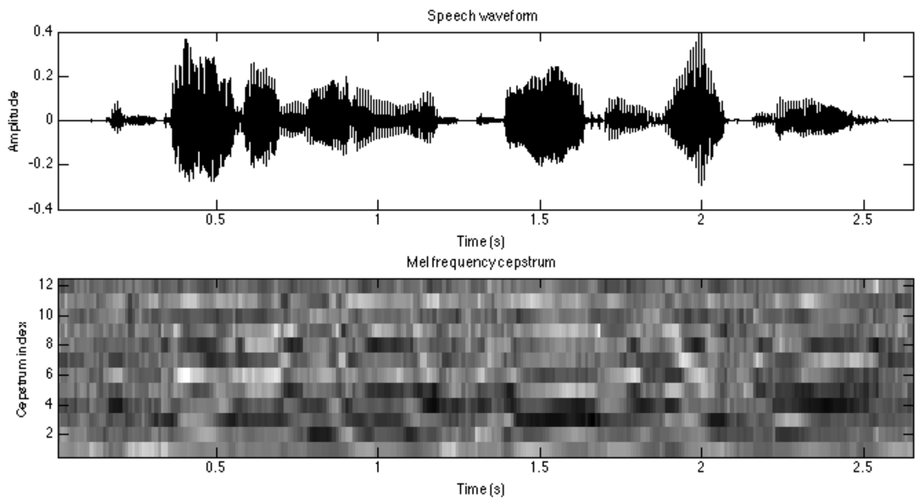

Figure 1: The original sound wave and its cepstral coefficients

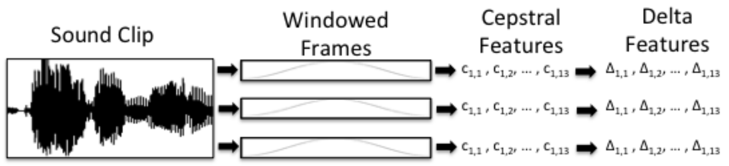

Figure 2: The feature extraction process

important factor in language recognition by humans. We can calculate these features as the central finite difference approximation of these derivatives

$$\Delta_{t,i} = \frac{c_{t+1,i} - c_{t-1,i}}{2} \qquad \text{and} \qquad \Delta^2_{t,i} = \frac{c_{t+1,i} - 2c_{t,i} + c_{t-1,i}}{4}, \tag{1}$$

where $c_{t,i}$ denotes the i-th coefficient of the t-th frame of the clip.

We also experimented using filterbank energies, defined as the energies of the signal in various regions of the cepstrum. However, adding these features to our model did not change the accuracy and significantly increased the computation time, so we did not include them in our final model.

3 Support Vector Machine

Our baseline algorithm is an SVM implementation. Training the SVM is done by first computing the cepstral and delta features for each frame. Each frame is then treated as an individual training example, and the feature values are inserted into the training matrix. The SVM is trained and tested with a linear kernel, as the computation time with nonlinear kernels was prohibitively expensive. Testing was done by making predictions on the individual frames, and then making a prediction on the clip's language by either taking a majority vote, or by multiplying the probabilities of each language across the frames. We found that both methods produced similar results.

4 Gaussian Mixture Model

We attempt to model each language as a weighted sum of M multivariate Gaussian distributions in 39 dimensions (the size of our feature space). Each language model is a probability distribution in the space of cepstral features that returns the likelihood that a set of features belongs to its label.

$M = 1$ corresponds to the case that we assume each language to be distributed as a multivariate Gaussian, and compute its mean vector $\boldsymbol{\mu}$ and diagonal covariance matrix $\boldsymbol{\Sigma}$ (making the implicit assumption of uncorrelated features). For larger values of M, we initialized the EM algorithm using centroids returned by a k-means algorithm, and computed the M most likely Gaussians to fit the training data for each language, each with an associated weight w_j that depends on the density distributions of training data. Therefore, for each $language = 1, ..., n$, where n is the number of languages modeled, we obtained a probability distribution

$$p(language|x) = \sum_{j=1}^{M} w_j p(x; \boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}) \tag{2}$$

where $p(x; \boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}) \sim \mathcal{N}(\boldsymbol{\mu_j}, \boldsymbol{\Sigma_j})$. For the testing phase, we assumed all 25ms frames were independent to simplify likelihood computations, which were as follows:

$$p(language|clip) = p(language) \prod_{f=1}^{T_c} \sum_{j=1}^{M} w_j p(x_f; \boldsymbol{\mu_j}, \boldsymbol{\Sigma_j}) \tag{3}$$

where T_c are the number of frames for each clip, $p(language) = \frac{1}{\#languages}$ is the prior on each language, and x_f is the 39 dimensional feature representation of frame f. Each clip is classified as the language that yields the highest likelihood from (3).

