SPEECH RECOGNITION WITH LOCALIZED TIME-FREQUENCY PATTERN DETECTORS

Ken Schutte, James Glass

MIT Computer Science and Artificial Intelligence Laboratory
32 Vassar St., Cambridge, MA 02139, USA
{*kschutte,glass*}*@mit.edu*

ABSTRACT

A method for acoustic modeling of speech is presented which is based on learning and detecting the occurrence of localized time-frequency patterns in a spectrogram. A boosting algorithm is applied to both build classifiers and perform feature selection from a large set of features derived by filtering spectrograms. Initial experiments are performed to discriminate digits in the Aurora database. The system succeeds in learning sequences of localized time-frequency patterns which are highly interpretable from an acoustic-phonetic viewpoint. While the work and the results are preliminary, they suggest that pursuing these techniques further could lead to new approaches to acoustic modeling for ASR which are more noise robust and offer better encoding of temporal dynamics than typical features such as frame-based cepstra.

Index Terms— automatic speech recognition, acoustic modeling

1. INTRODUCTION

The goal of this work is to explore front-end architectures for automatic speech recognition (ASR) which may offer ways to effectively model the time-frequency patterns crucial for speech perception. In particular, the work here primarily addresses methods to bring two improvements over current acoustic modeling: the use of localized features, and more explicit modeling of temporal dynamics.

Traditional ASR systems rely on frame-based spectral features, such as MFCCs or PLPs. There has been much work proposing alternatives which do not suffer from the *non-localized* nature of these features – i.e. each feature value can be affected by energy at any frequency. These ideas include sub-band recognizers, missing feature methods [2], TRAPs [5], and Gabor analysis [6]. Many of these systems have shown performance gains, but there is more work needed to effectively combine localized features (or estimate binary masks in the case of missing feature methods).

Another area in which traditional current modeling techniques are lacking is the modeling of temporal dynamics. Dynamic events, such as formant transitions, are very important in the determination of phonetic identity. However, the typical way to capture such information is through the use of crude delta features and discrete HMM states, which are not a natural fit to smooth temporal changes. We would like to use a set of features which are designed specifically to capture such information.

For example, consider the spectrogram in Figure 1. A particularly prominent acoustic-phonetic feature is the rising energy of the second and third formants during /$r\ i^y$/ starting at t=0.6 seconds. Next to the spectrogram is shown a localized *time-frequency filter* designed to locate an event such as this. The response shown below is the result of convolving this "patch" over the spectrogram. Notice that when the features themselves are designed to capture specific spectro-temporal events, the response can be very pronounced.

Our approach will be to utilize a large bank of time-frequency filters to generate features, and learn models of our phonetic units which capture the time-evolution of these features. This approach is influenced by past work which utilized spectro-temporal filters [3, 6], and models based on local spectrogram patterns [1].

2. FEATURES AND DATA

2.1. Time-Frequency Filters

While we ultimately might prefer to learn our set of filters from data, our initial work is based on manually designing a set of basic time-frequency patterns with which we will filter our spectrograms. A subset of such filters are shown in Figure 2. While we have explored other, more complicated shapes and parameterizations, the set used

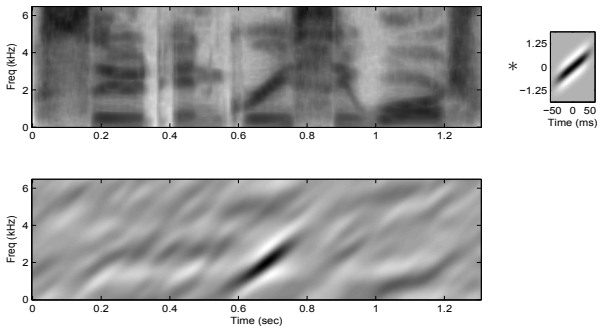

Fig. 1. The spectrogram of a section of a TIMIT utterance, and its response to a small time-frequency filter designed to detect formant transitions at slope 15Hz/ms. Notice the strong response for /r i^y/ in "greasy" starting at t=0.6 sec.

here is very simple, and are essentially basic edge detectors taking only the values $+1$ and -1. The selection includes vertical edges (of varying frequency span and temporal duration) for onsets and offsets; wide horizontal edges for frication cutoffs; and horizontal edges tilted at various slopes to model formant transitions. The choices for the ranges of the various parameters were made based on acoustic phonetic knowledge, such as typical formant bandwidths, average phone durations, typical rates of formant movement, etc [10].

