CS74 Milestone Report

Spoken Language Identification with Neural Networks

Jing Wei Pan, Chuanqi Sun

Introduction

Spoken language identification (LiD) is the problem of mapping continuous speech to the language it corresponds to [1]. LiD plays a significant role in the pre-processing of speech for further manipulation. Application includes the improvement for natural language processing interface such as Siri and online multilingual voice-based translator such as Google translate, both of which currently require manual selection of input language. LiD can also work with human operators. For instance, CIA agents may identify the language spoken by terrorists with LiD.

The project aims to solve LiD problem with artificial neural networks(ANN). The goal is to build and train ANN effectively and efficiently to match the performance of other known approaches to LiD, including support vector machine(SVM) and decision tree.

Methodology

Pre-processing

The preprocessing of the input audio signal is essential to the performance of ANN. Mapping the audio signal from the time domain to feature vectors both representative of the language identity and compatible with ANN requires a series of mathematical transformations (Figure 1).

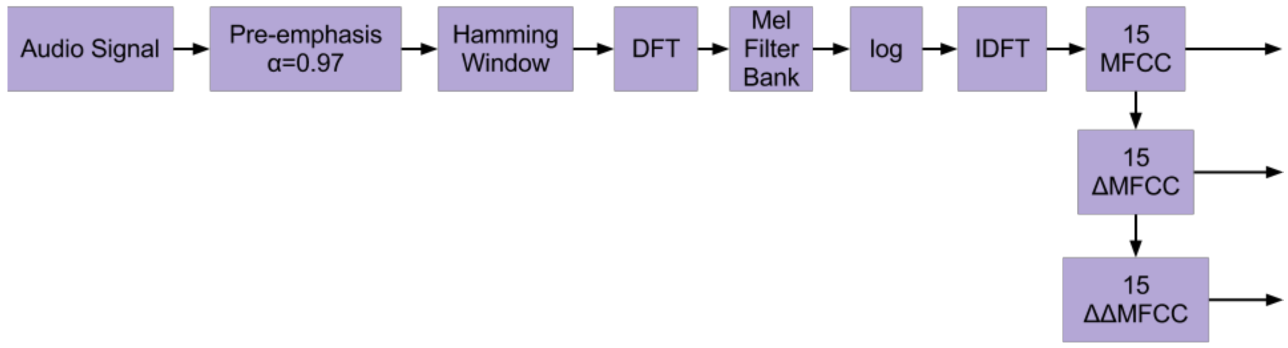


Figure 1

The audio signal is pre-emphasized with α=0.97 to improve the overall signal-to-noise ratio. Then the hamming window is applied for every 250 ms with 40% overlap so that the signal won't be cut off abruptly at frame boundaries [1]:

$$w(n) = 0.54 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right)$$

The first DFT converts the signal from time domain to frequency domain. A Mel-frequency filter bank is then applied to amplify the human hearing range:

$$Mel(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right)$$

The next log function compresses the dynamic range, and finally, Mel-frequency cepstral coefficients (MFCCs) are obtained by an inverse DFT. In the meanwhile, we compute the change and rate of change of MFCCs to capture the dynamic features of the language.

Learning Algorithm

We use per epoch backpropagation for training. The algorithm is summarized as following:
1. Randomize all the weights.
2. For each neuron, propagate forward by computing the inner products of weights and input vectors and computing the output with a tan-sigmoid transfer function.
3. Compute the error in the output layer.
4. Propagate the error backwards in a way symmetric to step 2
5. Update weights.

Features of our customized backpropagation include:

Sigmoid thresholding as the transfer function: the function introduces non-linearity to the network and limits the range from 0 to 1.

Conjugate Gradient Descent: the conjugate descent method searches for the steepest descent direction, determines the optimal distance to move along the direction, then advances the search in the conjugate direction. This method is more efficient than regular gradient descent.

Validation: A part of the training set is set aside for validation so that overfitting can be controlled.

