IDIAP RESEARCH REPORT

Theoretical Foundations for Large-Margin Kernel-Based Continuous Speech Recognition

Joseph Keshet

IDIAP–RR 07-44

September 2007

IDIAP Research Institute **www.idiap.ch**
Av. des Prés Beudin 20 P.O. Box 592 1920 Martigny – Switzerland
Tel: +41 27 721 77 11 Fax: +41 27 721 77 12 Email: info@idiap.ch

IDIAP Research Report 07-44

Theoretical Foundations for Large-Margin Kernel-Based Continuous Speech Recognition

Joseph Keshet

September 2007

1 Introduction

Automatic speech recognition (ASR) is the process of converting a speech signal to a sequence of words, by means of an algorithm implemented as a computer program. While ASR does work today, and it is commercially available, it is extremely sensitive to noise, speaker and environmental variability. The current state-of-the-art automatic speech recognizers are based on generative models that capture some temporal dependencies such as hidden Markov models (HMMs). While HMMs have been immensely important in the development of large-scale speech processing applications and in particular speech recognition, their performance is far from the performance of a human listener. In this work we present a different approach to speech recognition, which is not based on the HMM but on the recent advances in large margin and kernel methods.

Despite their popularity, HMM-based approaches have several known drawbacks such as convergence of the training algorithm (EM) to a local maxima, conditional independence of observations given the state sequence and the fact that the likelihood is dominated by the observation probabilities, often leaving the transition probabilities unused [13, 26]. However, the most acute weakness of HMMs for speech recognition task is that they do not aim at minimizing the word error rate.

Segment models were proposed by Ostendorf and her colleagues [13, 14] to address some of the shortcomings of the HMMs. In summary, segment models can be thought of as a higher dimensional version of a HMM, where Markov states generate random sequences rather than a single random vector observation. The basic segment model includes an explicit segment-level duration distribution and a family of length-dependent joint distributions. The model proposed in this work generalizes the segment model approach, so it addresses the same shortcomings of the HMM already addressed by the segment models. In addition, our model addresses two important limitations of the HMMs and the segment models as learning algorithms. Namely, the direct minimization of the word error rate and the convergence of the training algorithm to a global minima rather than to a local one. Moreover, our model is trained with a large margin algorithm which has been found to be robust to noise. Lastly, our model can be easily transformed into a non-linear model.

The alternative approach proposed in this work builds upon recent work on discriminative supervised learning. The advantage of discriminative learning algorithms stems from the fact that the objective function used during the learning phase is tightly coupled with the decision task one needs to perform (the word error rate, in our case). In addition, there is both theoretical and empirical evidence that discriminative learning algorithms are likely to outperform generative models for the same task (see for instance [4, 25]). One of the main goals of this work is to extend the notion of discriminative learning to the task of full-blown continuous speech recognition.

Our method is based on recent advances in kernel machines and large margin classifiers for sequences [21, 23], which in turn build on the pioneering work of Vapnik and colleagues [4, 25]. The speech recognizer we devise is based on mapping the speech signal along with the target word sequence into a vector-space endowed with an inner-product. Our learning procedure distills to a classifier in this vector-space which is aimed at ranking the word sequences according to their word error rate (Levenshtein distance) from the correct word sequence. On this aspect, our approach is hence related to support vector machine (SVM), which has already been successfully applied in speech applications [9, 20]. However, the model proposed in this paper is different from a classical SVM since we are not addressing a simple decision task such as binary classification or regression.

The remainder of the paper is organized as follows. In Sec. 2 we review the segmental and hidden Markov models for continuous speech recognition. In Sec. 3, we introduce the kernel-based model for speech recognition and show its relationship to the segment models. Our approach is based on non-linear phoneme recognition and segmentation functions. The specific feature functions we use presented in Sec. 4. In Sec. 5 we give a short description of the kernel-based decoder. Then, in Sec. 6 we present the large margin approach for learning the model parameters as a quadratic optimization problem. Since the standard solvers for such quadratic problem are not suitable for large speech datasets, we propose an iterative method for solving it in Sec. 7.

2 Segmental and Hidden Markov Models

The problem of speech recognition involves finding a sequence of words, $\bar{w} = (w_1, \ldots, w_N)$, given a sequence of d-dimensional acoustic feature vectors, $\bar{\mathbf{x}} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T)$, where $\mathbf{x}_t \in \mathcal{X}$, and $\mathcal{X} \subset \mathbb{R}^d$ is the domain the acoustic vectors. Each word $w_i \in \mathcal{V}$ belongs to a fixed and known vocabulary $\mathcal{V}$. Typically, each feature vector covers a period of 10 msec and there are approximately $T = 300$ acoustic vectors in a 10 word utterance. Our basic notation is depicted in Fig. 1.

In the segment model or the traditional HMM speech recognition systems, the problem of speech recognition is formulated as a statistical inference problem of finding the sequence of words $\bar{w}$ that is most likely given the acoustic signal $\bar{\mathbf{x}}$ by the *maximum a posteriori* (MAP) rule as follows

$$\bar{w}' = \arg\max_{\bar{w}} \mathbb{P}(\bar{w}|\bar{\mathbf{x}}) = \arg\max_{\bar{w}} \frac{\mathbb{P}(\bar{\mathbf{x}}|\bar{w})\mathbb{P}(\bar{w})}{\mathbb{P}(\bar{\mathbf{x}})} \ , \tag{1}$$

where we used Bayes' rule to decompose the posterior probability in the last equation. The term $\mathbb{P}(\bar{\mathbf{x}}|\bar{w})$ is the likelihood of observing the acoustic vector sequence $\bar{\mathbf{x}}$ given a specified word sequence $\bar{w}$ and it is known as *the acoustic model*. The term $\mathbb{P}(\bar{w})$ is the probability of observing a word sequence $\bar{w}$ and it is known as *the language model*. The term $\mathbb{P}(\bar{\mathbf{x}})$ can be disregarded, since it is constant under the max operation.

The HMM decoding process starts with a postulated word sequence $\bar{w}$. Each word w_i is converted into a sequence of phones $\bar{p}$, where $\bar{p} \in \mathcal{P}^\star$ and $\mathcal{P}$ is the set of all phone symbols. The conversion between a word to its corresponding phone sequence is done using a pronunciation lexicon lex, where $lex : \mathcal{V} \to \mathcal{P}^\star$. Namely,

$$\bar{p} = (lex(w_1), \ldots, lex(w_N)) \ .$$

The phone sequence is then converted to a state sequence, $\bar{q} \in \mathcal{Q}^\star$, where $\mathcal{Q}$ is the set of all HMM states. Usually every phone is represented as a sequence of 3 or 5 HMM states. The probability of the postulated sequence $\mathbb{P}(\bar{\mathbf{x}}|\bar{p})$ is computed by concatenating the models of

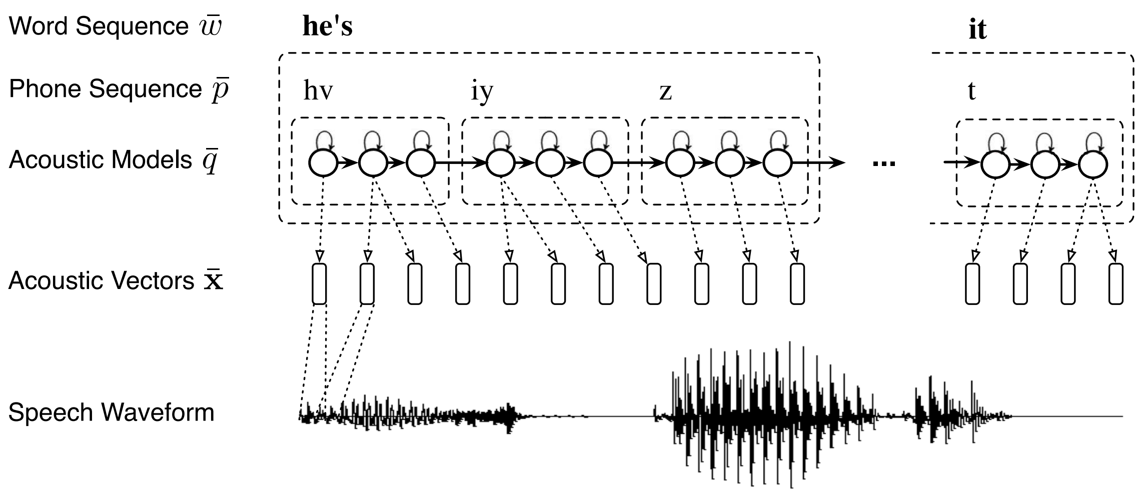

Figure 1: The basic notation of HMM speech recognizer.