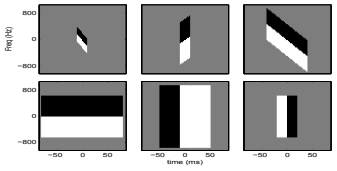

Fig. 2. Several examples of the time-frequency filters employed. The set used here had 291 such filters with various parameters.

Taking one such filter, $p_i(t, \omega)$, centering it at a particular frequency, ω_i, and convolving[1] with a spectrogram $x(t, \omega)$, results in a "feature", which is a function of time,

$$f_i(x; t) = \sum_{\tau} \sum_{g} p_i(\tau, g) x(t - \tau, \omega_i - g) \qquad (1)$$

The next section will describe how models are learned from such features.

[1] Actually, it is the convolution with the flipped version of this filter, $p(-t, -\omega)$, resulting in the dot-product of $p(t, \omega)$ centered at each point in the spectrogram. We will use the term "convolution" without making this distinction.

2.2. Biological Interpretation

While our goal is not to explicitly design a biologically-plausible system, it is worth noting that the features (and how they will be used, described later) have a natural biological interpretation. Recent work [3] has characterized subsets of neurons in the primary auditory cortex as having approximately a linear response to a particular time-frequency pattern, referred to as a spectro-temporal receptive field (STRF).

Each of our features, $f_i(t)$ is analogous to the firing rate for a single cortical neuron with STRFs shaped like the patches in Figure 2 and located at a particular center frequency. The overall classification system, as will be described, consists of linear combinations of the "firing" of these neurons in a particular temporal pattern. Therefore, the overall system output can be thought of as another neural-network layer on top of such cortical neurons.

2.3. Features

For each phonetic unit, we would like to learn a model of the temporal sequence of the filter responses, $f_i(t)$. For example, the model might encode pieces of information of the form, "phonetic unit m is characterized, in part, by a strong response from filter p_i centered at time-frequency point (t_i,ω_i)". As will be explained, the experiments described here use whole words as the fundamental units. To discretize the possible time-frequency locations for particular events, we sample filter responses over a 16x32 point grid over the T-F plane of each training token (each word). The 32 frequency points are taken linearly between 0 and 4kHz. Choosing the 16 time points needs to be treated more carefully. We use a conventional speech recognizer (HMM/GMM/MFCC-based) with 16 states per word to create forced-path alignments over all of our data. The centers of the 16 states in the state alignment are used as the time samples for each word. Therefore, in the discussion of *feature selection* below, a single "feature" refers to both the filter shape, and the time-frequency point (in this 16x32 point grid) at which it is centered.

3. BOOSTING ALGORITHM AND FEATURE SELECTION

Our goal is to take these filter outputs at particular times and frequencies, and learn a pattern to discriminate phonetic units (in the task explored here, to discriminate isolated words). Considering each filter at each point in

Input: $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)$, spectrograms $\mathbf{x}$, binary labels $y_i \in \{+1, -1\}$.
Output: Parameters $\{f_t, \theta_t, \alpha_t\}$, $t = 1, \ldots, T$ for use in Equation 2.
begin
$w_{0,i} = \begin{cases} 1/(2\sum_{Y_i=+1} 1), & Y_i = +1 \\ 1/(2\sum_{Y_i=-1} 1), & Y_i = -1 \end{cases}$
for $t = 1, \ldots, T$ **do**
$f_t, \theta_t, \epsilon_t \longleftarrow \min_{f,\theta} \epsilon$
i.e. find feature and threshold to minimize weighted error (solved via Algorithm 2).
$\alpha_t \longleftarrow \log((1-\epsilon_t)/\epsilon_t)$
$w_{t+1,i} = \begin{cases} Z_t w_{t,i}\, \epsilon_t/(1-\epsilon_t) & \mathbf{x}_i \text{ correct} \\ Z_t w_{t,i}, & \text{otherwise.} \end{cases}$
where Z_t normalizes s.t. $\sum w_{t+1} = 1$.
end

Algorithm 1: AdaBoost for classification and feature selection, as in [11].