Network Topology

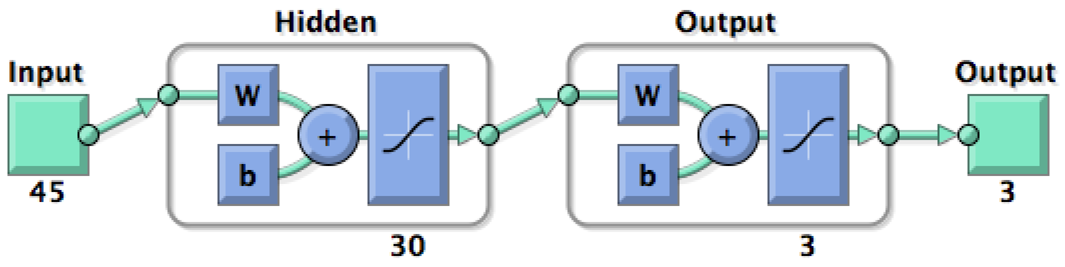


Figure 2

Our model consists of one input layer, one output layer, and one hidden layer (Figure 2). The input layer has 45 nodes, sequentially corresponding to 15 MFCCs, 15 ΔMFCCs, and 15 ΔΔMFCCs. We assign 30 units to the hidden layer by the following process: we build five networks with the same architecture, but different numbers of hidden units (5, 10, 20, 30 and 40); Each network is trained five times using the same training set, and the average error rates is collected; Network producing the lowest error rates (Figure 3) is chosen. We make no attempt to raise the number of hidden units above the number of input nodes since doing so leads to overfitting.

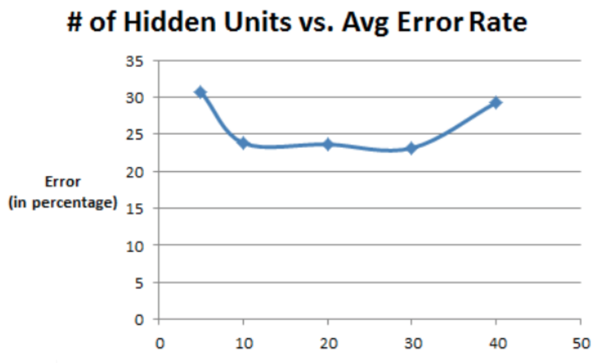
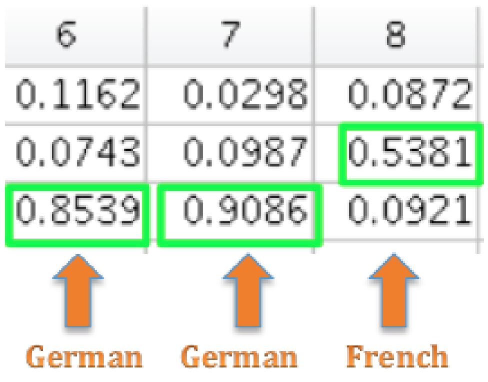


Figure 3

The output layer has three nodes, each corresponding to one language the network is trained to identify. Predictions are made based on the neurons with highest output values (Figure 4).



6	7	8
0.1162	0.0298	0.0872
0.0743	0.0987	0.5381
0.8539	0.9086	0.0921
German	German	French