the contextual phones composing the sequence. That is,

$$\begin{aligned} \mathbb{P}(\bar{\mathbf{x}}|\bar{p}) &= \sum_{\bar{q}} \mathbb{P}(\bar{\mathbf{x}}, \bar{q}|\bar{p}) = \sum_{\bar{q}} \mathbb{P}(\bar{\mathbf{x}}|\bar{q}, \bar{p}) \cdot \mathbb{P}(\bar{q}|\bar{p}) && (2) \\ &= \sum_{\bar{q}} \prod_{t=1}^{T} \mathbb{P}(\mathbf{x}_t|q_t) \cdot \mathbb{1}_{\{\bar{q},\bar{p}\}} \prod_{t=1}^{T} \mathbb{P}(q_t|q_{t-1}) \ , && (3) \end{aligned}$$

where $\mathbb{1}_{\{\bar{q},\bar{p}\}}$ is an indicator function that equals one if the state sequence $\bar{q}$ is permissible by the phone sequence $\bar{p}$ and zero otherwise. The Viterbi decode for HMMs involves finding the most likely state sequence,

$$\bar{q}' = \arg\max_{\bar{q}} \mathbb{P}(\bar{\mathbf{x}}|\bar{q})\mathbb{P}(\bar{q})$$

In the segment model, each phone p_l in $\bar{p}$ is modeled as a segment, including several frames. Let $\bar{s}$ be the timing (alignment) sequence corresponding to a phone sequence $\bar{p}$. Each $s_l \in \mathbb{N}$ is the start-time of phone p_l in frame units, that is s_l is the start-time of segment l. The probability of the postulated sequence $\mathbb{P}(\bar{\mathbf{x}}|\bar{p})$ using segment model is computed as follows

$$\begin{aligned} \mathbb{P}(\bar{\mathbf{x}}|\bar{p}) &= \sum_{\bar{s}} \mathbb{P}(\bar{\mathbf{x}}, \bar{s}|\bar{p}) = \sum_{\bar{s}} \mathbb{P}(\bar{\mathbf{x}}|\bar{s}, \bar{p}) \cdot \mathbb{P}(\bar{s}|\bar{p}) && (4) \\ &= \sum_{\bar{s}} \prod_{l=1}^{K} \mathbb{P}\left(\bar{\mathbf{x}}_{s_l}^{s_{l+1}-1}|s_l, p_l\right) \cdot \prod_{l=1}^{K} \mathbb{P}(s_l|s_{l-1}, p_l, p_{l-1}) \ , && (5) \end{aligned}$$

where $\bar{\mathbf{x}}_{s_l}^{s_{l+1}-1} = (\mathbf{x}_{s_l}, \ldots, \mathbf{x}_{s_{l+1}-1})$ is the sub-sequence of feature vector constitute the segment, and K is the number of phones (segments), that is, $K = |\bar{p}|$. Segment-based recognition involves finding

$$\bar{p}' = \arg\max_{\bar{p},K} \left[\ \max_{\bar{s}} \ \mathbb{P}(\bar{\mathbf{x}}|\bar{s}, \bar{p}) \ \mathbb{P}(\bar{s}|\bar{p}) \ \mathbb{P}(\bar{p}) \ \right] \ . \qquad (6)$$

The kernel-based model is closely related to the segment model and can be considered as a generalization of it. In the next section we present the kernel-based model and shows that it generalizes the segment model.

3 Kernel-Based Model

Recall that the problem of speech recognition can be stated as the problem of finding the most likely sequence of words $\bar{w}'$ given the acoustic feature vectors $\bar{\mathbf{x}}$. In its logarithm form it can be written as

$$\bar{w}' = \arg\max_{\bar{w}} \; \log \mathbb{P}(\bar{\mathbf{x}}|\bar{w}) \; + \; \log \mathbb{P}(\bar{w}) \; ,$$

where $\mathbb{P}(\bar{\mathbf{x}}|\bar{w})$ is the probability of the acoustic features given the word sequence known as the acoustic model, and $\mathbb{P}(\bar{w})$ is the probability of the sequence of words known as the language model.

The discriminative kernel-based construction for speech recognition is based on a predefined vector feature function $\boldsymbol{\phi} : \mathcal{X}^\star \times \mathcal{V}^\star \rightarrow \mathcal{H}$, where $\mathcal{H}$ is a reproducing kernel Hilbert space (RKHS). Thus, the input of this function is an acoustic representation, $\bar{\mathbf{x}}$, together with a candidate word sequence $\bar{w}$. The feature function returns a vector in $\mathcal{H}$, where, intuitively, each element of the vector represents the confidence that the suggested word sequence $\bar{w}$ is said in the speech signal $\bar{\mathbf{x}}$.

The discriminative kernel-based speech recognizer is not formulated with probabilities, but rather as a dot product of the vector feature function $\boldsymbol{\phi}$ and a weight vector $\boldsymbol{\omega} \in \mathcal{H}$,

$$\bar{w}' = \arg\max_{\bar{w}} \;\; \boldsymbol{\omega} \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{w}) \; ,$$

where $\boldsymbol{\omega} \in \mathcal{H}$ is a weight vector that should be learned. In the same spirit of generative HMM-based speech recognizer, this function can be written as a sum of the acoustic model and the language model,

$$\bar{w}' = \arg\max_{\bar{w}} \;\; \boldsymbol{\omega}^{\mathrm{acoustic}} \cdot \boldsymbol{\phi}^{\mathrm{acoustic}}(\bar{\mathbf{x}}, \bar{w}) \; + \; \boldsymbol{\omega}^{\mathrm{language}} \cdot \boldsymbol{\phi}^{\mathrm{language}}(\bar{w}) \; , \tag{7}$$

where both $\boldsymbol{\omega}^{\mathrm{acoustic}}$ and $\boldsymbol{\omega}^{\mathrm{language}}$ are sub-vectors of the vector $\boldsymbol{\omega}$, i.e., $\boldsymbol{\omega} = (\boldsymbol{\omega}^{\mathrm{acoustic}}, \boldsymbol{\omega}^{\mathrm{language}})$, and similarly $\boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{w}) = (\boldsymbol{\phi}^{\mathrm{acoustic}}(\bar{\mathbf{x}}, \bar{w}), \boldsymbol{\phi}^{\mathrm{language}}(\bar{w}))$. The first term $\boldsymbol{\omega}^{\mathrm{acoustic}} \cdot \boldsymbol{\phi}^{\mathrm{acoustic}}(\bar{\mathbf{x}}, \bar{w})$ in the equation is the *acoustic modeling*, and it assigns a confidence for every candidate word sequence $\bar{w}$ and a given acoustic signal $\bar{\mathbf{x}}$. Since we do not know how to state the feature functions of the acoustic modeling in terms of word sequences explicitly, we state them in terms of the phoneme[1] sequences and phoneme timing sequences. We define the timing (alignment) sequence $\bar{s}$ corresponding to a phoneme sequence $\bar{p}$ as the sequence of start times, where we denote by $s_l \in \mathbb{N}$ the start time of phoneme $p_l \in \mathcal{P}$. Since we do not know the start time sequence while decoding, we search for the best timing sequence, hence

$$\bar{w}' = \arg\max_{\bar{w}} \; \left[\; \max_{\bar{s}} \; \boldsymbol{\omega}^{\mathrm{acoustic}} \cdot \boldsymbol{\phi}^{\mathrm{acoustic}}(\bar{\mathbf{x}}, \bar{p}, \bar{s}) \;\; + \;\; \boldsymbol{\omega}^{\mathrm{language}} \cdot \boldsymbol{\phi}^{\mathrm{language}}(\bar{w}) \; \right] \; , \tag{8}$$

[1]Note that in the kernel-based model we use *phonemes* rather than *phones*.

where we use a lexicon *lex* mapping to generate phonemes from words, $\bar{p} = (lex(w_1), \ldots, lex(w_N))$, and we overload the definition of the acoustic modeling feature function as $\boldsymbol{\phi} : \mathcal{X}^\star \times \mathcal{P}^\star \times \mathbb{N}^\star \rightarrow \mathcal{H}$. The description of the concrete form of the feature function is deferred to Sec. 4.

The second term $\boldsymbol{\omega}^{\text{language}} \cdot \boldsymbol{\phi}^{\text{language}}(\bar{w})$ in Eq. (7) is the *language model*. Traditionally, the language model assigns a probability to a sequence of words by means of a probability distribution. In the discriminative setting, the language model gives confidence to a string of words in a natural language. Collins and his colleagues [18] described discriminative language modeling for a large vocabulary speech recognition task which would be suitable for the above setting. Another way to build a discriminative language model is by using prediction suffix trees (PSTs) [19, 6] as demonstrated in [15].

