Learning American English Accents Using Ensemble Learning with GMMs

Jonathan T. Purnell*
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
purnej@cs.rpi.edu

Malik Magdon-Ismail
Department of Computer Science
Rensselaer Polytechnic Institute
Troy, NY 12180
magdon@cs.rpi.edu

Abstract

Accent identification has grown over the past decade. There has been decent success when a priori knowledge about the accents is available. A typical approach entails detection of certain syllables and phonemes, which in turn requires phoneme-based models. Recently, Gaussian Mixture Models (GMMs) have been used as an unsupervised alternative to these phoneme-based models, but they have had limited success unless they used a priori knowledge. We studied extensions of the GMMs using ensemble learning (i.e. bagging and Boosting).

1 Introduction

The human voice carries a great deal of information, not only in the words spoken, but also by way of pitch, harmonics and other auditory features. With these features, the field of speech recognition has been succesful in identity and gender recognition. In this paper, we study accent recognition. The accent may be due to the fact that the speaker is a non-native speaker of the spoken language (e.g. a native German speaking English), or that the speaker is a native speaker that is using a particular dialect of the language (e.g. an English speaker with a 'Southern' United States accent). The accurate identification of a speaker's accent has important uses. A non-native speaker's accent can be used to identify their native language. The system can then switch to that native language or pre-process the non-native speech to improve the accuracy of speech recognition. A native speaker's accent can also be used as a corroborative biometric feature.

The non-native speaker's accent is mainly characterized by the difficulties in correctly pronouncing certain words or affixes (e.g. in English, cough vs. though) or in pronouncing sounds that do not occur in their native language. The accents of native speakers on the other hand, are more subtle, being differentiated by how they pronounce certain syllables or, in some cases, if they pronounce the syllable at all. In general identifying native speaker accents is considerably harder than identifying accents for non-native speakers. We look at the problem of identifying native speaker accents and our experimental test bed is a data set containing the Northern and Southern accents in American English. The speakers are labeled as Northerners or Southerners depending on where they were raised as children.

There are two typical approaches to speech identification. The typical Hidden Markov Model (HMM) approach and, a common ML approach, the Gaussian Mixture Model (GMM). The former is used to model words or phonemes, the atomic speech unit, and requires transcripts or some preprocessing. Alternatively, the latter can work with the raw speech signal using an unsupervised method.

We build on the GMM approach by exploring the effects of applying ensemble learning. Ensemble learning improves upon classifiers at the expense of complexity and increased training time. Since, for accent identification, training is only performed once and separately from classification, it is acceptable to increase training time for higher accuracy. Also, the improvements we implement mostly add complexity to the training step and has a small impact on the time required for the identification step. In addition to bagging and Boosting, we present a simple ensemble method for pruning data, based off of the bagging method, which outperforms all the other methods.

The outline of the paper is as follows. In Section 2, we review the related work. Then, we define the problem setup in Section 3. We describe our approaches to improving the accuracy of the model in Section 4. We show our experimental results in Section 5 and conclude in Section 6.

2 Related Work

Accent identification and dialect identification can be thought of as a generalization of speaker identification.

*http:\\www.cs.rpi.edu\~purnej\

In [1], a Gaussian mixture model is successfully (80.8%) used to identify individual speakers with 15 seconds of speech over the telephone. It is show that the GMM is successful because it captures some spectral shapes that make the speaker's voice unique from other speakers. In our case, the problem is to find the spectral shapes that make the voices of a particular group unique from other groups. A particularly challenging aspect of accents is that the spectral shapes of two groups may be quite similar.

The problem of accent identification originally was focused on non-native accents. A non-native accent in a spoken language arises when the speaker has a different primary language (e.g., a German speaking English). A commonly used approach [2][3] involves modeling phonemes with HMMs. Accuracy generally fell between 65% and 70%. Higher rates were achieved when using specific words that were found, a priori, to be discriminative between accents. Research has been done in finding the informativeness of parts of the spectral data, such as certain formants [4]. There has also been research in using less supervised learning techniques, such as Support Vector Machines [5]. Pedersen achieved accuracies of about 72% and then up to 97.5% when having the speakers reading from a single page of text.

By far the harder problem is dialect or accent identification (the spoken language of the speaker is their primary language). While there are several possible factors that contribute to its formation, a dialect is predominantly correlated with geographical location [6].

Zheng [7] attempts to discriminate between dialects of Mandarin, using properties known a priori of Shanghai-accented Mandarin. The speakers were separated by experts into two groups: those having little accent and those having a strong accent. They report an accuracy of 69% on the weakly accented group and 86% on the strongly accented group. Zheng does not report results when using all the data to train accent classifiers, so it is not clear how to avoid the human expert who initially classifies the data into weak and strong groups.

