Chapter 8

AUTOMATIC SPEECH PROCESSING BY INFERENCE IN GENERATIVE MODELS

Sam T. Roweis
Department of Computer Science, University of Toronto
roweis@cs.toronto.edu

Abstract Normally, algorithms which process speech signals to estimate quantities of interest (e.g. pitch) or perform various complex operations (e.g. denoising) are designed directly, by experts, to computing the final output from the input representation using a series of processing steps. Another approach is to build a *probabilistic generative model* of the input (waveform or short time spectra) in which the quantities of eventual interest are represented as hidden or latent variables. Estimation then takes the form of *statistical inference* in these models, for which well known algorithms exist. The model parameters themselves can be learned from example inputs. Often, the results of such inference can be extremely informative even when the trained model does not capture all of the complexity in the original input data.

In this chapter, we will give several examples of this paradigm, showing how inference in very simple generative models can be used to perform surprisingly complex speech processing tasks including denoising, source separation, pitch tracking, timescale modification and estimation of articulatory movements from audio.

Keywords: probabilistic generative models, inference, time scale modification, pitch tracking, F0 estimation, denoising, source separation, articulatory modeling

Say you want to perform some complex speech processing task. How should you develop the algorithm that you eventually use? Traditionally, you combine inspiration, carefully examination of previous work, and arduous trial-and-error to invent a sequence of operations to apply to the input waveform or short-time spectral representation. But there

is another approach: dream up a "generative model" –a probabilistic machine which outputs data in the same form as your data–in which the key quantities that you would eventually like to compute appear as hidden (latent) variables. Now perform *inference* in this model, estimating the hidden quantities. In doing so, the rules of probability will derive for you, automatically, a signal processing algorithm. While inference is well known to the speech community as a decoding step for HMMs, exactly the same type of calculation can be performed in many other models not related to recognition.

This chapter explores the use of inference in three separate models, and shows how it can be used to perform surprisingly complex speech processing tasks including denoising, source separation, pitch tracking, timescale modification and estimation of articulatory movements from audio.

1. A Factorial-MAX Model of Log Spectrograms

In this section, we review the astonishing max approximation to log spectrograms of mixtures, show why this motivates a "refiltering" approach to separation and denoising, and then describe how the process of inference in factorial probabilistic models performs a computation useful for deriving the masking signals needed in refiltering. A particularly simple model, Factorial-Max Vector Quantization (MAXVQ), is introduced along with a branch-and-bound technique for efficient exact inference and applied to both additive denoising and monaural separation. Our approach represents a return to the ideas of Ephraim, Varga and Moore (Varga and Moore, 1990) but applied to auditory scene analysis rather than to speech recognition.

Sparsity & Redundancy in Spectrograms

The *sparse* nature of the speech code across time and frequency is one of the key features exploited by many successful speech processing algorithms. Roughly speaking, most low noise narrow frequency bands carry substantial energy only a small fraction of the time and thus it is rare that two independent sources inject large amounts of energy into the same subband at the same time. (Figure 1.1b shows a plot of the relative energy of two simultaneous talkers in a narrow subband; most of the time at least one of the two sources shows negligible power.)

The speech code is also *redundant* across time-frequency. Different frequency bands carry, to a certain extent, independent information and so if information in some bands is suppressed or masked, even for significant durations, other bands can fill in. (A similar effect occurs over

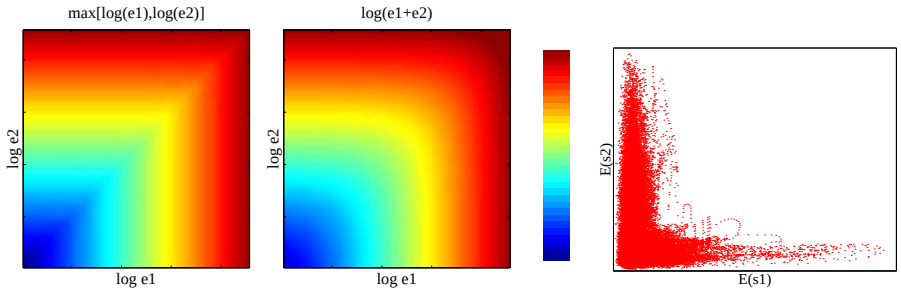

Figure 1.1. (left) Relationship between log of sum and max of logs; each function's value is shown using the color scale indicated in the middle. Significant differences occur only when $e_1 \approx e_2$ and both are large. (right) Relative energy of two sources in a single subband; few points appear on the diagonal.

time: if brief sections of the signal are obscured, even across all bands, the speech is still intelligible; while also useful, we do not exploit this here.) This is partly why humans perform so well on many monaural speech separation and denoising tasks. When we solve the cocktail party problem or recognize degraded speech, we are doing structural analysis, or a kind of "perceptual grouping" on the incoming sound. There is substantial evidence that the appropriate subparts of an audio signal for use in grouping may be narrow frequency bands over short times.

The Log-Max Approximation. When two clean speech signals are mixed additively in the time domain, what is the relationship between the individual log spectrograms of the sources and the log spectrogram of the mixture? Unless the sources are highly dependent (synchronized), the log spectrogram of the mixture is almost exactly the *maximum* of the individual log spectrograms, with the maximum operating over small time-frequency regions (fig. 1.2). This amazing fact, first noted by Roger Moore in 1983, comes from the fact that unless e_1 and e_2 are both large and almost equal, $\log(e_1 + e_2) \approx \max(\log e_1, \log e_2)$ (fig. 1.1a).

The sparsity of the speech code is what makes this approximation useful in practice, since the approximation only breaks down when two sources put a large amount of energy into the same narrow frequency band at the same time, which rarely occurs.

Masking and Refiltering. To exploit the redundancy of the speech code, we will focus on narrow frequency bands over short times and try to group these "subparts" of the signal together, based on whether they belong to the same source or not. If we can collect enough parts that

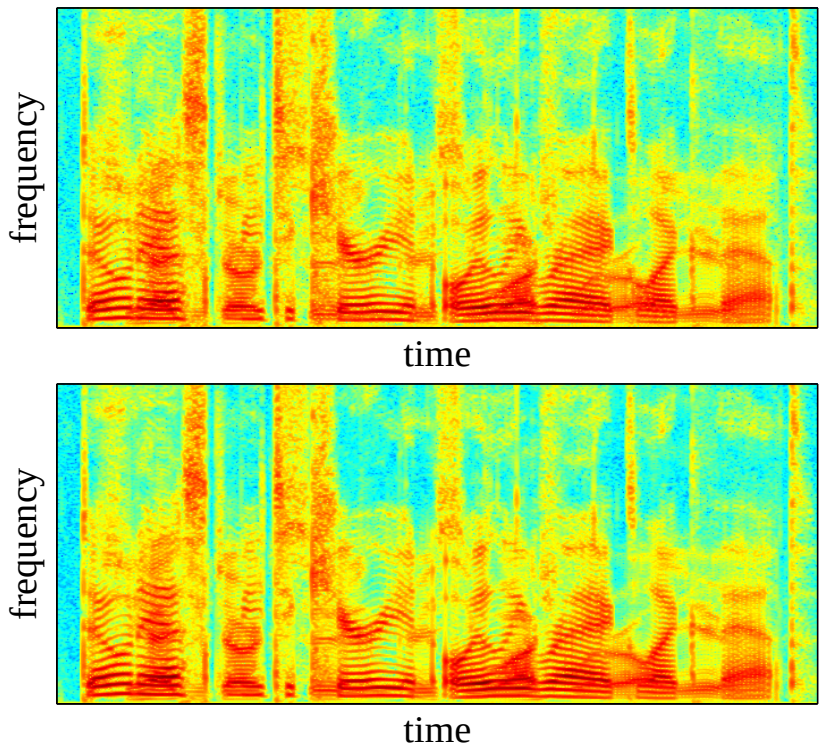

Figure 1.2. (top) Log spectrogram of a mixture of two sources. (bottom) Elementwise maximum (within each time-frequency bin) of log spectrograms of original sources. The two spectrograms are almost identical, although the bottom one is an approximation of what the top one ought to look like based on a very simple combination model.

we are confident belong together, we can discard the rest of the signal and recover the original source based only on the grouped parts.