Comparing the logarithm form of the segment model Eq. (6) with with the kernel-based model Eq. (8), it can be seen that the kernel-based model generalizes the segment model. Specifically, we can decompose the weight vector $\boldsymbol{\omega}^{\text{acoustic}}$ into two sub-vectors, $\boldsymbol{\omega}^{\text{acoustic}} = (\boldsymbol{\omega}_1^{\text{acoustic}}, \boldsymbol{\omega}_2^{\text{acoustic}})$, and similarly decompose the vector feature function as $\boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}, \bar{p}, \bar{s}) = (\boldsymbol{\phi}_1^{\text{acoustic}}(\bar{\mathbf{x}}, \bar{p}, \bar{s}), \boldsymbol{\phi}_2^{\text{acoustic}}(\bar{p}, \bar{s}))$. We get

$$\bar{w}' = \arg\max_{\bar{w}} \Big[\max_{\bar{s}} \; \boldsymbol{\omega}_1^{\text{acoustic}} \cdot \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}, \bar{p}, \bar{s}) \; + \; \boldsymbol{\omega}_2^{\text{acoustic}} \cdot \boldsymbol{\phi}^{\text{acoustic}}(\bar{p}, \bar{s}) \\ + \; \boldsymbol{\omega}^{\text{language}} \cdot \boldsymbol{\phi}^{\text{language}}(\bar{w}) \Big] . \quad (9)$$

Setting each element of the vector $\boldsymbol{\phi}$ to be an indicator function for every probability event in the segment model and each element in $\boldsymbol{\omega}$ to be the probability estimation of the corresponding indicator in $\boldsymbol{\phi}$, we get the segment model.

There are several advantages of the kernel-based model over the segment model. First as we show in Sec. 7.1 the kernel-based model estimated the weight vector as to minimize the word error rate, and the process is guaranteed to converge to a global minima. Moreover, we prove that this estimation of the weight vector leads to a minimization of the word error rate on unseen speech utterance (and not only on the training set). Last, the kernel-based model can be easily transformed into a non-linear model. As we see in the next sections the algorithm for estimating $\boldsymbol{\omega}$ solely depends on the dot product between feature functions $\boldsymbol{\phi}$. Wherever such a dot product is used, it is replaced with the kernel function, which express a non-linear dot product operation[2].

4 Feature Functions

In this section we present the set of feature functions we use. Our construction is based on a set of acoustic modeling feature functions, $\{\phi_j^{\text{acoustic}}\}_{j=1}^n$, which maps an acoustic-phonetic representation of a speech utterance as well as a suggested timing sequence into a vector-space. Our model is also based on a set of language modeling feature functions, $\{\phi_k^{\text{language}}\}_{k=1}^\ell$.

[2]Mercer's theorem [25] states that any continuous, symmetric, positive semi-definite kernel function $\mathcal{K}(u, v)$ can be expressed as a dot product in a high-dimensional space, $\boldsymbol{\phi}(u) \cdot \boldsymbol{\phi}(v)$, where u and v are vectors in some measurable space.

4.1 Acoustic Modeling Feature Functions

First, We introduce a specific set of base functions, which is highly adequate for the acoustic modeling. We utilize seven different feature functions ($n = 7$). Note that the same set of feature functions is also useful in the task of large-margin forced-alignment [11] and keyword spotting [10].

Our first four feature functions aim at capturing transitions between phonemes. These feature functions are the distance between frames of the acoustic signal at both sides of phoneme boundaries as suggested by an alignment sequence $\bar{s}$. The distance measure we employ, denoted by d, is the Euclidean distance between feature vectors. Our underlying assumption is that if two frames, $\mathbf{x}_t$ and $\mathbf{x}_{t'}$, are derived from the same phoneme then the distance $d(\mathbf{x}_t, \mathbf{x}_{t'})$ should be smaller than if the two frames are derived from different phonemes. Formally, our first four feature functions are defined as

$$\phi_j^{\text{acoustic}}(\bar{\mathbf{x}}, \bar{p}, \bar{s}) = \sum_{i=2}^{|\bar{p}|-1} d(\mathbf{x}_{-j+s_i}, \mathbf{x}_{j+s_i}), \quad j \in \{1, 2, 3, 4\} \ . \tag{10}$$

If $\bar{s}$ is the correct timing sequence then distances between frames across the phoneme change points are likely to be large. In contrast, an incorrect phoneme start time sequence is likely to compare frames from the same phoneme, often resulting in small distances.

The fifth feature function we use is built from any frame-wise phoneme classifier, for example the large margin phoneme classifier described in [5]. Formally, for each phoneme event $p \in \mathcal{P}$ and frame $\mathbf{x} \in \mathcal{X}$, there is a confidence, denoted $g_p(\mathbf{x})$, that the phoneme p is pronounced in the frame $\mathbf{x}$. The resulting feature function measures the cumulative confidence of the complete speech signal given the phoneme sequence and their start-times,

$$\phi_5^{\text{acoustic}}(\bar{\mathbf{x}}, \bar{p}, \bar{s}) = \sum_{i=1}^{|\bar{p}|} \sum_{t=s_i}^{s_{i+1}-1} g_{p_i}(\mathbf{x}_t) \ . \tag{11}$$

Our next feature function scores timing sequences based on phoneme durations. Unlike the previous feature functions, the sixth feature function is oblivious to the speech signal itself. It merely examines the length of each phoneme, as suggested by $\bar{s}$, compared to the typical length required to pronounce this phoneme. Formally,

$$\phi_6^{\text{acoustic}}(\bar{\mathbf{x}}, \bar{p}, \bar{s}) = \sum_{i=1}^{|\bar{p}|} \log\ \mathcal{N}(s_{i+1} - s_i; \hat{\mu}_{p_i}\ , \hat{\sigma}_{p_i}) \ \ , \tag{12}$$

where $\mathcal{N}$ is a Normal probability density function with mean $\hat{\mu}_p$ and standard deviation $\hat{\sigma}_p$. In our experiments, we estimated $\hat{\mu}_p$ and $\hat{\sigma}_p$ from the training set.

Our last feature function exploits assumptions on the speaking rate of a speaker. Intuitively, people usually speak in an almost steady rate and therefore a timing sequence in which speech rate is changed abruptly is probably incorrect. Formally, let $\hat{\mu}_p$ be the average length required to pronounce the pth phoneme. We denote by r_i the relative speech rate, $r_i = (s_{i+1} - s_i)/\hat{\mu}_{p_i}$. That is, r_i is the ratio between the actual length of phoneme p_i as suggested by $\bar{s}$ to its average length. The relative speech rate presumably changes slowly over time. In practice the speaking rate ratios often differ from speaker to speaker and within a given utterance. We measure the accomolated local change in the speaking rate as

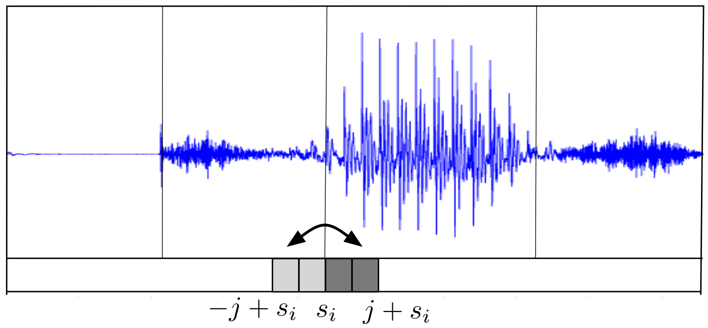

Figure 2: A schematic description of one of the first four feature functions. The depicted base function is the sum of the Euclidean distances between the sum of 2 frames before and after any presumed boundary s_i.

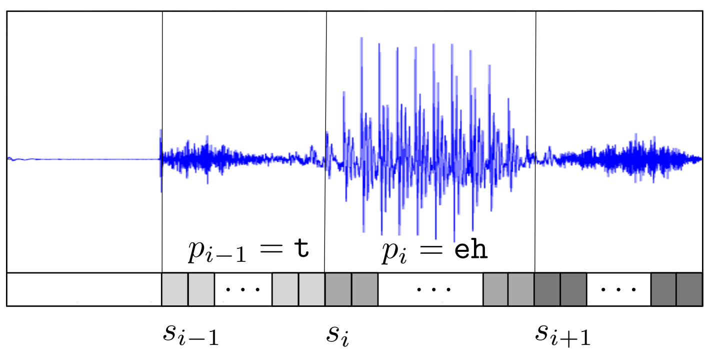

Figure 3: A schematic description of the fifth base function. This function is the sum of all the scores obtained from a large margin classifier, given a sequence of phonemes and a presumed sequence of start-times.