To avoid building a phonetic-based model, Chen [8] used supervised GMMs on the standard features, Mel-frequency cepstral coefficients (MFCCs). They had a classification error 88.3% for the male speakers.

The work by Zheng et al. indicates the advantage gained in identification accuracy when the accent is treated as a matter of degree. A similar perspective exists in the machine learning community in such forms as outlier detection [9] [10], categorization by intrinsic margin [11], and data cleansing [12]. Angelova [13] applied this to the facial recognition problem by removing, or pruning, the 'noisy' samples and features from the training set. Vezhnevets [14] used this pruning technique to improve AdaBoost accuracy. Our work studies the effect of pruning and boosting with accent identification.

3 Accent Identification from Speech

3.1 Feature Set

The spectral space of speech yields very effective features for various speaker identification problems. This is because the spectrum of speech reflects their vocal tract which is the dominant physiological factor in the uniqueness of a speaker's voice. The most popular feature set is the cepstral coefficients of a mel-frequency filterbank. The mel-frequency filterbank emphasizes particular ranges of frequencies in a way that is similar to how the human ear perceives sound.

Our feature set comes from the mel-frequency cepstral coefficients (MFCC) of the speech samples. The data samples were pre-emphasized with $H(z) = 1 - 0.97z^{-1}$, windowed to 25-ms frames with 15-ms overlap, and parameterized into 39 features, consisting of 12 cepstral coefficients, the signal energy, and the first and second order differences of the coefficients and energy. Cepstral mean subtraction was performed in each data sample to remove effects of channels. This processing was done with the HTK 3.0 toolkit [15]. Figure 1 illustrates the complete process.

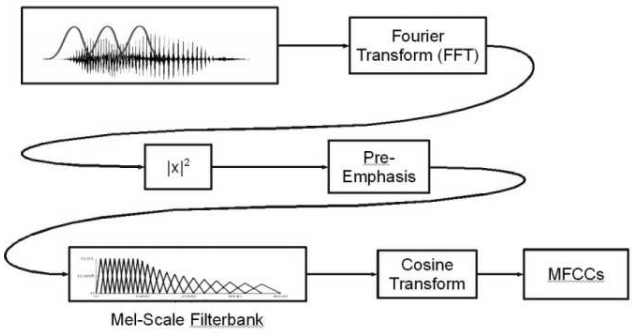

Figure 1. How the speech is pre-processed to produce MFCCs

3.2 Gaussian Mixture Model

A Gaussian mixture density is a weighted sum of M component densities. Each component density is a D variate Gaussian function with mean vector $\vec{\mu}_i$ and covariance matrix Σ_i. It is defined as:

$$p_i(\vec{x}) = \frac{\exp(-0.5(\vec{x} - \vec{\mu}_i)^T \Sigma^{-1} (\vec{x} - \vec{\mu}_i))}{(2\pi)^{D/2} \det(\Sigma)^{1/2}} \qquad (1)$$

where $\vec{x}$ is a D-dimensional random vector, $p_i(\vec{x})$, $i = 1, \ldots, M$, are the component densities and $\pi_i, i =, \ldots, M$

are the mixture weights, which satisfy the constraint that $\sum_{i=1}^{M} p_i = 1$.

The model is characterized by a parameter set consisting of the mixture weights, mean vectors, and covariance matrices for the M components. They are collectively notated

$$\lambda = \{\pi_i, \vec{\mu}_i, \Sigma_i\}\ i = 1, \ldots, M \tag{2}$$

The probability of a sample, $\vec{x}$, being generated by a given model is calculated

$$p(\vec{x}|\lambda) = \sum_{i=1}^{M} \pi_i p_i(\vec{x}) \tag{3}$$

Each of the N accents are modeled by GMMs that are independently trained using speech from the corresponding accent. In this case, we trained a GMM on Northern speakers and another GMM on Southern speakers. Training started with determining the initial mean values,$\vec{\mu}$, by choosing frames of speech randomly (and without replacement) from the training data. From the initial means, the covariance matrices were calculated. The initial mixture weights were set to be uniform.