Input: Pre-computed, pre-sorted features $f_j(\mathbf{x}_i)$; weights on each training point, w_i; labels $y_i \in \{+1, -1\}$
Output: $\{\theta^*, f^*, \epsilon^*\} = \min_{\theta,f} \epsilon$
begin
$W_{+1} \longleftarrow \sum_{Y_i=+1} w_i$
$W_{-1} \longleftarrow \sum_{Y_i=-1} w_i$
$\epsilon^* \longleftarrow \infty$
foreach f_j **do**
$\epsilon \longleftarrow W_{+1}$
$\overline{\epsilon} \longleftarrow W_{-1}$
for $i = 1, \ldots, N$ *(sorted order)* **do**
$\epsilon \longleftarrow \epsilon + Y_i w_i$
$\overline{\epsilon} \longleftarrow \overline{\epsilon} - Y_i w_i$
if $\epsilon < \epsilon^*$ **then** $\epsilon^*, f^*, i^* \longleftarrow \epsilon, f_j, i$
if $\overline{\epsilon} < \epsilon^*$ **then** $\epsilon^*, f^*, i^* \longleftarrow \overline{\epsilon}, -f_j, i$
$\theta^* = \text{mean}(f^*(\mathbf{x}_{i^*}), f^*(\mathbf{x}_{i^*+1}))$
end

Algorithm 2: Efficient method for optimal feature selection and calculation of weighted error, ϵ, as defined in Equation 4.

our "grid" results in a large number of possible features, many of which are likely to be uninformative.

To address these problems, we use a boosting method which was proposed by Viola and Jones [11] for face detection in images. The algorithm is essentially AdaBoost [4], in which the "weak classifier" in each iteration chooses the single best feature available – simultaneously performing both classification and greedy feature selection. The details of the algorithm are given in the figures above, and we discuss it below.

3.1. Algorithm

Boosting algorithms (and in particular, AdaBoost [4]) attempt to combine multiple "weak classifiers" into a single strong classifier. We choose our weak classifiers to be "decision stumps" (a popular choice for AdaBoost), meaning that each classifier simply chooses a single scalar dimension and a threshold – each side of the threshold is assigned to one of the two classes (performing binary classification). The boosting procedure creates a weighted vote of these simple decisions.

Mathematically, the algorithm results in a function to compute a score, $S(\mathbf{x})$, for an input spectrogram, $\mathbf{x}$, of the form,

$$S(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \,\text{sign}(f_t(\mathbf{x}) - \theta_t) \quad (2)$$

$$= \sum_{f_t(\mathbf{x}) \geq \theta_t} \alpha_t - \sum_{f_t(\mathbf{x}) < \theta_t} \alpha_t \quad (3)$$

The binary decision (hypothesized class $+1$ or -1), is taken as the sign of $S(\mathbf{x})$. This is a simple weighted vote over T decision stumps (using T features). If a feature value exceeds its threshold (i.e. that neuron "fires"), the weight is added; otherwise, it is subtracted.

AdaBoost chooses parameters, α_t, as shown in Algorithm 1 (more details in [9]). The weight given to each weak classifier is a function of its *weighted error*, ϵ_t, on the training set,

$$\epsilon_t = \frac{1}{2}\sum_{i=1}^{N} w_{t,i}\,[1 - Y_i \,\text{sign}(f_t(\mathbf{x}_i) - \theta_t)] \quad (4)$$

$$= \sum_{\substack{f_t,\theta_t \text{ mis-} \\ \text{classify } \mathbf{x}_i}} w_{t,i} \quad (5)$$

Each training point is given a weight, w_i, which changes on each iteration to reflect how well it has been classified by previous iterations. Each iteration "concentrates on" those examples which have been difficult to classify in previous iterations.

As was proposed in [11], we choose the weak learner in each iteration by searching our features for the best available decision stump classifier. We present an efficient method to perform this search in Algorithm 2. If the features are sorted, then the weighted error can be tracked while stepping through each training point one

at a time, hypothetically placing the threshold between each one. This algorithm simultaneously considers the feature for a particular filter, $p_j(t, \omega)$ and the feature corresponding to the filter with the opposite sign, $-p_j(t, \omega)$, which doubles the number of potential features with little overhead. Comments on the computational time for this search are discussed in Section 3.3.

3.2. Multi-class

The algorithm as described is for binary classification. While there are a number of proposed approaches for AdaBoost for multi-class problems [8], for now, we use a simple *one-vs-all* (OVA) scheme to do 11-way classification. A model is trained for each word (with all others as negative examples), and we choose the class with the highest normalized score,

$$k^* = \underset{k \in W}{\operatorname{argmax}} \frac{1}{\sum_{t=1}^{T} \alpha_{t,k}} \sum_{t=1}^{T} \alpha_{t,k} \operatorname{sign}(f_{t,k}(\mathbf{x}) - \theta_{t,k}) \quad (6)$$

An additional subscript, k, has been added to denote the class, i.e. the word, $W = \{\textit{one}, \textit{two}...\}$. The weights are normalized to facilitate comparisons across classes. A benefit of the OVA approach is that the set of features learned for a particular word can be viewed as word "prototypes", as will be shown.