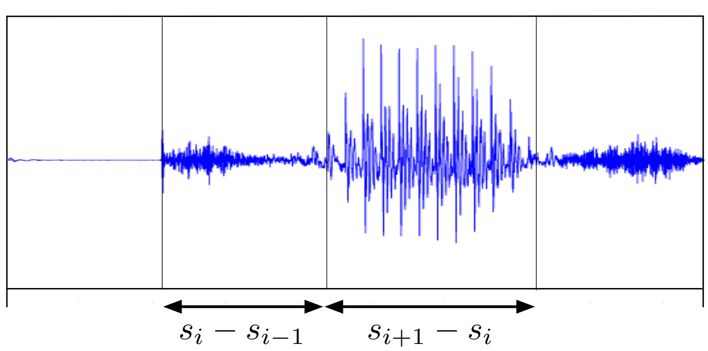

Figure 4: A schematic description of the sixth base function. This function is the sum of the confidences in the duration of each phoneme given its presumed start-time.

$(r_i - r_{i-1})^2$ and we define the feature function ϕ_7^{acoustic} as the local change in the speaking rate,

$$\phi_7^{\text{acoustic}}(\bar{\mathbf{x}}, \bar{p}, \bar{s}) = \sum_{i=2}^{|\bar{p}|} (r_i - r_{i-1})^2 \quad . \tag{13}$$

4.2 Language Modeling Feature Functions

Now we define a set of feature functions for the language modeling. This set of feature functions is taken from [18], and it is given here only for completeness. These features are restricted to be functions over the transcription $\bar{w}$ alone and they track all n-grams up to some length (say $n = 3$), for example:

$$\phi_1^{\text{language}}(\bar{w}) = \text{Number of times } \textit{“the of”} \text{ is seen in } \bar{w}.$$

In practice such feature have to be smoothed according to a methods known in the language modeling literature (See for example [8, 7] and the references therein).

5 The Decoder

Assuming we know the optimal weight vectors $\boldsymbol{\omega}^{\text{acoustic}}$ and $\boldsymbol{\omega}^{\text{language}}$, the maximization over all word sequences in Eq. (8) is a search problem (known also as *the inference problem*). A direct search for the maximizer is not feasible since the number of possible word sequences (and hence the number of possible phoneme and timing sequences) is exponential in the size of the vocabulary, $|\mathcal{V}|$. In this section we provide details about performing the search for kernel-based model. Basically, the search algorithm for the kernel-based model is similar to that used for the HMMs and segment models, using dynamic programming to find the most likely phoneme sequence.

First we derive the domain of all word sequences, which is the feasible region of our search problems. Let us denote the gammer network $\mathcal{G}$ as a directed graph of all possible word sequences (sentences) in the given language. The network $\mathcal{G}$ is also known as a finite-state transducer (FST) [12]. Also denote the lexicon network $\mathcal{L}$ as a set of graphs, where each graph in the set is the phonetic network, generated from the mapping *lex*, corresponds to every word in the vocabulary $\mathcal{V}$. The network $\mathcal{L}$ is also an FST. The FST, generated from the composition of the gammer graph $\mathcal{G}$ and the set of the lexicon network $\mathcal{L}$, namely $\mathcal{L} \circ \mathcal{G}$, is the mapping from phoneme sequences to word sequences. As an optimization step the resulted FST is determinized and minimized. We denote the final FST by $\mathcal{N}$, that is, $\mathcal{N} = min(det(\mathcal{L} \circ \mathcal{G}))$.

Finding the best word sequence is done by efficiently searching the FST network $\mathcal{N}$ for a sequence of phonemes (and a sequence of words) which is optimally aligned to the acoustic signal and hence maximizes Eq. (8). For simplicity, we assume that each base feature function, ϕ_j^{acoustic}, can be decomposed as follows. Let ψ_j be any function from $\mathcal{X}^* \times \mathcal{P}^* \times \mathbb{N}^3$ into the reals, which can be computed in a constant time. That is, ψ_j receives as input the signal, $\bar{\mathbf{x}}$, the sequence of phonemes, $\bar{p}$, and three time points. Additionally, we use the convention $s_0 = 0$ and $s_{|\bar{p}|+1} = T + 1$. Using the above notation, we assume that each ϕ_j^{acoustic} can be

INPUT: speech signal $\bar{\mathbf{x}}$, FST $\mathcal{N}$, weight vector $\boldsymbol{\omega} = (\boldsymbol{\omega}^{\text{acoustic}}, \boldsymbol{\omega}^{\text{language}})$,

maximal length of a phoneme L

INITIALIZE: $\forall (1 \le t \le L),\ D(0,0,t,0) = 0$

RECURSION:

For every state $q \in Q_{\mathcal{N}}$ of the FST $\mathcal{N}$

For every state $q' \in Q_{\mathcal{N}}$ the next state of the FST $\mathcal{N}$

For $t = 1, \ldots, |\bar{\mathbf{x}}|$

For $t' = t - L, \ldots, t-1$

$$D(q,q',t,t') \ = \ \max_{t'-L \le t'' < t'} D(q-1, e-1, t', t'') \ + \ \boldsymbol{\omega}^{\text{acoustic}} \cdot \boldsymbol{\psi}(\bar{\mathbf{x}}, \bar{p}_q, t'', t', t) \ + \ \boldsymbol{\omega}^{\text{language}} \cdot \boldsymbol{\phi}^{\text{language}}(\bar{w}_q)$$

TERMINATION: $D^\star = \max_{t', q_f, q_f} D(q_f, q_f, T, t')$

Figure 5: An efficient procedure for decoding the function given in Eq. (8). Note that $\bar{p}_q$ and $\bar{w}_q$ the phoneme sequence and the word sequence corresponding with state q, respectively.

decomposed to be

$$\phi_j^{\text{acoustic}}(\bar{\mathbf{x}}, \bar{p}, \bar{s}) \ = \ \sum_{i=1}^{|\bar{s}|} \psi_j(\bar{\mathbf{x}}, \bar{p}, s_{i-1}, s_i, s_{i+1}) \quad . \tag{14}$$

Given a state $q \in Q_{\mathcal{N}}$ of FST $\mathcal{N}$, the next state $q' \in Q_{\mathcal{N}}$ and two time indices $t, t' \in \{1, \ldots, T\}$, denote by $D(q,q',t,t')$ the score for the prefix of the sequence $1, \ldots, i$, assuming that their actual start times are $s_1, \ldots, s_i$, where $s_i = t'$ and assuming that $s_{i+1} = t$. This variable can be computed efficiently in a similar fashion to the forward variables calculated by the Viterbi procedure in HMMs (see for instance [17]). The pseudo code for computing $D(q,q',t,t')$ recursively is shown in Fig. 5. The best sequence of actual start times, $\bar{s}'$, is obtained from the algorithm by saving the intermediate values that maximize each expression in the recursion step. The complexity of the algorithm is $\mathcal{O}(|Q_{\mathcal{N}}|\ |\mathcal{P}|\ |\bar{\mathbf{x}}|^3)$. However, in practice, we can use the assumption that the maximal length of an event is bounded, $t - t' \le L$. This assumption reduces the complexity of the algorithm to be $\mathcal{O}(|Q_{\mathcal{N}}|\ |\mathcal{P}|\ |\bar{\mathbf{x}}|\ L^2)$.

6 Large Margin Training

The ultimate goal in speech recognition is usually to minimize the word error rate, that is, the Levenshtein distance (edit distance) between the predicted sequence of words and the correct one. In this section we present an algorithm for learning the weight vector $\boldsymbol{\omega}$, which directly aims at minimizing the Levenshtein distance. Throughout this paper we denote by

$\gamma(\bar{w}, \bar{w}')$ the Levenshtein distance between the predicted word sequence $\bar{w}'$ and the true word sequence $\bar{w}$.