After initializing the GMM parameters, a number of iterations of the EM algorithm are applied. In the expectation step, the probabilities of each frame of speech data are calculated by:

$$p_{ij} = \pi_i p_i(\vec{x}_j) \tag{4}$$

During the maximization step, the parameters are updated using the values from the previous expectation step, as follows:

$$\hat{\mu}_i = \frac{\sum_j p_{ij}\ \vec{x}_j}{\sum_j p_{ij}} \tag{5}$$

$$\hat{\Sigma}_i = \frac{\sum_j p_{ij}(\vec{x}_j - \hat{\mu}_i) \otimes (\vec{x}_j - \hat{\mu}_i)}{\sum_j p_{ij}} \tag{6}$$

$$\hat{\pi}_i = \frac{1}{A} \sum_i p_{ij} \tag{7}$$

where A is the number of data samples in the training set.

During identification, we looked at the sum of the logarithmic probabilities over all the speech data for a given speaker. This sum is defined

$$\sum_{t=1}^{T} \log p(\vec{x}_t|\lambda_k), k = 1, \ldots, N \tag{8}$$

where T is the number of frames of speech data. The speaker was then classified according to which accent model yielded the greatest average logarithmic probability for the speaker's speech, $\hat{k}$, as shown in

$$\hat{k} = \arg\max_{k=1}^{N} \sum_{t=1}^{T} p(\vec{x}_t|\lambda_k) \tag{9}$$

4 Extensions of the GMM

In Section 5, we will see that the GMM is a decent tool to use for accent identification, especially since it doesn't require a transcript for the speech or an explicit phoneme-based structure. However, learning is shown to suffer if 'hard' or 'confusing' samples exist in the training data. In the case of our corpus of American English accents, some of the samples come from speakers that have grown up in the Southern United States but, have very few indications of a Southern accent in their speech. By identifying and omitting possibly confusing samples, the models of the accents can be more accurate. In this section we look at some methods of dealing with these confusing samples.

4.1 Bagging

Bagging, also known as bootstrap aggregating and introduced by Breiman [16], is our first extension of the GMM. It is straight forward to implement, given the underlying classifier. A number of subsets, S_j, $j = 1 \ldots M$, are chosen randomly with replacement from the training set, S_{train}. Each subset has the same number of samples as the training set. In training, the samples from a subset are divided by their labeled accent, $n = 1 \ldots N$, and used to train a classifier, $h_j^n(\vec{x})$. So, we have a GMM for each subset-accent pair.

During classification the data is passed through each classifier. The probabilities are then summed over each accent. The data is classified by the accent that yields the greatest sum.

$$H_{bagging}(\vec{x}) = \arg\max_{n=1}^{N} \sum_{j=1}^{M} h_j^n(\vec{x}) \tag{10}$$

By combining the results over several classifiers, the intent is to decrease the influence of 'noisy' data on the final classification.

4.2 AdaBoost

The next extension we looked at was AdaBoost. This is a well known meta-algorithm that also works on the assumption that 'noisy' samples may be present in the training set. Instead of ignoring or compensating the noisy samples, AdaBoost focuses on learning to properly classify the noisy samples.

AdaBoost is an iterative algorithm where several classifiers are, in turn, trained and then applied to the training set. On each iteration the training samples are weighted based on how many times they were incorrectly classified by the previous classifiers. For the first iteration, the weighting is uniform. On each subsequent iteration the weights are updated to emphasize the noisy samples [17].

To find noisy samples, the classifier from the previous iteration is applied to the weighted training set. Each sample that is misclassified is determined as 'noisy' and has its weight increased. Conversely, the samples that are properly classified have their weights decreased. Thus, when the next classifier is trained it will put emphasis on correctly classifying samples proportional to the times they are misclassified over previous iterations.

To determine the proper weighting, we start with the following measure of error,$epsilon_t$, and error rate,α_t:

$$\epsilon_t = \sum_{j=1}^{A} w_t(j)[y_j \neq h_t(\vec{x}_j)]\alpha_t = 0.5 \ln \frac{1-\epsilon_t}{\epsilon_t} \quad (11)$$

where A is then number of samples, $w_t(j)$ is the weight for sample j at iteration t and $h_t(\vec{x}_j) = sign(p(j))$. The error rate is then used to update the sample weights as follows

$$w_{t+1}(j) = \frac{w_t(j) \exp^{-\alpha_t y_j h_t(\vec{x}_j)}}{\sum_{j=1}^{A} w_t(j) \exp^{-\alpha_t y_j h_t(\vec{x}_j)}} \quad (12)$$

After several iterations are completed, a 'strong' classifier, H, is formed by a linear combination of the iterative classifiers, h_t. The iterative classifiers are weighted, α_t, proportionally to their accuracy on classifying the training set. Assuming that the classifiers return +1 or -1, we define the relation between the strong classifier and the iterative classifiers as

$$H(\vec{x}) = sign(\sum_{t=1}^{T} \alpha_t h_t(\vec{x})) \quad (13)$$

where T is the number of iterative classifiers that were trained.