3.3. Computational Issues

Choosing the best decision stump classifier on each iteration performs a brute-force search over features and thresholds; however, as shown in Algorithm 2, this can be accomplished very efficiently. For one set of parameters we used, there were essentially 297,984 potential features,[2] from which a single one is chosen as f_t in each iteration. While each iteration of AdaBoost must be run in sequence, the search over features in each iteration can be parallelized. Using a cluster of roughly 50 standard desktop machines, this technique can find the best feature for all 11 binary classifiers in about 5 minutes. Note that 11 times 297,984 results in over 3.2 million features, and for each one it is checking every potential unique threshold value on the training set of 27,727 words.

[2] $297,984 = 291$ "prototype" filter shapes $\times$ 2 signs (+1/-1) $\times$ 32 frequency points $\times$ 16 time points.

4. EXPERIMENTS AND ANALYSIS

4.1. Data

We explore these techniques on the task of isolated digit recognition on the Aurora database [7]. While this is a simple task, it is well suited to our goals. Using more complex units (such as whole words or syllables) offers more temporal structure to be modeled, compared to other reduced ASR tasks such as phonetic classification or recognition. Our preliminary goals are to show the feasibility for new approaches, rather than requiring cutting-edge performance on difficult tasks. All results presented use the clean training set and test set A-N1 (train noise).

4.2. Selected Features

Figure 3 shows a sample of the output from running the learning algorithm on the clean Aurora training set. Notice that the features selected have very recognizable phonetic identities, as described in the figure's caption. Having such interpretable features is a very desirable property for analyzing the weaknesses of a system, which is typically not easy with traditional frame-based features and classifiers.

4.3. Classification Performance

To perform classification, we use the times of the 16 states taken from the original HMM alignments corresponding to the test word. While this is an unrealistic test condition, it presents data in the way done in training, so it can effectively test the performance of the learning algorithm. The digit classification results for this case are shown in the Table 1. In general, it is very accurate. It is interesting to note how well a *single feature* can perform – e.g. the best feature for the word "six" has a 0.15% test error (5/3257 in test set).[3] This feature happens to detect the low frequency offset of voicing at the /k/ closure.

Testing was also done on various noise conditions in Aurora. The first row in Table 2 shows these results using $T = 100$ localized features. The table also shows results using a baseline HMM-based recognizer using MFCCs and GMMs. Note, however, that for the results of our system, additional information is known – namely the 16 time points at which to measure the feature values. Therefore, the HMM results should be taken as a reference for typical performance, rather than a direct comparison to the described system. Future work must ad-

[3] The method used tries to minimizes the equal-error-rate for each classifier. Minimizing the total-error-rate for the "six" actually results in 3/3257 = 0.09% error for the single best feature.

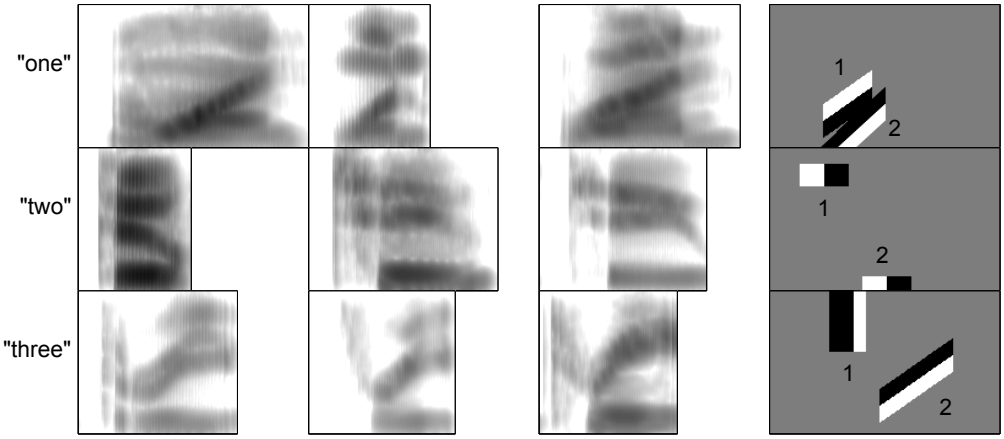

Fig. 3. Example of learned features. The left three columns show three example spectrograms of the words "one", "two", and "three".The column on the right shows the first two features that the algorithm chooses to distinguish the respective words (white=-1, black=$+1$). Note the recognizable phonetic interpretations: for "one", the upper and lower edges of a rising F2 for /$w\,ah$/; for "two", the onset of high-frequency aspiration for /t/, followed by low-frequency voicing onset (note the implicit encoding of voice-onset-time); for "three", high-frequency offset for the end of /θ/, followed by the lower edge of a rising F2/F3 for /$r\,i^y$/.