To generate these parts computationally, we can perform multiband analysis – break the original speech signal $y(t)$ into many subband signals $b_i(t)$ each filtered to contain only energy from a small portion of the spectrum. The results of such an analysis are often displayed as a spectrogram which shows log energy (using color or grayscale) as a function of time and frequency. (Think of a spectrogram like a musical score in which the color or grey level of the each note tells you how hard to hit the piano key.)

The basic idea of *refiltering* (Roweis, 2001; Green et al., 2001) is to separate or denoise sources by selectively reweighting the $b_i(t)$ obtained from multiband analysis of the original mixed or corrupted recording. Crucially, unlike in unmixing algorithms, the reweighting is not constant

over time; it is controlled by a set of *masking signals*. Given a set of masking signals, denoted $\alpha_i(t)$, a clean source $s(t)$ can be recovered by modulating the corresponding subband signals from the original input and summing:

$$\underbrace{s(t)}_{\text{estimated source}} = \overbrace{\alpha_1(t)}^{\text{mask 1}} \underbrace{b_1(t)}_{\text{sub-band 1}} + \ldots + \overbrace{\alpha_K(t)}^{\text{mask K}} \underbrace{b_K(t)}_{\text{sub-band K}} \tag{1.1}$$

The $\alpha_i(t)$ are gain knobs on each subband that we can twist over time to bring bands in and out of the source as needed. This performs masking on the original spectrogram.[1] This approach, illustrated in figure 1.3, forms the basis of many CASA systems(Green et al., 2001; Brown and Cooke, 1994). The basic intuition is to "gate in" subbands deemed to have high signal to noise and to be part of the source we are trying to separate and "gate out" subbands when they are deemed to be noisy or part of another source.

For any specific choice of masking signals $\alpha_i(t)$, refiltering attempts to isolate a *single clean source* from the input signal and suppress all other sources and background noises. Different sources can be isolated by choosing a different set of masking signals. Although, in general, masking signals are real-valued, positive quantities that may take on values greater than unity, in practice the (strong) simplifying assumption that $\alpha_i(t)$ are binary and constant over a timescale τ of roughly 30ms can be made. This assumption is physically unrealistic, because the energy in each small region of time-frequency never comes entirely from a single source. However, for small numbers of sources, this approximation works quite well(Roweis, 2001), in part because of the effect illustrated in figure 1.1b. (Think of ignoring collisions by assuming separate piano players do not often hit the same note at the same time.) (Refiltering can also be thought of as a highly nonstationary Wiener filter in which both the signal and noise spectra are re-estimated at a rate $1/\tau$; the binary assumption is equivalent to assuming that over a timescale τ the signal and noise spectra are nonoverlapping.) It is a fortunate empirical fact that refiltering, even with piecewise constant binary masking signals, *can* cleanly separate sources from a single mixed recording.[2]

Multiband grouping as a statistical pattern recognition problem

Since refiltering for separation and denoising is indeed possible if the masking signals are well chosen, the essential statistical problem is: how can the $\alpha_i(t)$ be computed automatically from a single input record-

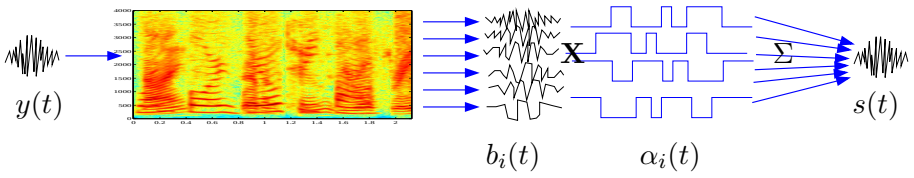

Figure 1.3. Refiltering for separation and denoising. Multiband analysis of the original signal $y(t)$ gives sub-band signals $b_i(t)$ which are modulated by masking signals $\alpha_i(t)$ (binary or real valued between 0 and 1) and recombined to give an estimated source $s(t)$.

ing? The goal is to group together regions of the spectrogram that belong to the same auditory object (and have high signal-to-noise). Fortunately, natural auditory signals—especially speech—exhibit a lot of regularity in the way energy is distributed across the time-frequency plane. Grouping cues based on these regularities have been studied by psychophysicists and are hand built into many CASA systems. Cues are based on the idea of suspicious coincidences: roughly, "things that move together likely belong together". Thus, frequencies which exhibit common onsets, offsets, or upward/downward sweeps are more likely to be grouped into the same stream. Also, many real world sounds have harmonic spectra; so frequencies which lie exactly on a harmonic "stack" are often perceptually grouped together. (Musically, piano players do not hit keys randomly, but instead use chords and repeated melodies.) The approach advocated here to use statistical learning methods to discover these regularities from a large amount of speech data and then to use the learned models to compute the masking signals for new signals in order to perform refiltering.

MAXVQ: Factorial-Max Vector Quantization. It is often advantageous to model complicated sensory observations using a number of separate but interacting causes. One general way to pursue this modeling idea is to have a fixed number M of vector quantizers (or mixture models), each of which proposes an output, and then have some way of combining the output proposals into a final observation. We can think of this as a bank of quantizers, which feed their chosen prototypes into a nonlinear combination box that computes the final output.[3]

Motivated by the observation above regarding the MAX approximation to log spectrograms of mixtures, we propose such a model, called Factorial-Max Vector Quantization (MAXVQ), which uses the MAX operation to combine outputs from the various causes. The model has a bank of M independent vector quantizers, each of which stochastically

selects a prototype with which to model the observation vector. The final output vector is a noisy composite of the set of proposed prototypes, obtained by taking the *elementwise maximum* of the set and adding nonnegative noise. This generative model is illustrated in figure 1.4.

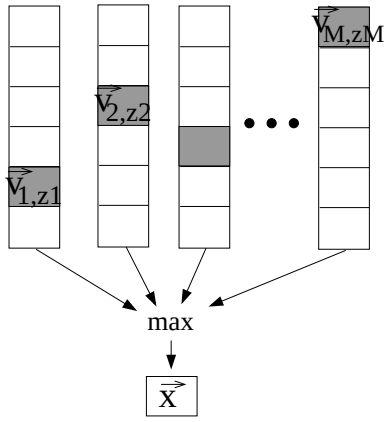

Figure 1.4. Generative model for the Factorial-MAX model. Each of M quantizers selects its z_m^{th} codebook vector $\boldsymbol{v}_{m,z_m}$ and these are combined using elementwise maximum to produce the final output $\boldsymbol{x}$.

The MAXVQ model is useful in situations where there are multiple "objects", "sources" or "causes" in the world but there is some kind of occlusion or sparseness governing how the sources interact to produce observations. For example, as noted above, in clean speech recordings, the log spectrogram of a mixture of speakers is almost exactly the elementwise maximum of the log spectrograms of the individual speakers. For noisy mixtures of speech signals, each clean speaker and each noise source can be thought of as an independent cause contributing to the observed signal. We will use the short-time log power in linearly spaced narrow frequency bands as our vectors when analyzing speech with this model. (As another example, in range finding using laser or acoustic sensors, the distance reading in any direction is the minimum of the distances of all objects from the sensor in that direction.)