At a high level, we can interpret each of the M Gaussians as an accent within the language. This model then uses the EM algorithm to solve the unsupervised problem of identifying accents within a language, and then solves the supervised problem of matching an audio sample to the correct language model.

5 Neural Network

For our neural network implementation, we took a different approach to features. Instead of computing features for each 25ms frame and then making predictions on the frame level, we modeled each clip as a multivariate Gaussian distribution by computing the mean and variance of each feature across each clip, changing the dimension of the feature vector to 78. This has the effect of averaging out any noise in the signal, as well as reducing the number of training examples, considerably decreasing computation time.

We built a feedforward neural network with 3 layers of neurons; the input layer, IL, the hidden layer, HL and the output layer OL. IL has $n = 78$ neurons, one for each feature. HL has $k = 10$ neurons. OL has $p = 2$ variables, one for each testing language. Each neuron in IL and HL outputs a value to each neuron in HL and OL respectively. Every neuron i in HL has bias b_i^{HL} and weight $w_{j,i}^{HL}$ for each input j. We define s to be the sigmoid function $s(x) = \frac{1}{1+e^{-x}}$.

We first trained our model using the backward propagation method, a form of gradient descent to find the optimal $b's$ and $w's$. After running each training data point through the network we look at the difference between the actual category and the outputted category by the model, δ. We update our weights and bias at each layer using a weighted sum of the δ from the next layer. The results yielded a low accuracy. Furthermore, accuracy varied with different permutations of the training matrix.

To improve our accuracy, we instead used scaled conjugate gradient back propagation (SCGBP). This training method is not susceptible to accuracy changes with different permutations of the training matrix. SCGBP is a form of gradient descent that is developed around the idea of conjugate directions and typically converges more quickly than standard backward propagation. Using the SCGBP training method we observed far higher accuracy rates.

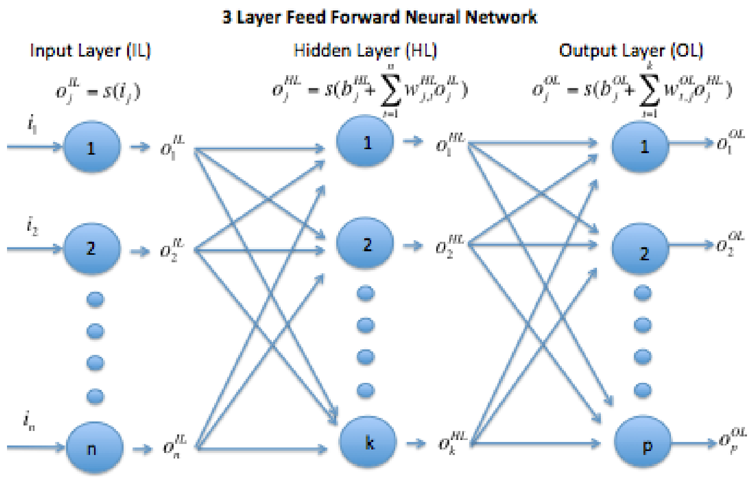
3 Layer Feed Forward Neural Network
Input Layer (IL)
Hidden Layer (HL)
Output Layer (OL)
$o_j^{IL} = s(i_j)$
$o_j^{HL} = s(b_j^{HL} + \sum_{i=1}^{n} w_{j,i}^{HL} o_j^{IL})$
$o_j^{OL} = s(b_j^{OL} + \sum_{i=1}^{k} w_{i,j}^{OL} o_j^{HL})$
1
2
n
k
p