	Training Set			**Test Set**		
	% Error		min T w/	% Error		
	T=1	T=10	zero error	T=1	T=10	T=100
one	2.27	0.13	18	2.03	0.18	0.03
two	1.37	0.01	14	1.60	0.12	0.03
three	4.98	0.16	20	5.43	0.37	0.00
four	1.78	0.21	25	1.87	0.25	0.03
five	4.02	0.32	32	3.65	0.55	0.06
six	0.16	0.00	3	0.15	0.03	0.00
seven	12.14	0.80	34	11.82	1.14	0.00
eight	1.60	0.04	14	1.78	0.09	0.00
nine	14.12	0.70	30	14.83	0.95	0.09
zero	5.68	0.13	15	6.08	0.40	0.06
oh	7.13	1.87	53	7.34	2.21	0.06
M.C.	12.42	0.18	23	13.85	0.46	0.06

Table 1. Table of classification error rates with T iterations of boosting, i.e. using T features. The third column shows how many iterations it takes for the training error to reach zero. The bottom row gives multi-class error. The test set has 3257 words, so 0.03% is a single error.

dress hypothesizing these time points, as discussed in the next section.

5. DISCUSSION

We have presented initial work on modeling speech with a collection of localized time-frequency pattern detectors. While the results are only at the level of proof-of-concept, we consider them encouraging signs for future

	SNR (dB)						
	clean	20	15	10	5	0	-5
Local features	0.06	0.43	1.57	5.00	14.6	35.2	58.9
HMM	0.40	0.86	2.18	7.55	21.7	48.3	71.9

Table 2. Classification error rates at a range of signal-to-noise (SNR) ratios. Note that these two systems use different information – see section 4.3.

work. For each model, the system learns a sequence of localized events which are easily interpretable as known acoustic-phonetic phenomena, and have strong discriminative ability. Other potential benefits not discussed here include the fast decoding times and the potential for learning from few training examples.

There are many directions for future work. The results presented here do not deal with locating the time points at which to sample the features during decoding. Ongoing work is attempting to use a dynamic programming algorithm to choose the time points such that the score in Equation 2 is maximized. Another future direction is to move to more complex tasks, and explore ways to use these techniques in a full, continuous speech recognition system. Doing so will likely require choosing basic modeling units other than whole words, such as syllables, triphones, or diphones.

6. REFERENCES

[1] Y. Amit, A. Koloydenko, and P. Niyogi. Robust acoustic object detection. *Journal of the Acoustical Society of America*, 118:2634–2648, 2005.

[2] J. Barker, M. Cooke, and D. Ellis. Decoding speech in the presence of other sources. *Speech Communication*, 45:5–25, 2005.

[3] T. Chi, P. Ru, and S. Shamma. Multiresolution spectrotemporal analysis of complex sounds. *Journal of the Acoustical Society of America*, 118:887–906, 2005.

[4] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[5] H. Hermansky and S. Sharma. Temporal patterns (TRAPs) in ASR of noisy speech. In *International Conference on Acoustics, Speech, and Signal Processing*, 1997.

[6] M. Kleinschmidt and D. Gelbart. Improving word accuracy with gabor feature extraction. In *International Conference on Spoken Language Processing*, 2002.

[7] D. Pearce and H.-G. Hirsch. The Aurora experimental framework for the performance evaluation of speech recognition systems under noisy conditions. 2000.

[8] R. E. Schapire. Using output codes to boost multiclass learning problems. In *Proc. 14th International Conference on Machine Learning*, pages 313–321. Morgan Kaufmann, 1997.

[9] R. E. Schapire. *The Boosting Approach to Machine Learning – An Overview*. Springer, 2003.

[10] K. N. Stevens. *Acoustic Phonetics*. The MIT Press, Cambridge, MA, 1998.

[11] P. Viola and M. Jones. Robust real-time object detection. In *Workshop on Statistical and Computational Theories of Vision*, Vancouver, Canada, July 2001.