Formally, MAXVQ is a latent variable probabilistic model for D-dimensional data vectors $\boldsymbol{x}$. The model consists of M vector quantizers, each with K_m codebook vectors $\boldsymbol{v}_m^k$. Latent variables $z_m \in \{1 \dots K_m\}$ control which codebook vector each vector quantizer selects. Given these selections, the final output $\boldsymbol{x}$ is generated as a noisy version of the elementwise maximum of the selected codewords. If we assume that the each vector quantizer chooses its codebook entries independently with

fixed rates π_m^k, then the model can be written as:

$$\begin{aligned} p(z_m = k|\pi) &= \pi_m^k \qquad m \in \{1 \dots M\}, k \in \{1 \dots K_m\} \\ p(\boldsymbol{z}) &= \prod_m p(z_m), \quad \boldsymbol{z} = (z_1, \dots, z_M) \\ a_d &= \arg\max_m (v_{md}^{z_m}) \\ p(x_d|a_d, \boldsymbol{v}, \Sigma) &= \mathcal{N}^+(x_d|v_{ad}^{z_a}, \Sigma_{ad}) \\ p(\boldsymbol{x}|\boldsymbol{v}, \Sigma, \pi) &= \sum_{\boldsymbol{z}} p(\boldsymbol{z}|\pi) p(\boldsymbol{x}|\boldsymbol{z}, \boldsymbol{v}, \Sigma) \end{aligned}$$

were z_m are latent variables, $\boldsymbol{v}_m^k$ are the codebook vectors, Σ_{md} are noise variances (shared across k), and M, K_m are structural size parameters chosen to control complexity. The distribution $\mathcal{N}^+$ is the positive side of a Gaussian.

MAXVQ can be thought of as an exponentially large mixture of positive Gaussians having $(\prod_m K_m)$ components, with the mean of each component constrained to be the elementwise max of some underlying parameters $\boldsymbol{v}$. This technique, of representing an exponentially large codebook using a factorial expansion of a small number of underlying parameters has been very influential and successful in recent machine learning algorithms (Jojic and Frey, 2000; Ross and Zemel, 2003).

This model can also be extended through time to generate a Factorial-Max Hidden Markov Model (Roweis, 2001; Varga and Moore, 1990). There are some additional complexities, and the details of the heuristics used for inference are slightly different but in our experience the frame-independent MAXVQ model performs almost as well and so for simplicity, we will not discuss the full HMM model.

Parameter Estimation from Isolated Sources. Given some isolated (clean) recordings of individual speech or noise sources, we can estimate the codebook means v_d^k, noise variances Σ_d and the selection probabilities π^k associated with the source's model by training a mixture density or a vector quantizer on the columns of a short-time narrowband log spectrogram. Some care must be taken in training to properly obey the nonnegativity assumption on the noise and to avoid too many codebook entries (mixture components) representing low energy (silent) segments (which are numerous in the data). Also, it is often advantageous to represent two or more adjacent (in time) columns of the spectrogram as a single input vector to allow the model to take some small advantage of temporal continuity.

Inference for Refiltering. The key idea in this section is that the process of inference (i.e. deducing the values of the hidden variables

given the parameters and observations) in a MAXVQ model performs a computation which is extremely useful for computing the masking signals required to perform refiltering for denoising or separation. Because the number of possible joint settings of the hidden selection variables $\boldsymbol{z}$ is exponentially large, we are usually only interested in finding the single most likely (MAP) setting of $\boldsymbol{z}$ given $\boldsymbol{x}$ or the N-best settings. (For unsupervised learning and likelihood computations we may also be interested in efficiently summing over all possible joint settings of $\boldsymbol{z}$ to compute the marginal likelihood of a given observation $\boldsymbol{x}$.) Computing these Viterbi settings (or the sum) is intractable either by direct summation or by naive dynamic programming because of the factorial nature of the model. We must resort to branch-and-bound algorithms for efficient decoding or else approximations (e.g. variational methods) to estimate likely settings of $\boldsymbol{z}$.

Once we have computed the MAP (or approximate) setting of $\boldsymbol{z}$, we can use this to estimate the refiltering masking signals as follows: for each (overlapping) frame of the input spectrogram, set the masking signal to unity for every frequency at which the output proposed by the model corresponding to the source to be recovered is the maximum proposal over all models. Other frequencies have their masks set to zero. Actual refiltering is then performed by retaining the phase from the spectrogram of the original (noisy/mixed) recording, applying the (binary) masking signals to the log magnitude of each frequency, and reconstituting the clean signal using overlap-and-add reconstruction. The windowing function used to compute the original spectrogram must be known (or estimated) in order to remove its effect properly during refiltering.

Branch-and-Bound for Efficient Inference. As discussed above, naive computation of the MAP joint settings of the hidden selection variables in MAXVQ is exponentially expensive. Fortunately, there is a clever branch and bound trick which can be used, based on the following observation: if $z_m = k$, we can *upper bound* the log likelihood we can achieve on a data case $\boldsymbol{x}$, no matter what values the other $z_{m' \neq m}$ take on. The bound $\log p(\boldsymbol{x}|z_m = k) \leq B_{mk}$ is constructed as follows (using constant Σ for simplicity):

$$B_{mk} = -\frac{1}{2}\sum_d [x_d - v_{md}^k]_+^2 - \frac{D}{2}\log|\Sigma| - \log \pi_m^k \tag{1.2}$$

where $[r]_+$ takes the max of zero and r. The intuition is that either $\boldsymbol{v}_m^k$ is *greater than* $\boldsymbol{x}$ along a certain dimension d of the output, in which case the error will be at least $(x_d - v_{md}^k)^2$ or else it is *less than* $\boldsymbol{x}$ along

dimension d in which case the error on that dimension could potentially be zero.

This bound can be used to quickly search for the MAP setting of $\boldsymbol{z}$ given $\boldsymbol{x}$ as follows. For each $m \in \{1 \ldots M\}$ and each $k \in \{1 \ldots K_m\}$, compute the bound B_{mk}. Initially set the guess of the best configuration to the settings with the best bounds: $z_m^* = \arg\min_k B_{mk}$ and compute the true likelihood achieved by that guess: $\ell^* = \log p(\boldsymbol{x}|\boldsymbol{z}^*)$. Now, for each $m \in \{1 \ldots M\}$, we can eliminate all k for which $B_{mk} < \ell^*$. In other words, we can definitively say that certain codebook choices are impossible for certain models, independent of what other models choose because they would incur a minimum error worse that what has already been achieved. Now, for each m, and for all possible settings of k that remain for that m, systematically evaluate $\log p(\boldsymbol{x}|\boldsymbol{z})$ and if it is less than ℓ^*, eliminate the setting. If the likelihood is greater than ℓ^*, we accept it as the new best setting and reset $\boldsymbol{z}^*$ and ℓ^*; we also re-eliminate all settings of k that are now invalid because of this improved bound, and repeat until all settings have been either pruned or checked explicitly. This method is guaranteed to find the *exact* MAP setting, but it comes with no guarantees about its time complexity. In practice, however, we have found it to prune very aggressively and almost always find the MAP configuration in reasonable time. This technique is illustrated in figure 1.5.