Figure 3: Layout of neural network

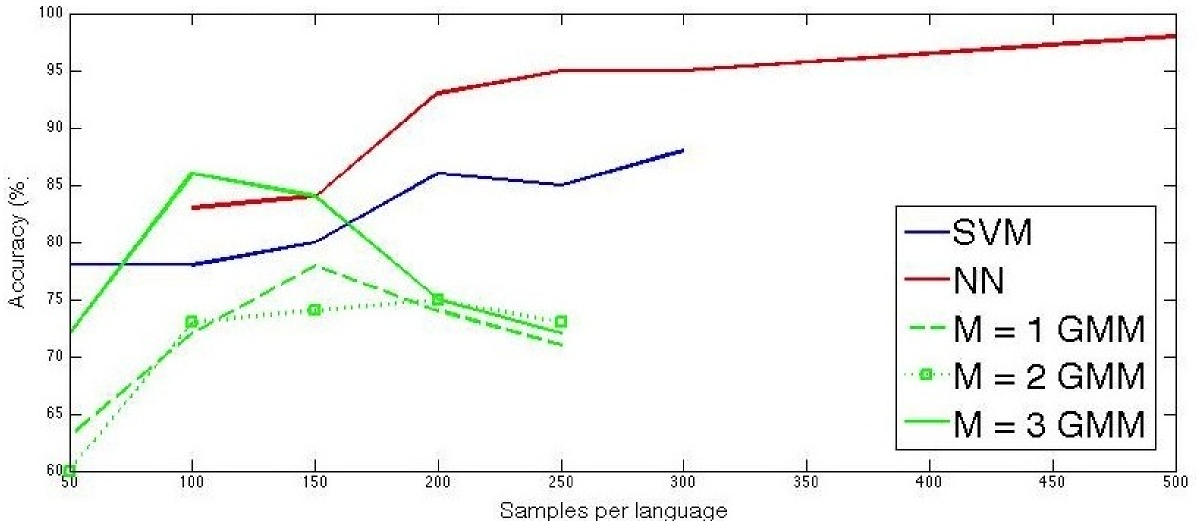

Figure 4: Testing curve for English vs. French (67% training, 33% testing)

6 Results

Overall, our algorithms performed well, with the neural network performed the best of the three. When training the English vs. French model with 1,000 samples per language, the neural network surpassed 98% accuracy. In fact, even by making a prediction based on just a single 25ms frame, the neural network achieved 85% accuracy. For both the neural network and the SVM, the accuracy increased along with the size of the training set (see Figure 4). In all of our trials, the accuracy of the Gaussian mixture model with all three M values tended to converge, leading us to believe that the M Gaussians converge to one with a large enough sample size.

We also tested the three algorithms on the German vs. Italian task, in which case both the neural network and the Gaussian mixture model had 91% accuracy, and the SVM had 83% accuracy, both with 150 samples per language. The confusion matrices for these tests can be seen in Table 1.

	German	Italian
German	**45**	12
Italian	5	**38**

Accuracy: 83%

	German	Italian
German	**49**	8
Italian	1	**42**

Accuracy: 91%

	German	Italian
German	**46**	5
Italian	4	**45**

Accuracy: 91%

Table 1: German vs. Italian testing, 100 training examples per language: SVM, NN, and 3 GMM respectively

7 Limitations and Future Considerations

Many limitations in our implementations were caused by the data set. The three main problems we observed with our data were that several of the audio clips were recorded by the same speaker, many contained background noise and many speakers were monotone, not reflecting ordinary speech. To more accurately test our algorithms, we would need a data set that better represents how the languages are actually spoken.

Despite the strong results obtained from our neural network implementation, there is still plenty of room for improvement. We did not rigorously determine the optimal number of hidden neurons to use. We would like to attempt to model the neural network with the number of neurons on the order of the dimension of the feature vector.

In our mixture of Gaussians model we modeled each language with a fixed number M of Gaussians. Future research could investigate a different value of M for each language.

In our neural network we obtained significantly higher results when inputting the mean and variance of the frames rather than inputting each individual frame. Applying this technique to our other models could result in a similar increase in accuracy, and would also significantly speed up the algorithms' runtime. To improve further computation time, we could also reimplement our models in C rather than in MATLAB, enabling us to run our algorithms on larger data sets more quickly.

Although there is room for future work and improvement, we hope this paper provides an interesting multi-pronged approach to the spoken language classification problem.

References

[1] M. Brookes. VOICEBOX: Speech Processing Toolbox for MATLAB. Available: `http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html` [Dec. 14, 2012].

[2] M. Haggblade, et al. "Music genre classification," CS 229 Final Paper, Stanford University, Dec. 2011.

[3] Madhusudanan, A. Designing and Implementing a Neural Network Library. Available: `http://www.codeproject.com/Articles/14342/Designing-And-Implementing-A-Neural-Network-Library` [Dec. 12, 2012].

[4] Mathworks. Conjugate Gradient Back Propagation Algorithm. Available: `http://matlab.izmiran.ru/help/toolbox/nnet/backpr59.html` [Dec. 12, 2012].

[5] Matlab Neural network Toolbox. Available: `www.mathworks.com/help/pdf_doc/nnet/nnet_ug.pdf` [Dec. 12, 2012].

[6] MFCC Matlab library for Mel ceptrum coefficients. Available: `http://www.mathworks.com/matlabcentral/fileexchange/23119-mfcc` [Dec. 12, 2012].

[7] G. Montavon. "Deep learning for spoken language identification," Machine Learning Group, Berlin Institute of Technology.

[8] K.K. Paliwal. "On the use of filter-bank energies as features for robust speech recognition," ISSPA '99, Brisbane, Australia, 1999.

[9] VoxForge, Open Source Speech Recognition Engines. Available: `http://www.voxforge.org/home/listen` [Dec. 12, 2012].

[10] M.A. Zissman. "Comparison of four approaches to automatic language identification of telephone speech," IEEE Transactions on Speech and Audio Processing, vol. 4, no. 1, pp. 31-44, Jan. 1996.