We now describe a large margin approach for learning the weight vectors $\boldsymbol{\omega}^{\text{acoustic}}$ and $\boldsymbol{\omega}^{\text{language}}$, which defines the continuous speech recognizer in Eq. (8), from a training set S of examples. Each example in the training set S is composed of a speech utterance $\bar{\mathbf{x}}$, and its corresponding word sequence $\bar{w}$. Overall we have m examples, that is, $S = \{(\bar{\mathbf{x}}_1, \bar{w}_1), \ldots, (\bar{\mathbf{x}}_m, \bar{w}_m)\}$. We assume that we have a pronunciation lexicon lex, which maps every word to a phoneme sequence. The phonetic transcription of a speech utterance $\bar{\mathbf{x}}$ given its orthographic transcription $\bar{w} = (w_1, \ldots, w_N)$ can be generated by lookup each word in the sequence $\bar{w}$ to find the phoneme sequence $\bar{p} = (lex(w_1), \ldots, lex(w_N))$. We also assume that given a speech utterance $\bar{\mathbf{x}}$ and its corresponding phonetic transcription $\bar{p}$, we have access to the correct time alignment sequence $\bar{s}$ between them. This assumption is actually not restrictive since such an alignment can be inferred relying on an alignment algorithm [11].

Recall that we would like to train the acoustic model and the language model discriminatively as to minimize the Levenshtein distance between the predicted word sequence and the correct one. Similar to the SVM algorithm for binary classification [2, 25], our approach for choosing the weight vector $\boldsymbol{\omega}^{\text{acoustic}}$ and $\boldsymbol{\omega}^{\text{language}}$ is based on the idea of large-margin separation. Theoretically, our approach can be described as a two-step procedure: first, we construct the feature functions $\boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}_i, \bar{s}_i)$ and $\boldsymbol{\phi}^{\text{language}}(\bar{w}_i)$ based on each instance $(\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)$, its phonetic transcription $\bar{p}_i$ and its timing sequence $\bar{s}_i$. We also construct the feature functions $\boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}, \bar{s})$ and $\boldsymbol{\phi}^{\text{language}}(\bar{w})$ based on $\bar{\mathbf{x}}_i$ and every possible word sequence $\bar{w}$, where $\bar{p}$ is the phoneme transcription corresponding to the word sequence $\bar{w}$. Second, we find the weight vectors $\boldsymbol{\omega}^{\text{acoustic}}$ and $\boldsymbol{\omega}^{\text{language}}$, such that the projection of feature functions onto $\boldsymbol{\omega}^{\text{acoustic}}$ and $\boldsymbol{\omega}^{\text{language}}$, ranks the feature functions constructed for the correct word sequence above the feature functions constructed for any other word sequence. Ideally, we would like the following constraint to hold

$$\begin{aligned}
\boldsymbol{\omega}^{\text{acoustic}} \cdot \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}_i, \bar{s}_i) + \boldsymbol{\omega}^{\text{language}} \cdot \boldsymbol{\phi}^{\text{language}}(\bar{w}_i) \; - \\
\max_{\bar{s}} \;\; \boldsymbol{\omega}^{\text{acoustic}} \cdot \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}, \bar{s}) \; - \; \boldsymbol{\omega}^{\text{language}} \cdot \boldsymbol{\phi}^{\text{language}}(\bar{w}) \\
\geq \; \gamma(\bar{w}_i, \bar{w}) \quad \forall \bar{w} \neq \bar{w}_i \; . \quad (15)
\end{aligned}$$

That is, $\boldsymbol{\omega} = (\boldsymbol{\omega}^{\text{acoustic}}, \boldsymbol{\omega}^{\text{language}})$ should rank the correct word sequence above any other word sequence by at least the Levenshtein distance between them. We refer to the difference on the left hand side of Eq. (15) as the *margin* of $\boldsymbol{\omega}$ with respect to the best alignment. Note that if the prediction of $\boldsymbol{\omega}$ is incorrect then the margin is negative. Naturally, if there exists $\boldsymbol{\omega}$ satisfying all the constraints Eq. (15), the margin requirements are also satisfied by multiplying $\boldsymbol{\omega}$ by a large scalar. The SVM algorithm solves this problem by selecting the weights $\boldsymbol{\omega}$ minimizing $\frac{1}{2}\|\boldsymbol{\omega}\|^2 = \frac{1}{2}\|\boldsymbol{\omega}^{\text{acoustic}}\|^2 + \frac{1}{2}\|\boldsymbol{\omega}^{\text{language}}\|^2$ subject to the constraints given in Eq. (15), as it can be shown that the solution with the smallest norm is likely to achieve better generalization [25].

In practice, it might be the case that the constraints given in Eq. (15) cannot be satisfied. To overcome this obstacle, we follow the soft SVM approach [2, 25] and define the following hinge-loss function,

$$\ell(\boldsymbol{\omega}; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) = \max_{\bar{w}} \Big[\gamma(\bar{w}_i, \bar{w}) - \boldsymbol{\omega}^{\text{acoustic}} \cdot \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}_i, \bar{s}_i) - \boldsymbol{\omega}^{\text{language}} \cdot \boldsymbol{\phi}^{\text{language}}(\bar{w}_i) + \max_{\bar{s}} \;\; \boldsymbol{\omega}^{\text{acoustic}} \cdot \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}, \bar{s}) + \boldsymbol{\omega}^{\text{language}} \cdot \boldsymbol{\phi}^{\text{language}}(\bar{w}) \Big]_+ \,, \quad (16)$$

where $[a]_+ = \max\{0, a\}$. The hinge loss measures the maximal violation for any of the constraints given in Eq. (15). The soft SVM approach for our problem is to choose the vector $\boldsymbol{\omega}$ which minimizes the following optimization problem

$$\min_{\boldsymbol{\omega}} \;\; \frac{1}{2}\|\boldsymbol{\omega}^{\text{acoustic}}\|^2 + \frac{1}{2}\|\boldsymbol{\omega}^{\text{language}}\|^2 + C \sum_{i=1}^{m} \ell(\boldsymbol{\omega}; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) \;\; , \quad (17)$$

where the parameter C serves as a complexity-accuracy trade-off parameter: a low value of C favors a simple model, while a large value of C favors a model which solves all training constraints (see [4]). Solving the optimization problem given in Eq. (17) is expensive since it involves a maximization for each training example. Most of the solvers for this problem, like SMO [16], iterate over the whole dataset several times until convergence. In the next section, we propose a slightly different method, which visits each example only once, and is based on our previous work [3].

7 Iterative Algorithm

We now describe a simple iterative algorithm for learning the weight vectors $\boldsymbol{\omega}^{\text{acoustic}}$ and $\boldsymbol{\omega}^{\text{language}}$. The algorithm receives as input a training set $S = \{(\bar{\mathbf{x}}_1, \bar{w}_1, \bar{p}_1, \bar{s}_1), \ldots, (\bar{\mathbf{x}}_m, \bar{w}_m, \bar{p}_m, \bar{s}_m)\}$ of examples. Each example is constituted of a spoken acoustic signal $\bar{\mathbf{x}}_i$, its corresponding word sequence $\bar{w}_i$, phoneme sequence $\bar{p}_i$ and alignment sequence $\bar{s}_i$. At each iteration the algorithm updates $\boldsymbol{\omega}^{\text{acoustic}}$ and $\boldsymbol{\omega}^{\text{language}}$ according to the ith example in S as we now describe. Denote by $\boldsymbol{\omega}_{i-1}^{\text{acoustic}}$ and $\boldsymbol{\omega}_{i-1}^{\text{language}}$ the value of the weight vectors before the ith iteration. Let $\bar{w}'_i$, $\bar{p}'_i$ and $\bar{s}'_i$ be the predicted word sequence, phoneme sequence and timing sequence for the ith example, respectively, according to $\boldsymbol{\omega}_{i-1}^{\text{acoustic}}$ and $\boldsymbol{\omega}_{i-1}^{\text{language}}$,

$$(\bar{w}'_i, \bar{p}'_i, \bar{s}'_i) = \arg\max_{\bar{w}} \;\; \Big[\;\max_{\bar{s}} \;\; \boldsymbol{\omega}_{i-1}^{\text{acoustic}} \cdot \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}, \bar{s}) \;\; + \;\; \boldsymbol{\omega}_{i-1}^{\text{language}} \cdot \boldsymbol{\phi}^{\text{language}}(\bar{w}) \;\Big] \quad (18)$$

Denote by $\gamma(\bar{w}_i, \bar{w}'_i)$ the Levenshtein distance between the correct word sequence $\bar{w}_i$ and the predicted word sequence $\bar{w}'_i$. Also denote by $\Delta\boldsymbol{\phi}_i = (\Delta\boldsymbol{\phi}_i^{\text{acoustic}}, \Delta\boldsymbol{\phi}_i^{\text{language}})$ the difference between the feature function of the correct word sequence minus the feature function of the predicted word sequence as

$$\Delta\boldsymbol{\phi}_i^{\text{acoustic}} = \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}_i, \bar{s}_i) - \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}', \bar{s}') \;,$$

and

$$\Delta\boldsymbol{\phi}_i^{\text{language}} = \boldsymbol{\phi}^{\text{language}}(\bar{w}_i) - \boldsymbol{\phi}^{\text{language}}(\bar{w}') \;.$$