Experiments with Factorial-MAX VQ

As an illustration of the methods presented above, we performed simple denoising and separation experiments using TIMIT prompts read by a single speaker and noise (babble) from the NOISEX database. Narrowband spectrograms we constructed from isolated, clean training examples of the speaker and noise. (Signals were downsampled to 12.5kHz, frames of length 512 were used with Hanning windows and frame shifts of 64 samples, resulting in one 257-vector of log energies each 5ms representing the signal over the last 40ms.) A simple vector-quantization codebook with 512 codewords was trained on the speech and one with 32 codewords was trained on the noise. Approximately 5 minutes of speech (with low energy frames eliminated) and 100 seconds of noise were used for training. A modified k-means algorithm which includes split-and-merge heuristics for finding good local optima was used. (We have also experimented with training "scaled" vector quantizers which cluster onto rays in the input space rather than on points, although this technique was not used in the results below.) The trained models were then used to perform MAXVQ inference on previously unseen test

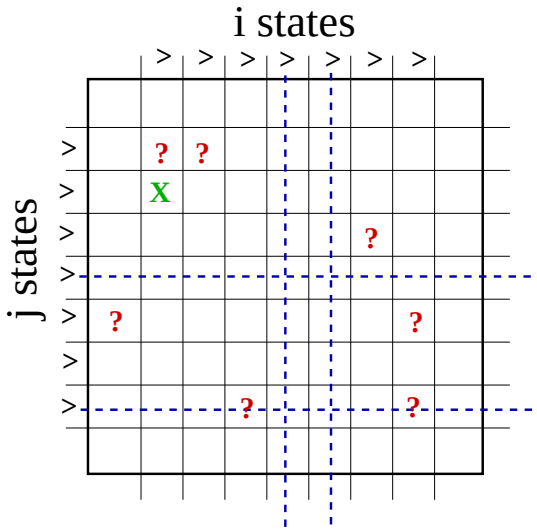

Figure 1.5. Branch and bound inference trick for a factorial-max VQ model with $M = 2$ quantizers. For each codebook selection that quantizer i could make, it is possible to compute a bound on the error (likelihood), regardless of the choice made by quantizer j. Similarly, for each codebook selection that j can make a bound (independent of i's choice) can be calculated. The current best joint selection of i and j is instantiated (shown by an x in the diagram), and its true error (likelihood) is computed. Any choices for either quantizer which are worse than this already achieved value are eliminated since they cannot possibly be involved in the MAP configuration. Remaining valid choices are explored, ordered by their bound values (indicated by ? in the diagram). Once all choices have either been explored or eliminated, we are guaranteed to have found the MAP configuration.

data, using the branch-and-bound technique. Based on this inference, refiltering was performed as described above to recover clean/isolated sources. In the denoising experiment, a 6 second speech segment was linearly mixed with 6 seconds of babble noise at 0dB SNR (equal power). Figure 1.6 shows the results of denoising with MAXVQ and also with a simple spectral subtraction trained on the same isolated noise sample as used for the VQ model. For the separation experiment, two different utterances, spoken by the same speaker, were mixed at equal power and the speech model was used (symmetrically) to perform MAXVQ inference. The results of this monaural separation are shown in figure 1.7. Of course, these results do not represent state of the art performance on either denoising or separation tasks; they are merely a proof of concept that the marriage of refiltering and inference in factorial models can be used for powerful speech processing tasks.

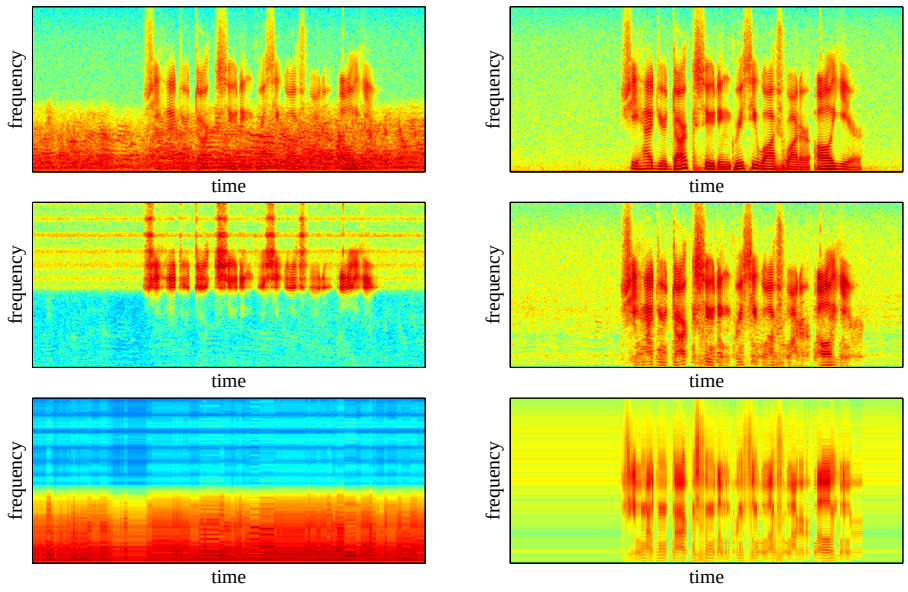

Figure 1.6. Denoising using MAXVQ. Top row: noisy input, original clean source. Middle row: spectral subtraction estimate (trained on isolated noise), MAXVQ estimate after exact branch-and-bound inference and refiltering (trained on isolated speech and noise). Bottom row: proposed codebook output sequence from noise model, proposed codebook output sequence from speech model.

MAXVQ: Discussion, Related & Future Work

We have argued that the *refiltering* approach to separation and denoising can be successfully achieved by using the inference step in a factorial model to provide the masking signals. Varga and Moore (Varga and Moore, 1990) proposed a factorial model for spectrograms (focusing on the factorial nature and using the log-max approximation) as did Gales and Young (Gales and Young, 1996) (focusing on the combination operation) but these models were used for speech recognition in the presence of noise only, and not for refiltering to do separation and denoising. In a series of papers, Green et.al. (Green et al., 2001) have studied masking (refiltering) for denoising, but do not employ factorial model inference as an engine for finding masking signals. There have also been several approaches to monaural separation and denoising that operate mainly in the time domain, without using refiltering or factorial models. Cauwenberghs (Cauwenberghs, 1999) investigated separation based on maximizing periodic coherence; Wan and Nelson (Wan and Nelson, 1998) use nonlinear autoregressive networks and extended Kalman filtering.