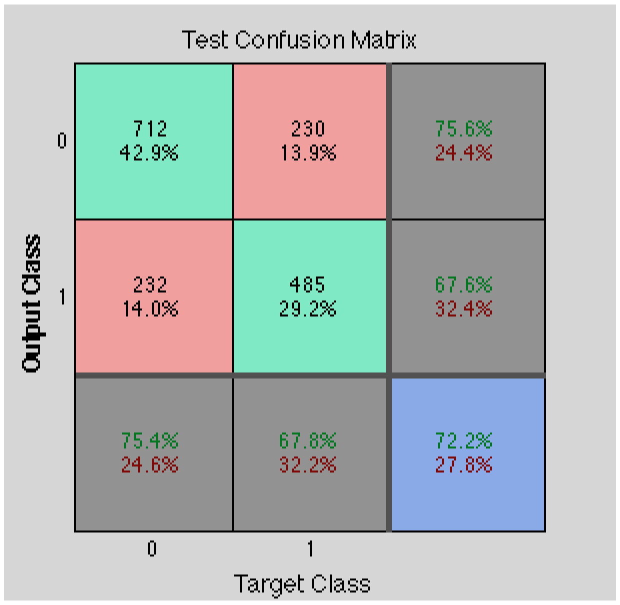
Figure 4

Results

Two Languages Comparisons

We first test our network by making predictions between two languages. Performance is measured by error rate, obtained by computing the percentage of misclassified frames. Notice that English and German both belong to the Germanic language family while French belongs to the Latin family.

The highest error rate (27.8%) occurs between English and French (Figure 5). This result is in accordance with Hosford's research in which a 25% error rate was obtained when using 13 MFCCs[1].



Test Confusion Matrix

Output Class \ Target Class	0	1	
0	712 42.9%	230 13.9%	75.6% 24.4%
1	232 14.0%	485 29.2%	67.6% 32.4%
	75.4% 24.6%	67.8% 32.2%	72.2% 27.8%

Figure 5

Our test between English and German produces a 18.9% error rate while Hosford showed an error rate of 15% when using 13 MFCCs [1]. (Figure 6)

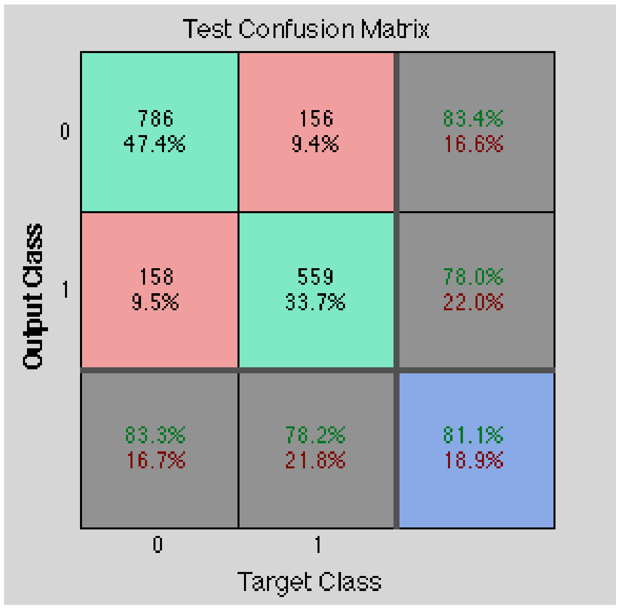


Figure 6

Finally, our network scores highest in distinguishing French and German. While we produce 11.8% error rate, Hosford showed an error rate as low as 5% using 13 MFCCs. (Figure 7) Although our error rates are generally higher, the relative performance between language pairs are in line with the Hosford's research.