We set the next weight vectors $\boldsymbol{\omega}_i^{\text{acoustic}}$ and $\boldsymbol{\omega}_i^{\text{language}}$ to be the minimizer of the following optimization problem,

$$\begin{aligned} \min_{\boldsymbol{\omega}\in\mathcal{H},\xi\geq 0} \quad & \frac{1}{2}\|\boldsymbol{\omega}^{\text{acoustic}} - \boldsymbol{\omega}_{i-1}^{\text{acoustic}}\|^2 + \frac{1}{2}\|\boldsymbol{\omega}^{\text{language}} - \boldsymbol{\omega}_{i-1}^{\text{language}}\|^2 \ + \ C\xi \\ \text{s.t.} \quad & \boldsymbol{\omega}^{\text{acoustic}} \cdot \Delta\boldsymbol{\phi}_i^{\text{acoustic}} + \boldsymbol{\omega}^{\text{language}} \cdot \Delta\boldsymbol{\phi}_i^{\text{language}} \ \geq \ \gamma(\bar{w}_i, \bar{w}'_i) - \xi \ , \end{aligned} \tag{19}$$

where C serves as a complexity-accuracy trade-off parameter (see [3]) and ξ is a non-negative slack variable, which indicates the loss of the ith example. Intuitively, we would like to minimize the loss of the current example, i.e., the slack variable ξ, while keeping the weight vector $\boldsymbol{\omega}$ as close as possible to the previous weight vector $\boldsymbol{\omega}_{i-1}$. The constraint makes the projection of the feature function s of the correct word sequence onto $\boldsymbol{\omega}$ higher than the projection of the feature functions of any other word sequence onto $\boldsymbol{\omega}$ by at least $\gamma(\bar{w}_i, \bar{w}'_i)$. It can be shown (see [3]) that the solution to the above optimization problem is

$$\begin{aligned} \boldsymbol{\omega}_i^{\text{acoustic}} &= \boldsymbol{\omega}_{i-1}^{\text{acoustic}} + \alpha_i \Delta\boldsymbol{\phi}_i^{\text{acoustic}} \\ \boldsymbol{\omega}_i^{\text{language}} &= \boldsymbol{\omega}_{i-1}^{\text{language}} + \alpha_i \Delta\boldsymbol{\phi}_i^{\text{language}} \ . \end{aligned} \tag{20}$$

The value of the scalar α_i is based on the difference $\Delta\boldsymbol{\phi}_i$, the previous weight vector $\boldsymbol{\omega}_{i-1}$, and a parameter C. Formally,

$$\alpha_i = \min\left\{ C,\ \frac{\left[\gamma(\bar{w}_i, \bar{w}'_i) - \boldsymbol{\omega}_{i-1}^{\text{acoustic}} \cdot \Delta\boldsymbol{\phi}_i^{\text{acoustic}} - \boldsymbol{\omega}_{i-1}^{\text{language}} \cdot \Delta\boldsymbol{\phi}_i^{\text{language}}\right]_+}{\|\Delta\boldsymbol{\phi}_i^{\text{acoustic}}\|^2 + \|\Delta\boldsymbol{\phi}_i^{\text{language}}\|^2} \right\} \ . \tag{21}$$

A pseudo-code of our algorithm is given in Fig. 6.

Under some mild technical conditions, it can be shown that the cumulative Levenshtein distance of an iterative procedure , $\sum_{i=1}^{m} \gamma(\bar{w}_i, \bar{w}'_i)$, is likely to be small. Moreover, it can be shown [1] that if the cumulative Levenshtein distance of the iterative procedure is small, there exists among the vectors $\{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m\}$ at least one weight vector which attains small averaged Levenshtein distance on unseen examples as well. A detailed analysis of the algorithm is given in Appendix.

7.1 Non-linear Recognition Function

We extend the family of linear word sequence recognizers given to non-linear recognition functions. This extension is based on Mercer kernels often used in SVM algorithms [25]. Recall that the update rule of the algorithm is $\boldsymbol{\omega}_i = \boldsymbol{\omega}_{i-1} + \alpha_i \Delta\boldsymbol{\phi}_i$ and that the initial weight vector is $\boldsymbol{\omega}_0 = \mathbf{0}$. Thus, $\boldsymbol{\omega}_i$ can be rewritten as, $\boldsymbol{\omega}_i = \sum_{j=1}^{i} \alpha_j \Delta\boldsymbol{\phi}_j$ and f can be rewritten as

$$f(\bar{\mathbf{x}}) = \operatorname*{argmax}_{\bar{p}} \ \max_{\bar{s}} \ \sum_{j=1}^{i} \alpha_j \Big(\Delta\boldsymbol{\phi}_j \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{w})\Big) \ . \tag{22}$$

By substituting the definition of $\Delta\boldsymbol{\phi}_j$ and replacing the inner-product in Eq. (22) with a general kernel operator $\mathcal{K}(\cdot,\cdot)$ that satisfies Mercer's conditions [25], we obtain a non-linear phoneme recognition function,

$$f(\bar{\mathbf{x}}) = \operatorname*{argmax}_{\bar{p}} \ \max_{\bar{s}} \ \sum_{j=1}^{i} \alpha_j \Big(\mathcal{K}(\bar{\mathbf{x}}_j, \bar{w}_j; \bar{\mathbf{x}}, \bar{w}) - \mathcal{K}(\bar{\mathbf{x}}_j, \bar{w}'_j; \bar{\mathbf{x}}, \bar{w})\Big) \ . \tag{23}$$

Input: training set $S = \{(\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)\}_{i=1}^{m}$; validation set $S_{\text{val}} = \{(\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)\}_{i=1}^{m_{\text{val}}}$; parameter C

Initialize: $\boldsymbol{\omega}_0 = (\boldsymbol{\omega}_0^{\text{acoustic}}, \boldsymbol{\omega}_0^{\text{language}}) = \mathbf{0}$

For $i = 1, \ldots, m$

Predict: $(\bar{w}', \bar{p}', \bar{s}') = \arg\max_{\bar{w}} \left[\max_{\bar{s}} \; \boldsymbol{\omega}_{i-1}^{\text{acoustic}} \cdot \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}, \bar{s}) \; + \; \boldsymbol{\omega}_{i-1}^{\text{language}} \cdot \boldsymbol{\phi}^{\text{language}}(\bar{w}) \; \right]$

Set: $\Delta\boldsymbol{\phi}_i^{\text{acoustic}} = \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}_i, \bar{s}_i) - \boldsymbol{\phi}^{\text{acoustic}}(\bar{\mathbf{x}}_i, \bar{p}', \bar{s}')$

$\Delta\boldsymbol{\phi}_i^{\text{language}} = \boldsymbol{\phi}^{\text{language}}(\bar{w}_i) - \boldsymbol{\phi}^{\text{language}}(\bar{w}')$.

Set: $\ell(\boldsymbol{\omega}_{i-1}; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) = \max_{\bar{w}} \left[\gamma(\bar{w}_i, \bar{w}') - \boldsymbol{\omega}_{i-1}^{\text{acoustic}} \cdot \Delta\boldsymbol{\phi}_i^{\text{acoustic}} - \boldsymbol{\omega}_{i-1}^{\text{language}} \cdot \Delta\boldsymbol{\phi}_i^{\text{language}} \right]_+$

If $\ell(\boldsymbol{\omega}_{i-1}; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) > 0$

Set: $\alpha_i = \min \left\{ C \; , \; \dfrac{\ell(\boldsymbol{\omega}_{i-1}; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i))}{\|\Delta\boldsymbol{\phi}_i^{\text{acoustic}}\|^2 + \|\Delta\boldsymbol{\phi}_i^{\text{language}}\|^2} \right\}$

Update: $\boldsymbol{\omega}_i^{\text{acoustic}} = \boldsymbol{\omega}_{i-1}^{\text{acoustic}} + \alpha_i \Delta\boldsymbol{\phi}_i^{\text{acoustic}}$

$\boldsymbol{\omega}_i^{\text{language}} = \boldsymbol{\omega}_{i-1}^{\text{language}} + \alpha_i \Delta\boldsymbol{\phi}_i^{\text{language}}$.

Output: The weight vector $\boldsymbol{\omega}^*$ which achieves best performance on a validation set S_{val}:

$$\boldsymbol{\omega}^* = \underset{\boldsymbol{\omega} \in \{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m\}}{\text{argmin}} \sum_{j=1}^{m_{\text{val}}} \gamma\Big(\bar{w}_j, f(\bar{\mathbf{x}}_j)\Big)$$

Figure 6: An iterative algorithm for phoneme recognition.