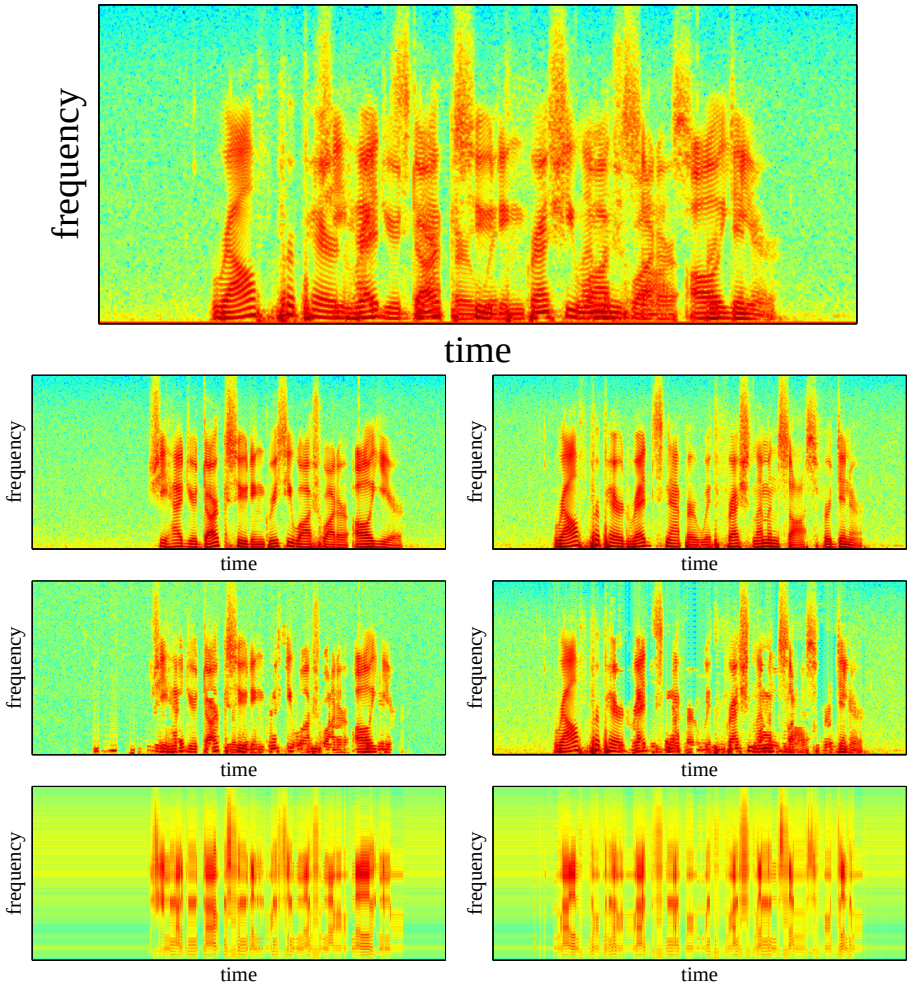

Figure 1.7. Monaural separation using MAXVQ. Top row: narrowband spectrogram of mixed input containing two different utterances spoken simultaneously by the same speaker. Second row: original isolated utterances. Third row: MAXVQ estimates of original utterances after exact branch-and-bound inference and refiltering (trained on isolated speech, not including this test example). Bottom row: proposed codebook output sequence for each stream.

Our work here and previously(Roweis, 2001) is closest in spirit to that of Ephraim et.al. (Ephraim et al., 1989) who model speech using a HMM and noise using an AR model and then attempt to approximately infer the clean speech by alternating between Wiener filtering to find the noise and Viterbi decoding in the HMM. Logan and Moreno (Logan and

Moreno, 1998) also investigated the use of factorial HMMs for modeling speech and found standard HMMs to be just as good, but they did not compose their model using the MAX of two underlying models; rather they learned separate parameters for each combination of states. Reyes et.al. (Reyes et al., 2003) investigated factorial HMMs for separation but using multi-channel inputs.

The main challenge for future work is to develop techniques for learning from only mixed/noisy data, without requiring clean, isolated examples of individual sources or noises at training time. In a maximum likelihood formulation of this purely unsupervised learning setup, we would be given many realizations of $\boldsymbol{x}$ from the model, $\boldsymbol{X} = [\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N]$, assumed to be IID, and we attempt to adjust the model parameters so as to make the observed data more likely. Using the view proposed above, in which MAXVQ is seen as a very large mixture of Gaussians with parameter tying on the means, we can learn the parameters of a MAXVQ model using the standard EM algorithm for maximum likelihood. However, this requires summing over all K^M possible settings of $\boldsymbol{z}$ explicitly. If K^M is small enough for this to be feasible, then this is one possible way to do learning. Otherwise, approximate inference techniques must be used to allow tractable computations.

Along this line, we are investigating a technique which gives approximate rather than exact answers but has a fixed and known time complexity. This idea is to introduce a *factorized variational distribution* which tries to approximate the true (joint) posterior as well as possible. In this setup, we approximate the true posterior $p(\boldsymbol{z}|\boldsymbol{x})$ with a factorized posterior $q(\boldsymbol{z}|\boldsymbol{x}) = \prod_m q(z_m|\boldsymbol{x})$ and proceed to find the functions $q(z_m|\boldsymbol{x})$ which maximize a lower bound on the data likelihood.

2. A Segmental HMM for Speech Waveforms

4

In the following section, we turn our attention to another simple generative model, this time one which operates directly on the speech waveform. Pursuing inference in this model leads to a purely time domain approach to pitch processing which identifies waveform samples at the boundaries between glottal pulse periods (in voiced speech) or at the boundaries between unvoiced segments. An efficient algorithm for inferring these boundaries is derived from a simple probabilistic generative model for segments, which gives excellent results on pitch tracking, voiced/unvoiced detection and timescale modification.

Speech Segments in the Time Domain

Processing of speech signals directly in the time domain is commonly regarded to be difficult and unstable, due to fact that perceptually very similar utterances exhibit very large variability in their raw waveforms. As a result, by far the most common preprocessing step for most speech systems is to convert the raw waveform into a time-frequency representation, using a variety of spectral analysis and filterbank techniques. In this section we explore a purely time domain approach to speech processing in which we identify the samples at the boundaries between glottal pulse periods (in voiced speech) or at the boundaries between unvoiced segments of similar spectral shape.

Having identified these segment boundaries, we can perform a variety of important low level speech analysis and manipulation operations directly and conveniently. For example, we make a voiced/unvoiced decision on each segment by examining the periodicity of the waveform in that segment only. In voiced segments we can estimate the pitch as the reciprocal of the segment length. Timescale modification without pitch or format distortion can be achieved by stochastically eliminating or replicating segments in the time domain directly. More sophisticated operations, such as pitch modification, gender and voice conversion, and companding (volume equalization) are also naturally performed by operating on waveform segments one by one without the need for a cepstral or other such representation. In effect, our model chops up the original speech wave into natural "atomic" units which can be examined or manipulated in very flexible ways.

The computational challenge with this approach is in efficiently and robustly identifying the segment boundaries, across silence, unvoiced and voiced segments. In this section we describe a segmental Hidden Markov Model(Achan et al., 2004), defined on variable length sections of the time domain waveform, and show that performing inference in this model allows us to identify segment boundaries and achieve excellent results on the speech processing tasks described above.

A probabilistic generative model of time-domain speech segments

The goal of our algorithm is to break the time domain speech signal $s_1, \ldots, s_N$ into a set of segments, each of which corresponds to a glottal pulse period or a segment of unvoiced colored noise. Let b_k denote the time index of the beginning of the kth segment and $\mathbf{s}_k = (s_{b_k}, \ldots, s_{b_{k+1}-1})$ denote the waveform in the kth segment, where $k = 1, \ldots, K$ indexes segments. Our algorithm searches for the segment

boundaries, $b_1, b_2, \ldots, b_{K+1}$, so that *each segment can be accurately modeled as a time-warped, amplitude-scaled and amplitude-shifted version of the previous segment.* We denote the transformation used to map segment $\mathbf{s}_{k-1}$ into segment $\mathbf{s}_k$ by $\mathbf{T}_k$.

Given the segment boundaries $b_1, \ldots, b_{K+1}$ and the transformations $\mathbf{T}_1, \ldots, \mathbf{T_k}$ we assume the probability of each segment depends only on the previous segment and the transformation for that segment: in other words we assume the segments are generated by first order Markov chain:

$$P(\mathbf{s}_1, \mathbf{s}_2, \ldots, \mathbf{s}_K | b_1, \ldots, b_{K+1}, \mathbf{T}_1, \ldots, \mathbf{T}_K) = \prod_{k=1}^{K} P(\mathbf{s}_k | \mathbf{s}_{k-1}, b_{k-1}, b_k, b_{k+1}, \mathbf{T}_k). \tag{1.3}$$

Each segment is modeled as a noisy copy of the transformed version of the previous segment. These assumptions simplify the inference and estimation algorithm described below. Of course, number of segments and the segment boundaries are unknown and must be inferred from the speech wave: this inference is the main computation performed by our algorithm.