4.3 Pruning

Pruning starts off by training classifiers on random subsets of the training set, as in the bagging approach. These classifiers are called 'weak' classifiers, to differentiate them from the 'strong' classifier that is trained at the end.

First, the 'weak' classifiers are trained over the entire training set. For each data sample, the sum of log probabilities over all weak classifiers is calculated for each accent. A data sample is classified according to which accent yielded the greatest sum. A data sample that is classified differently from how its labeled, $y(\vec{x})$, is considered a 'hard' or 'noisy' sample. These samples are removed from the training set to create a new pruned training set. The 'strong' classifier is defined as a classifier trained on this pruned training set.

$$S_{pruned} \subset \cup_{\vec{x} \in S_{train}} H_{pruning}(\vec{x}) = y(\vec{x}) \quad (14)$$

where $H_{pruning}$ is defined in the same manner as $H_{bagging}$ in Eq. 10. The outline of the algorithm is listed in Algorithm 1.

Algorithm 1 Pruning

Input: data x_i, num. of subsets m, subset size n, accents a_j
for $k = 1$ **to** m **do**
  Generate subset k by choosing n data from x, uniformly and with replacement.
  Train 'weak' classifier on subset k.
end for
for each data x_i **do**
  for each accent a_j **do**
    for each subset k **do**
      Classify x_i with model for accent/subset (a_j,k)
      Add log prob. of x_i for (a_j,k) in a sum for accent a_j
    end for
  end for
  Find the accent with the greatest sum, a'
  if x_i's label matches a' **then**
    Add x_i to the prune training set
  else
    Discard x_i
  end if
end for
Train 'strong' classifier on pruned training set

5 Experiments

The experiments conducted involved classifying data samples, which were 25ms frames of speech. After classifying these samples, they were grouped by spoken word. The spoken word was classified based on which accent yielded the greatest average log probability over all samples. Similarly, the speakers were classified based on which accent yields the greatest average log probability over all of the speaker's spoken words.

The complete (i.e.unpruned) training set was recorded from 98 speakers and contained 8,295 words, with 4,490 coming from the 53 Northern speakers and 3,805 coming from the 45 Southern speakers. The testing set was recorded from 35 speakers and contained 2,988 words with 1,542

coming from the 18 Northern speakers and 1,446 coming from the 17 Southern speakers. Due to the small number of speakers in the given test set, we used cross-validation as described below.

The speech was taken from the TIMIT's Acoustic-Phonetic Continuous Speech Corpus. The speakers were primarily Texas Instrument personnel and selected to be representative of different geographical dialect regions. The dialect region was defined as the geographical area of the U.S. where the speaker lived during their childhood years (ages 2 to 10). The Northern region primarily consisted of New York, Michigan, Wisconsin, Minnesota, and the Dakotas. The Southern region primarily consisted of Louisiana, Mississippi, Alabama, southern Georgia, the eastern half of the Carolinas and Florida.

5.1 Speech Processing and GMM Training

Speech processing was done with the HTK toolkit [15]. Although the HTK provides tools for training GMMs it did not provide a way of applying weights to the samples, so separate code was used in order to perform boosting on the frames of speech. In training the GMM, we used an EM approach. On each iteration of training, the parameters were initialized and then re-estimated once. After looking at using 8, 16 and 32 components in the GMM (see Table 1), we decided to use 8 mixtures for the GMMs used in the experiments since this allowed the experiments to run more quickly and still revealed the differences in accuracies between the methods.

5.1.1 Experiments

Classifiers were trained with 4 iterations of the EM approach. For bagging and the pruning methods, 8 weak classifiers were used. For boosting, 8 iterations of boosting were performed. The methods were tested by cross-validation, leaving out one speaker for testing and training on the remaining 132 speakers. The results of the experiments are shown in Table 1.

First, it can be seen that the increase in the number of components in the GMM seems to have little or no effect. The accuracy per sample and per word remains constant and the accuracy per speaker increases slightly. Applying the Bagging and Boosting technique shows an overall increase in accuracy, with Bagging having particularly increased the per speaker accuracy.

Pruning shows an interesting trend. While the per sample and per word accuracy are similar to the GMM accuracies, the per speaker accuracy increases significantly over the Bagging and Boosting. The reason behind this is that the correctly classified words were more evenly distributed over the speakers with pruning rather than in bagging or boosting. In other words, it is more likely that a speaker will have more than 50% of their words correctly classified with pruning than with the other techniques. This results in getting a higher per speaker accuracy despite a lower per word accuracy.