It is easy to verify that the definition of α_i given in Eq. (21) can also be rewritten using the kernel operator.

7.2 Complexity

To conclude this section we discuss the global complexity of our proposed method. In the training phase, our algorithm performs m iterations, one iteration per each training example. At each iteration the algorithm evaluates the recognition function once, updates the recognition function, if needed, and evaluates the new recognition function on a validation set of size m_{val}. Each evaluation of the recognition function takes an order of $\mathcal{O}(|Q_{\mathcal{N}}| \; |\mathcal{P}| \; |\bar{\mathbf{x}}| \; L^2)$ operations. Therefore the total complexity of our method becomes $\mathcal{O}(m \; m_{\text{val}} \; |Q_{\mathcal{N}}| \; |\mathcal{P}| \; |\bar{\mathbf{x}}| \; L^2)$. Finally, we compare the complexity of our method to the complexity of other algorithms which directly solve the SVM optimization problem given in Eq. (19). The algorithm given in [23] is based on the SMO algorithm for solving SVM problems. While there is no direct complexity analysis for this algorithm, in practice it usually required at least m^2 iterations which results in a total complexity of the order $\mathcal{O}(m^2 \; |Q_{\mathcal{N}}| \; |\mathcal{P}| \; |\bar{\mathbf{x}}| \; L^2)$. The complexity of the algorithm presented in [24] depends on the choice of several parameters. For reasonable choice of these parameters the total complexity is also of the order $\mathcal{O}(m^2 \; |Q_{\mathcal{N}}| \; |\mathcal{P}| \; |\bar{\mathbf{x}}| \; L^2)$.

Acknowledgments

We are in debt to Philip Garner for his clarifications and simplifications of the HMM-based decoder. This research was supported by the DIRAC project[3].

References

[1] N. Cesa-Bianchi, A. Conconi, and C. Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, September 2004.

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, September 1995.

[3] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive aggressive algorithms. *Journal of Machine Learning Research*, 7, 2006.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[5] O. Dekel, J. Keshet, and Y. Singer. Online algorithm for hierarchical phoneme classification. In *Workshop on Multimodal Interaction and Related Machine Learning Algorithms; Lecture Notes in Computer Science*, pages 146–159. Springer-Verlag, 2004.

[6] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The power of selective memory: Self-bounded learning of prediction suffix trees. In *Advances in Neural Information Processing Systems 17*, 2004.

[7] F. Jelinek. *Statistical Methds for Speech Recognition*. The MIT Press, 1998.

[8] F. Jelinek, R.L. Mercer, and S. Roukos. Principles of lexical languge modeling for speech reconition. In S. Furui and M.M. Sondhi, editors, *Advances in Speech Signal Processing*. Marcel Decker, 1991.

[9] J. Keshet, D. Chazan, and B.-Z. Bobrovsky. Plosive spotting with margin classifiers. In *Proceedings of the Seventh European Conference on Speech Communication and Technology*, pages 1637–1640, 2001.

[10] J. Keshet, D. Grangier, and S. Bengio. Discriminative keyword spotting. In *Proc. of Workshop on Non-Linear Speech Processsing*, 2007.

[11] J. Keshet, S. Shalev-Shwartz, Y. Singer, and D. Chazan. Phoneme alignment based on discriminative learning. In *Interspeech*, 2005.

[12] M. Mohri, F.C.N. Pereira, and M. Riley. Speech recognition with weighted finite-state transducers. In J. Benesty, M.M. Sondhi, and Y. Huang, editors, *Springer Handbook of Speech Processing*. Springer, 2007.

[3] http://www.diracproject.org/

[13] M. Ostendorf. From hmm's to segment models: Stochastic modeling for CSR. In C.-H. Lee, F.K. Soong, and K.K. Plaiwal, editors, *Automatic Speech and Speaker Recognition - Advanced Topics*. Kluwer Academic Publishers, 1996.

[14] M. Ostendorf, V.V. Digalakis, and O.A. Kimball. From hmm's to segment models: A unified view of stochastic modeling for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(5), Sept 1996.

[15] F. Pereira, Y. Singer, and N. Tishby. Beyond word N-grams. In *Proceedings of the Third Workshop on Very Large Corpora,*, 1995.

[16] J. C. Platt. Fast training of Support Vector Machines using sequential minimal optimization. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1998.

[17] L. Rabiner and B.H. Juang. *Fundamentals of Speech Recognition*. Prentice Hall, 1993.

[18] B. Roark, M. Saraclar, and M. Collins. Discriminative n-gram language modeling. *Computer Speech and Language*, 21:373–392, 2007.

[19] D. Ron, Y. Singer, and N. Tishby. The power of amnesia: learning probabilistic automata with variable memory length. *Machine Learning*, 25(2):117–150, 1996.

[20] J. Salomon, S. King, and M. Osborne. Framewise phone classification using support vector machines. In *Proceedings of the Seventh International Conference on Spoken Language Processing*, pages 2645–2648, 2002.

[21] S. Shalev-Shwartz, J. Keshet, and Y. Singer. Learning to align polyphonic music. In *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004.

[22] S. Shalev-Shwartz and Y. Singer. Online learning meets optimization in the dual. In *Proceedings of the Nineteenth Annual Conference on Computational Learning Theory*, 2006.

[23] B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *Advances in Neural Information Processing Systems 17*, 2003.

[24] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[25] V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.

[26] S. Young. A review of large-vocabulary continuous speech recognition. *IEEE Signal Processing Mag.*, pages 45–57, September 1996.

Appendix

We now analyze our iterative algorithm presented in Sec. 7 and in Fig. 6. Our first theorem shows that under some mild technical conditions, the cumulative Levenshtein distance of the iterative procedure, $\sum_{i=1}^{m} \gamma(\bar{w}_i, \bar{w}'_i)$, is likely to be small.

Theorem 1.: *Let* $S = \{(\bar{\mathbf{x}}_1, \bar{w}_1), \ldots, (\bar{\mathbf{x}}_m, \bar{w}_m)\}$ *be a set of training examples and assume that for all* i, $\bar{w}'$ *we have that* $\|\boldsymbol{\phi}(\bar{\mathbf{x}}_i, \bar{w}')\| \leq 1/2$. *Assume there exists a weight vector* $\boldsymbol{\omega}^\star$ *that satisfies*

$$\boldsymbol{\omega}^\star \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}_i, \bar{w}_i) - \boldsymbol{\omega}^\star \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}_i, \bar{w}) \geq \gamma(\bar{w}_i, \bar{w}')$$

for all $1 \leq i \leq m$ *and* $\bar{w}'$. *Let* $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m$ *be the sequence of weight vectors obtained by the algorithm in Fig. 6 given the training set* S. *Let* $\ell(\boldsymbol{\omega}_i; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i))$ *be defined as in Eq. (16). Then,*

$$\frac{1}{m}\sum_{i=1}^{m} \ell(\boldsymbol{\omega}_i; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) \;\leq\; \frac{1}{m}\sum_{i=1}^{m} \ell(\boldsymbol{\omega}^\star; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) \;+\; \frac{1}{C\,m}\,\|\boldsymbol{\omega}^\star\|^2 + \frac{1}{2}\,C \quad . \tag{24}$$

In particular, if $C = 1/\sqrt{m}$ *then,*

$$\frac{1}{m}\sum_{i=1}^{m} \ell(\boldsymbol{\omega}_i; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) \;\leq\; \frac{1}{m}\sum_{i=1}^{m} \ell(\boldsymbol{\omega}^\star; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) \;+\; \frac{1}{\sqrt{m}}\left(\|\boldsymbol{\omega}^\star\|^2 + \frac{1}{2}\right) \quad . \tag{25}$$

Proof. Our proof relies on Thm. 2 in [22]. We first construct a sequence of binary classification examples, $(\Delta\boldsymbol{\phi}_1, +1), \ldots, (\Delta\boldsymbol{\phi}_m, +1)$. For all i and for all $\boldsymbol{\omega} \in \mathbb{R}^n$, define the following classification hinge-loss,

$$\ell^c_i(\boldsymbol{\omega}) \;=\; \max\{\gamma(\bar{w}_i, \bar{w}'_i) - \boldsymbol{\omega} \cdot \Delta\boldsymbol{\phi}_i \,,\, 0\} \quad .$$