For concreteness, we assume that each successive segment $\mathbf{s}_k$ is equal to a transformed version of the previous segment, plus isotropic, zero-mean normal noise with variance σ_k^2. Denoting the transformed version of segment $k-1$ by $\mathbf{T}_k \mathbf{s}_{k-1}$, the conditional probability density of $\mathbf{s}_k$ is:

$$P(\mathbf{s}_k | \mathbf{s}_{k-1}, b_{k-1}, b_k, b_{k+1}, \mathbf{T}_k) = \frac{1}{(2\pi\sigma_k^2)^{(b_{k+1}-b_k+1)/2}} \cdot \exp\Big(-\frac{1}{2\sigma_k^2} (\mathbf{s}_k - \mathbf{T}_k \mathbf{s}_{k-1})^{\mathrm{T}} (\mathbf{s}_k - \mathbf{T}_k \mathbf{s}_{k-1}) \Big). \tag{1.4}$$

The noise levels $\sigma_2^2, .., \sigma_K^2$ are estimated automatically by the inference procedure along with the segment boundaries

(As the boundary condition of the Markov chain, we assume that the segment before the first is a vector of all zeros ($\mathbf{s}_0 = \mathbf{0}$) and hence the probability density of the initial segment is given by $(2\pi\sigma_1^2)^{-b_2/2} \exp(-\mathbf{s}_1^{\mathrm{T}} \mathbf{s}_1 / 2\sigma_1^2)$. We also set σ_1^2 equal to the variance of all time-domain samples, since *a priori* we do not know what the content of the initial segment should be.)

We assume that the boundaries and transformations are independent, and that the prior distribution over transformations is uniform on some bounded set. In our experiments, we parameterize the transformation by $\mathbf{T}_k(\alpha_k, \beta_k, \gamma_k)$, where α_k, β_k and γ_k are time-warp, amplitude-scaling and amplitude-shift. We use a prior that is uniform over a 3-dimensional hypercube that includes all reasonable values for these parameters.

Generally the joint prior probability mass function on segment boundaries $P(b_1, \ldots, b_{K+1})$ can be quite complex. Since the computational complexity of the inference algorithm will depend on the number of allowed configurations of segment boundaries, we use a prior that is nonzero only on an appropriate subset of configurations. In particular, we exploit a very simple heuristic (first suggested by John Hopfield in 1998) by *restricting segments to begin and end only on zero crossings of the signal* (or possibly only on upward or downward going zero crossings). This restriction also allows arbitrary segments to be relocated beside each other and still preserve waveform continuity, which will be important in our later applications. To further restrict the range of inferred segment lengths, we require that $\Delta_{\min} \leq b_k - b_{k-1} \leq \Delta_{\max}$, where $\Delta_{\min}$ and $\Delta_{\max}$ are the minimum and maximum segment lengths, satisfying $\Delta_{\max} > \Delta_{\min} > 0$. These minimum and maximum segment lengths are chosen to represent the widest possible range of pitch periods we expect to see in our signals. We assume the probability $P(b_1, \ldots, b_{K-1})$ is otherwise uniform, subject to the above constraints. The number of segments K is also unknown, and its optimal value is inferred automatically as well. We assume that $b_1 = 1$ (the first segment begins on the first signal sample) and that $b_{K+1} = N + 1$ (the last segment ends on the last signal sample).

The joint distribution over segments, segment boundaries and transformations can now be written as:

$$P(\mathbf{s}_1, \ldots, \mathbf{s}_K, b_1, \ldots, b_{K+1}, \mathbf{T}_1, \ldots, \mathbf{T}_K) \propto P(b_1, \ldots, b_{K+1}) \prod_{k=1}^{K} P(\mathbf{T}_k) P(\mathbf{s}_k | \mathbf{s}_{k-1}, b_{k-1}, b_k, b_{k+1}, \mathbf{T}_k), \qquad (1.5)$$

where $P(b_1, \ldots, b_{K+1})$ enforces the constraints on the boundaries; constraints on the allowable limits of the time domain scale, amplitude-domain scale and amplitude-domain shift are enforced by $\left(\prod_{k=1}^{K} P(\mathbf{T}_k)\right)$ although these constraints rarely affect the optimization.

Using dynamic programming to efficiently infer segment boundaries and transformations

Given a time-domain signal, the computational task now at hand is to determine the segment boundaries and transformations. Of course, the number of valid configurations of the boundary variables is exponential in the length of the waveform, so exhaustive search would be intractable. Fortunately, the optimal segmentation (which maximizes likelihood, or equivalently, minimizes the mismatch penalties) can be found using a generalized dynamic programming algorithm.

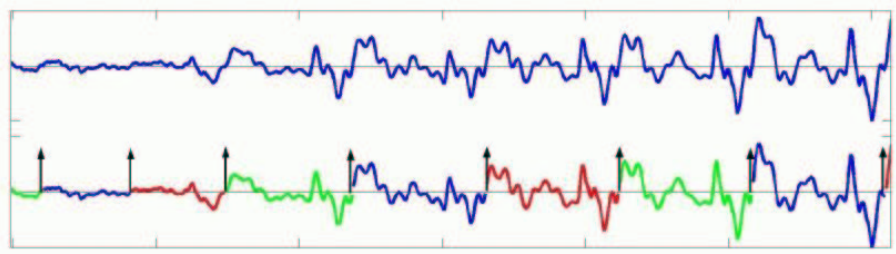

*Figure 1.8. (top)*Input signal; notice the transition from unvoiced to voiced region. *(bottom)* Inferred maximum likelihood segmentation found using generalized dynamic programming. The upward arrows are used to mark the inferred segment boundaries.

First, note that according to 1.5, given the boundary variables, the MAP estimates of the transformations can be computed locally:

$$\begin{aligned}&\arg\max_{\mathbf{T}_k} P(\mathbf{s}_1,\ldots,\mathbf{s}_K,b_1,\ldots,b_{K+1},\mathbf{T}_1,\ldots,\mathbf{T}_K)\\&=\arg\max_{\mathbf{T}_k} P(\mathbf{T}_k)P(\mathbf{s}_k|\mathbf{s}_{k-1},b_{k-1},b_k,b_{k+1},\mathbf{T}_k).\end{aligned}\tag{1.6}$$

In particular, the time-warping is unique and is given by $\alpha_k = (b_{k+1} - b_k)/(b_k - b_{k-1})$. The warped version of $\mathbf{s}_{k-1}$ is denoted by $\hat{\mathbf{s}}_{k-1}$ and can be obtained using standard signal processing techniques for time-domain interpolation or decimation. Note that whereas $\mathbf{s}_{k-1}$ contains $b_k - b_{k-1}$ samples, $\hat{\mathbf{s}}_{k-1}$ contains $b_{k+1} - b_k$ samples. The amplitude-domain scale β_k and shift γ_k are obtained by performing a least-squares regression of $\hat{\mathbf{s}}_{k-1}$ onto $\mathbf{s}_k$, *i.e.* by solving

$$\arg\min_{\beta_k,\gamma_k} \sum_{j=1}^{b_{k+1}-b_k} (\beta_k\hat{\mathbf{s}}_{k-1}(j) + \gamma_k - \mathbf{s}_k(j))^2,\tag{1.7}$$

where (j) indexes the elements of $\mathbf{s}_k$ and $\hat{\mathbf{s}}_{k-1}$. After optimizing β_k and γ_k, the estimate of the variance σ_k^2 is set to the argument in the above minimization, divided by $b_{k+1} - b_k$. For a given configuration of b_{k-1}, b_k, b_{k+1}, we denote the optimal transformation obtained in the above fashion by $\mathbf{T}_k^*$.