Table 1. Accuracies of GMM and the Extensions

METHOD	PER SAMPLE	PER WORD	PER SPEAKER
GMM-8	51.5%	54.6%	60%
GMM-16	51.1%	52.6%	62.9%
GMM-32	51.7%	53.5%	62.9%
BAGGING	52.8%	55.1%	65.9%
BOOSTING	51.9%	54.4%	63.9%
PRUNING	51.2%	53.2%	69.92%

6 Conclusion

Recent research has shown what seems to be an upper bound of about 65% when it comes to correctly identifying a speaker's accent without using a priori knowledge of the accent's characteristics. Applying ensemble learning methods we have demonstrated the ability to increase speaker accent accuracy without using additional information. This is beneficial in that there is no requirement of have a transcript of the speech, training phoneme-based models, or extensive supervision.

In this paper, our experiments involved discerning a Northern American English accent from a Southern accent. In the future, we plan to explore multi-accent identification and the ability to discern similar sounding accents (eg. Southern accent vs. Carolina accent). Also, our results show that there may be a chance to improve on identifying a speaker's native language. In this case, training would be done on speakers using a language that is their secondary language with the goal of identifying the primary language. Since the approaches to accent identification are highly related to non-native accent identification, it is very likely that advances in one area will work well in the other area.

Further, we plan to explore other ensemble learning techniques that replicate the ability of human experts to determine the strength of an accent, as in Zheng's work [7]. This may include combining or hybridizing the ensemble techniques we explored in this paper.

References

[1] D. Reynolds and R. Rose, "Robust text-independent speaker identification using gaussian mixture speaker models," *Speech and Audio Processing, IEEE Transactions on*, vol. 3, no. 1, pp. 72–83, Jan 1995.

[2] C. Teixeira, I. Trancoso, and A. Serralheiro, "Accent identification," *Spoken Language, 1996. ICSLP 96. Proceedings., Fourth International Conference on*, vol. 3, pp. 1784–1787 vol.3, Oct 1996.

[3] L. M. Arslan and J. H. L. Hansen, "Language accent classification in american english," *Speech Commun.*, vol. 18, no. 4, pp. 353–367, 1996.

[4] L. W. Kat and P. Fung, "Fast accent identification and accented speech recognition," *Acoustics, Speech, and Signal Processing, 1999. ICASSP '99. Proceedings., 1999 IEEE International Conference on*, vol. 1, pp. 221–224 vol.1, Mar 1999.

[5] C. Pedersen and J. Diederich, "Accent classification using support vector machines," *Computer and Information Science, 2007. ICIS 2007. 6th IEEE/ACIS International Conference on*, pp. 444–449, July 2007.

[6] H. Kurath, *A Word Geography of the Eastern United States*. University of Michigan Press, 1967.

[7] Y. Zheng, R. Sproat, L. Gu, I. Shafran, H. Zhou, Y. Su, D. Jurafsky, R. Starr, and S.-Y. Yoon, "Accent detection and speech recognition for shanghai-accented mandarin," *INTERSPEECH*, pp. 217–220, 2005.

[8] T. Chen, C. Huang, E. Chang, and J. Wang, "Automatic accent identification using gaussian mixture models," *Automatic Speech Recognition and Understanding, 2001. ASRU '01. IEEE Workshop on*, pp. 343–346, Dec. 2001.

[9] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, p. 2004, 2004.

[10] W. R. Mebane and J. S. Sekhon, "Robust estimation and outlier detection for overdispersed multinomial models," *American Journal of Political Science*, vol. 48, pp. 391–410, 2004.

[11] L. Li, A. Pratap, H. tien Lin, and Y. S. Abu-mostafa, "Improving generalization by data categorization," in *Knowledge Discovery in Databases: PKDD 2005, volume 3721 of Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2005, pp. 157–168.

[12] I. Guyon, N. Matic, and V. Vapnik, "Discovering informative patterns and data cleaning," 1996.

[13] A. Angelova, Y. Abu-mostafa, and P. Perona, "Pruning training sets for learning of object categories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, 2005, pp. 494–501.

[14] A. Vezhnevets and O. Barinova, "Avoiding boosting overfitting by removing confusing samples," in *Machine Learning: ECML 2007*. Springer-Verlag, 2007, pp. 430–441.

[15] "Hidden markov model toolkit (htk)," http://htk.eng.cam.ac.uk, 2007.

[16] L. Breiman and L. Breiman, "Bagging predictors," in *Machine Learning*, 1996, pp. 123–140.

[17] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Japanese Society for Artificial Intelligence*, vol. 14, no. 5, pp. 771–780, 1999.