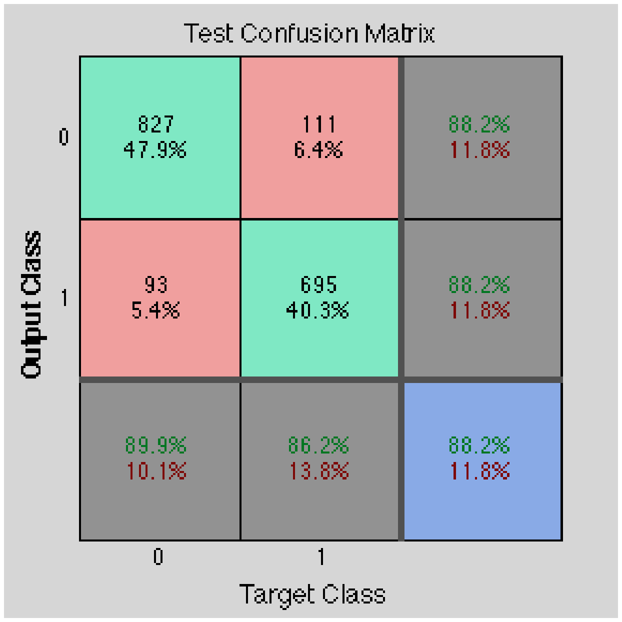
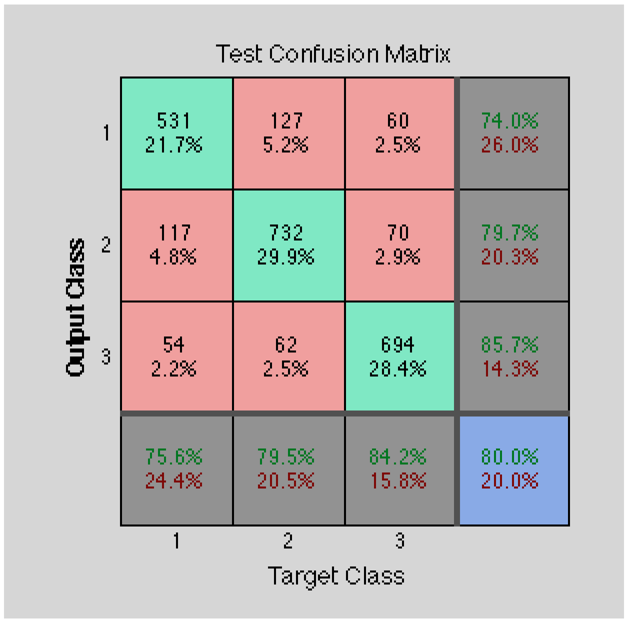


Figure 7

Three Languages Comparisons

We expect the network to make more mistakes as the number of options for prediction increases. However, after running ANN on all three languages, we only obtain a 20% error rate (Figure 8), which is lower than what is predicted given the 11.8%, 18.9%, and 27.8% error rates obtained from testing on languages pairs. The cause of this low error rate is still under investigation.



Test Confusion Matrix

Output Class \ Target Class	1	2	3	
1	531 21.7%	127 5.2%	60 2.5%	74.0% 26.0%
2	117 4.8%	732 29.9%	70 2.9%	79.7% 20.3%
3	54 2.2%	62 2.5%	694 28.4%	85.7% 14.3%
	75.6% 24.4%	79.5% 20.5%	84.2% 15.8%	80.0% 20.0%

Figure 8

File Based Three Languages Comparisons

		target class			
	Correction Rate	English	French	German	
output	English	232|21%	46|4%	6|1%	
	French	136|12%	262|24%	182|17%	
	German	13|1%	52|5%	169|15%	
		61.00%	73.00%	45.00%	60.38%

Figure 9

Finally we accumulate the per-frame predictions for each file and assign the language label to the neuron with the highest accumulated prediction. The error rates are higher than those of individual samples. (Figure 9) Since we interpret the output label in a way different from how they are used for training, the higher error rates are expected. Notice that German is mislabeled as French more often than it is correctly labeled as German. The high confusion between French and German has already been shown in the per-frame prediction between two languages.

Future Work

Comparison with SVM and DT

Our current model will be compared with support vector machine and decision tree to justify the choice of ANN as a good approach to the problem at hand.

Using all data of each file as input

Instead of training and making predictions based on each frame, we will build a deep learning network that takes all MFCCs (approx. 135000 examples) from each file. The results of the new architecture with be compared with the old one to determine a better approach.

Larger Training Set

More audio clips will be extracted from Voxforge for all three languages so that at least 600 clips can be set aside for training and validation, and 400 will remain for testing.

Adjustment of Window Size and Frame Shift

By keeping the training set constant, the window size of each sample and the frame shift of windows will be reduced to achieve better results. We expect a dramatic increase in training time.

References

[1] Hosford, Alexander W. “Automatic Language Identification (LiD) Through Machine Learning” (2011)

[2] Zissman, Marc A. “Comparison of Four Approaches to Automatic Language Identification of Telephone Speech” (1996)