Thus, the search is one over boundary segment positions, which for efficiency we constrain to lie only at (just after) zero crossings of the waveform. Finding the optimal segmentation requires performing dynamic programming, using a table indexed with two adjacent boundary points. In order to make the optimization Markovian, we must actually consider *adjacent pairs* of boundary points (b_{k-1}, b_k) as the states in the dynamic programming. In particular, we fill in a table C whose entry

$C(m,n)$ holds the best possible log likelihood of the segmentation ending with the segment defined by the m^{th} zero crossing at its left edge and the n^{th} zero crossing at its right edge. We can iteratively fill in this table forwards for all values $m < n$, by using the following recursion:

$$C(m,n) = \arg\max_i \left[C(m,i) + \log P(\mathbf{s}_k|\mathbf{s}_{k-1}, b_{k-1} = m, b_k = i, b_{k+1} = n, \mathbf{T}_k)\right] \tag{1.8}$$

Segmental HMM Experiments

We have applied our segmental inference procedure to clean, wideband recordings of single-talker speech, from both males and females taken from the the Keele pitch reference dataset (Plante et al., 1995) and from the Wall Street Journal (WSJ) corpus. Dynamic programming was applied with segment length thresholds of $\Delta_{\min}$=3ms and $\Delta_{\max}$=25ms (corresponding to pitch range of 40Hz-333Hz) to find the optimal segmentation of the raw waveforms directly.

We can apply the results of our segment inference algorithm to a wide range of speech processing tasks, as discussed below. By replicating or deleting some or all of the inferred segments, we can easily achieve high quality timescale modification without changing the perceived pitch or formant structure of the utterance. By examining the periodicity of each segment, we can attempt to distinguish voiced from unvoiced portions of the waveform. In voiced regions, we can directly estimate the pitch by taking the reciprocal of the segment length. Below, we present results on timescale modification, voiced/unvoiced discrimination, and pitch tracking. Other applications such as gender and voice conversion, companding and concert hall effects are also possible. We emphasize that all the experiments were performed in *time domain* using the inferred pitch periods.

Voicing Detection and Pitch Tracking. For voicing detection and pitch tracking, we evaluated the estimates obtained using our algorithm using the Keele dataset, since it has ground truth values for these quantities. (In particular, the Keele data has utterances spoken by both male and female speakers and includes a reference estimate for the fundamental frequency at a resolution of 10ms. Each utterance is approximately 30 seconds long and the sampling frequency is 20kHz.)

Once the waveform segments are inferred by the algorithm, we can estimate the periodicity of each segment in a simple way by computing the discrete Fourier transform of the segment waveform and then reconstructing it using a limited number of Fourier coefficients. (This is illustrated in figure 1.9.)

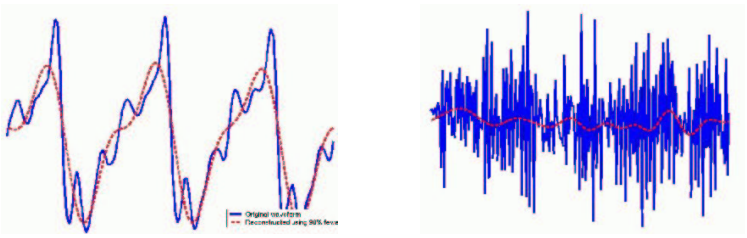

Figure 1.9. Simple voicing detection given waveform segmentation. Each segment is reconstructed using a small number of Fourier coefficients. Segments whose reconstruction error is below some threshold (and whose energy is above the silence threshold) are tagged as voiced. Examples above show typical voiced (left) and unvoiced (right) segments and their reconstructions.

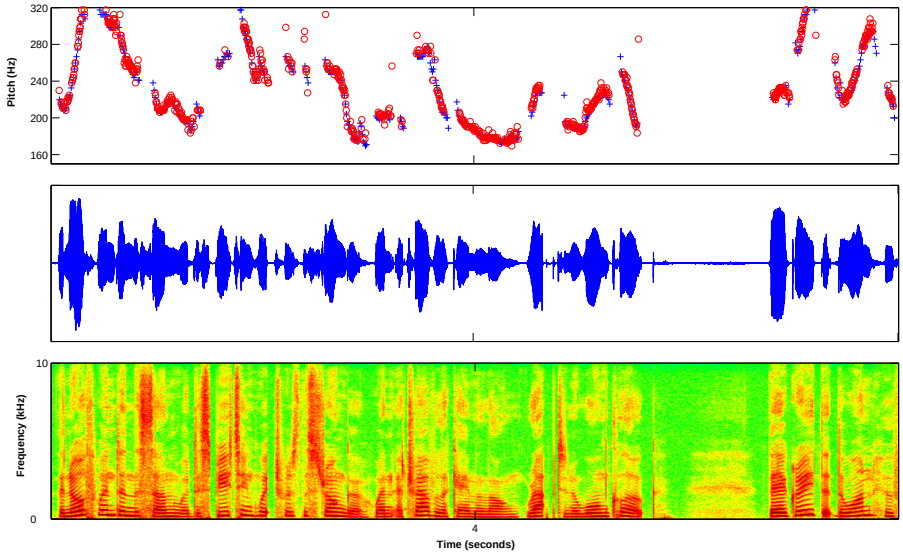

Figure 1.10. *(top)*Pitch estimates using segmental HMM for a female speaker in the Keele dataset. Notice that the inferred pitch(red circle) consistently agrees with the reference provided (blue plus mark). Further, our approach clearly discriminates between voiced/unvoiced regions (samples without reference estimates are unvoiced). *(center)* input time domain signal *(bottom)* spectrogram of input

Since unvoiced regions tend to be much less periodic, they will have a substantially larger reconstruction error than voiced regions and by selecting an appropriate threshold, we can discriminate between voiced and unvoiced segments. Our method was able to correctly identify 87.2 % of the voiced segments averaged over all the 10 utterances of males and females in the Keele dataset. In Fig.1.10, the true unvoiced regions