Thm. 2 in [22] implies that the following bound holds for all $\boldsymbol{\omega} \in \mathbb{R}^n$,

$$\sum_{i=1}^{m} \mu(\ell^c_i(\boldsymbol{\omega}_i)) \;\leq\; \frac{1}{C}\|\boldsymbol{\omega}\|^2 + \sum_{i=1}^{m} \ell^c_i(\boldsymbol{\omega}) \quad , \tag{26}$$

where,

$$\mu(a) \;=\; \frac{1}{C}\left(\min\{a, C\}\left(a - \frac{1}{2}\min\{a, C\}\right)\right) \quad . \tag{27}$$

Let $\boldsymbol{\omega}^\star$ denote the optimum of the recognition problem given by Eq. (17). The bound of Eq. (26) holds for any $\boldsymbol{\omega}$ and in particular for the optimal solution $\boldsymbol{\omega}^\star$. Furthermore, the definition of ℓ^c_i implies that $\ell^c_i(\boldsymbol{\omega}^\star) \leq \ell(\boldsymbol{\omega}^\star; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i))$ and $\ell^c_i(\boldsymbol{\omega}_i) = \ell(\boldsymbol{\omega}_i; (\bar{\mathbf{x}}_i, \bar{p}_i, \bar{s}_i))$ for all i. Using the latter two facts in Eq. (26) gives that,

$$\sum_{i=1}^{m} \mu(\ell(\boldsymbol{\omega}_i; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i))) \;\leq\; \frac{1}{C}\|\boldsymbol{\omega}^\star\|^2 + \quad \sum_{i=1}^{m} \ell(\boldsymbol{\omega}^\star; (\bar{\mathbf{x}}_i, \bar{w}_i\bar{p}_i, \bar{s}_i)) \quad . \tag{28}$$

By definition, the function μ is bounded below by a linear function, that is, for any $a > 0$,

$$\mu(a) \geq a - \frac{1}{2}\,C \quad .$$

Using the lower bound with the argument $\ell(\boldsymbol{\omega}_i; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i))$ and summing over i we obtain,

$$\sum_{i=1}^{m} \ell(\boldsymbol{\omega}_i; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) - \frac{1}{2}Cm \leq \sum_{i=1}^{m} \mu(\ell(\boldsymbol{\omega}_i; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i))) \ .$$

Combining the above inequality with Eq. (28) and rearranging terms gives the bound stated in the theorem and concludes our proof. □

The loss bound of Theorem 1 can be translated into a bound on the Levenshtein distance error as follows. Note that the hinge-loss defined by Eq. (16) is greater than $\gamma(\bar{p}_i, \bar{p}'_i)$,

$$\gamma(\bar{p}_i, \bar{p}') \leq \ell(\boldsymbol{\omega}_i; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)).$$

Therefore, we get the following corollary.

Corollary 2.: *Under the conditions of Theorem 1 the following bound on the cumulative Levenshtein distance holds,*

$$\sum_{i=1}^{m} \gamma(\bar{w}_i, \bar{w}'_i) \ \leq \ \frac{1}{m} \sum_{i=1}^{m} \ell(\boldsymbol{\omega}^\star; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) \ + \ \frac{1}{\sqrt{m}} \left(\|\boldsymbol{\omega}^\star\|^2 + \frac{1}{2} \right) \ . \tag{29}$$

The next theorem tells us that if the cumulative Levenshtein distance of the iterative procedure is small, there exists at least one weight vector among the vectors $\{\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m\}$ which attains small averaged Levenshtein distance on unseen examples as well. To find this weight vector we simply calculate the averaged Levenshtein distance attained by each of the weight vectors on a validation set. The average Levenshtein distance $\mathbb{E}\left[\gamma(\bar{w}, \bar{w}')\right]$ in defined as the expectation of the Levenshtein distance between the true word sequence and predicted word sequence when the expectation is taken with respect to an unkown distribution D over the domain of the examples, $\mathcal{X}^* \times \mathcal{V}^*$.

Theorem 3.: *Under the same conditions of Theorem 1. Assume that the training set S and the validation set S_{val} are both sampled i.i.d. from a distribution D. Denote by m_{val} the size of the validation set. Assume in addition that $\gamma(\bar{w}, \bar{w}') \leq 1$ for all $\bar{w}$ and $\bar{w}'$. Then, with probability of at least $1 - \delta$ we have that,*

$$\mathbb{E}\left[\gamma(\bar{w}, \bar{w}')\right] \ \leq \ \frac{1}{m} \sum_{i=1}^{m} \ell(\boldsymbol{\omega}^\star; (\bar{\mathbf{x}}_i, \bar{w}_i, \bar{p}_i, \bar{s}_i)) + \frac{\|\boldsymbol{\omega}^\star\|^2 + \frac{1}{2} + \sqrt{2\ln(2/\delta)}}{\sqrt{m}} + \frac{\sqrt{2\ln(2m/\delta)}}{\sqrt{m_{\text{val}}}} \ . \tag{30}$$

Proof. Denote by $f_1, \ldots, f_m$ the alignment prediction functions corresponding to the weight vectors $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_m$ that are found by the iterative algorithm, that is,

$$f_i(\bar{\mathbf{x}}) = \arg\max_{\bar{w}} \ \ \boldsymbol{\omega}_i \cdot \boldsymbol{\phi}(\bar{\mathbf{x}}, \bar{w}) \ .$$

Proposition 1 in [1] implies that with probability of at least $1-\delta_1$ the following bound holds,

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[\gamma(\bar{w}, f_i(\bar{\mathbf{x}}))\right] \;\leq\; \frac{1}{m}\sum_{i=1}^{m}\gamma(\bar{w}_i, f_i(\bar{\mathbf{x}}_i)) + \frac{\sqrt{2\ln(1/\delta_1)}}{\sqrt{m}} \quad .$$

By definition, the hinge-loss $\ell(\boldsymbol{\omega}_i;(\bar{\mathbf{x}}_i,\bar{w}_i,\bar{p}_i,\bar{s}_i))$ bounds from above the loss $\gamma(\bar{w}_i, f_i(\bar{\mathbf{x}}_i))$. Combining this fact with Theorem 1 we obtain that,

$$\frac{1}{m}\sum_{i=1}^{m}\mathbb{E}\left[\gamma(\bar{w}, f_i(\bar{\mathbf{x}}))\right] \;\leq\; \frac{1}{m}\sum_{i=1}^{m}\ell(\boldsymbol{\omega}^\star;(\bar{\mathbf{x}}_i,\bar{w}_i,\bar{p}_i,\bar{s}_i)) + \frac{\|\boldsymbol{\omega}^\star\|^2+\frac{1}{2}+\sqrt{2\ln\left(1/\delta_1\right)}}{\sqrt{m}} \quad . \tag{31}$$

The left-hand side of the above inequality upper bounds $\mathbb{E}\left[\gamma(\bar{w}, f_b(\bar{\mathbf{x}}))\right]$, where $b = \arg\min_i \mathbb{E}\left[\gamma(\bar{w}, f_i(\bar{\mathbf{x}}))\right]$. Therefore, among the finite set of alignment functions, $F = \{f_1,\ldots,f_m\}$, there exists at least one recognition function (for instance the function f_b) whose true risk is bounded above by the right hand side of Eq. (31). Recall that the output of our algorithm is the recognition function $f_{\boldsymbol{\omega}} \in F$, which minimizes the Levenshtein distance over the validation set S_{val}. Applying Hoeffding inequality together with the union bound over F we conclude that with probability of at least $1-\delta_2$,

$$\mathbb{E}\left[\gamma(\bar{w}, f_{\boldsymbol{\omega}}(\bar{\mathbf{x}}))\right] \leq \mathbb{E}\left[\gamma(\bar{w}, f_b(\bar{\mathbf{x}}))\right] + \sqrt{\frac{2\ln\left(m/\delta_2\right)}{m_v}} \quad ,$$

where to remind the reader $m_{\text{val}} = |S_{\text{val}}|$. We have therefore shown that with probability of at least $1-\delta_1-\delta_2$ the following inequality holds,

$$\mathbb{E}\left[\gamma(\bar{w}, f_{\boldsymbol{\omega}}(\bar{\mathbf{x}}))\right] \;\leq\; \frac{1}{m}\sum_{i=1}^{m}\ell(\boldsymbol{\omega}^\star;(\bar{\mathbf{x}}_i,\bar{p}_i,\bar{s}_i)) \;+ \frac{\|\boldsymbol{\omega}^\star\|^2+\frac{1}{2}+\sqrt{2\ln(1/\delta_1)}}{\sqrt{m}} + \frac{\sqrt{2\ln(m/\delta_2)}}{\sqrt{m_{\text{val}}}} \quad .$$

Setting $\delta_1 = \delta_2 = \delta/2$ concludes our proof. $\square$