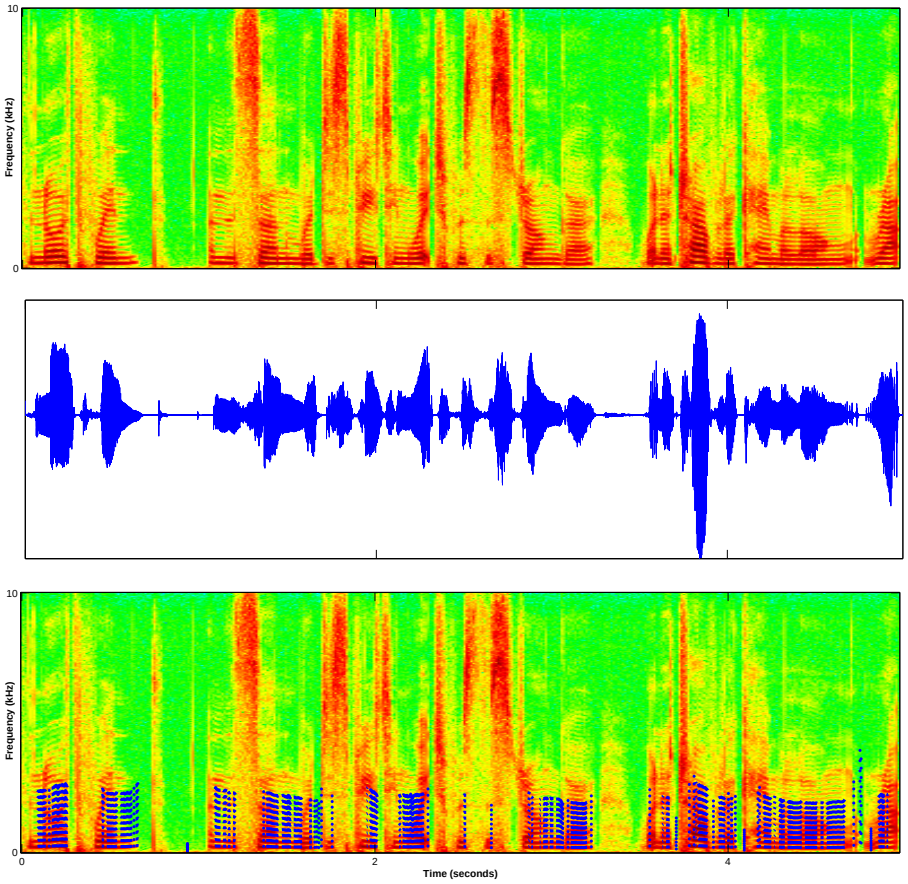

*Figure 1.11. (Middle and Top)*Time domain signal and the corresponding spectrogram *(Bottom)*The spectrogram of the signal is marked with the pitch estimates obtained using our algorithm (blue marker); for clarity we have marked only the first 10 integer multiples of the fundamental frequency

are the segments without any reference pitch shown, and the unvoiced regions detected by our algorithm are those without estimated pitches.

Pitch tracking is trivially achieved by taking the reciprocal of the segment lengths in the voiced regions. Results for a single utterance in the Keele dataset spoken by a female speaker is shown in Fig.1.10. Pitch estimates obtained using our approach are very consistent with the reference estimates; similar performance was obtained on other utterances in the dataset as well. Averaged over 10 utterances the median absolute pitch error was 9Hz.

It is well known that excitation for voiced speech manifests as sharp bursts at integer multiples of fundamental frequency. In Fig.1.11, we

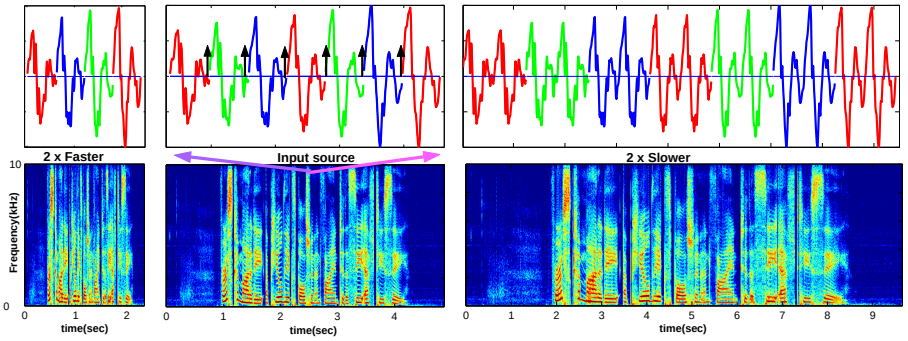

Figure 1.12. The spectrogram of time scale modified faster and slower versions of a signal are shown. The actual time domain operation is shown on top for a particular time instant in the spectrogram.

have shown a few integer multiples of the fundamental frequency of a signal on its spectrogram using pitch estimates obtained from the application of our algorithm.

Time Scale Modification. For timescale modification experiments, we have used utterances from the WSJ corpus. Once the segments are identified by our algorithm, we can play the signal twice as fast by deleting every other segment and concatenating the remaining ones; similarly by replicating each segment we can achieve the effect of playing the at half the speed (two times slower); this is further illustrated in Fig.1.12. This approach is substantially different from methods such as (S. and Wilgus., 1985) that manipulate spectrograms. By doing all of our operations directly in the time domain we never need to worry about inconsistent phase estimates.

Segmental HMM: Discussion & Conclusions

We have presented a simple segmental Hidden Markov Model for generating a speech waveform and derived an efficient algorithm for approximate inference in the model. Applied to an observed signal, this inference algorithm operates entirely in the time domain and is capable of identifying the boundaries of glottal pulse periods in voiced speech and of unvoiced segments. Using these inferred boundaries we are able to easily and accurately detect voicing, track pitch and modify the timescales. We are investigating other possible applications of the same basic model, including voice conversion, volume equalization and reverberant filtering.

3. Constrained Hidden Markov Models for Articulatory Modeling

Structured time-series are generated by systems whose underlying state variables change in a continuous way but whose state to output mappings are highly nonlinear, many to one and not smooth. Probabilistic unsupervised learning for such sequences requires models with two essential features: latent (hidden) variables and *topology* in those variables.

By thinking of each state in a hidden Markov model as corresponding to some spatial region of a fictitious *topology space* it is possible to naturally define neighbouring states of any state as those which are connected in that space. The transition matrix of the HMM can then be constrained to allow transitions only between neighbours; this means that all valid state sequences correspond to connected paths in the topology space. This strong constraint makes structure discovery in sequences easier. We show how such *constrained HMMs* can learn to discover underlying structure in complex sequences of high dimensional data, and apply them to the problem of recovering mouth movements from acoustics in continuous speech. This problem has a long history in speech science, and exemplifies exactly the sort of structured time series analysis problem discussed above.

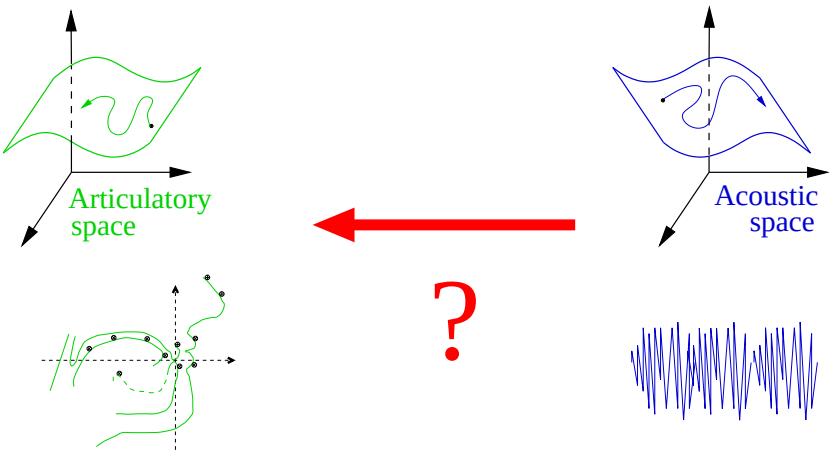

Figure 1.13. Can you hear the shape of the mouth? The problem of recovering articulator motions from acoustics is a classic inversion problem involving physics, speech science and statistical signal processing.

Constrained HMMs as latent variable models

Hidden Markov models (HMMs) can be thought of as dynamic generalizations of discrete state static data models such as Gaussian mixtures, or as discrete state versions of linear dynamical systems (LDSs) (which are themselves dynamic generalizations of continuous latent variable models such as factor analysis). While both HMMs and LDSs provide probabilistic latent variable models for time-series, both have important limitations. Traditional HMMs have a very powerful model of the relationship between the underlying state and the associated observations because each state stores a private distribution over the output variables. This means that any change in the hidden state can cause arbitrarily complex changes in the output distribution. However, it is extremely difficult to capture reasonable dynamics on the discrete latent variable because in principle any state is reachable from any other state at any time step and the next state depends only on the current state. This allows only "random jump" movement in the hidden state space. For example, one well known difficulty is that the lifetime of any single state is distributed according to a decaying exponential, which is often an inappropriate distribution for state dwell times. LDSs, on the other hand, have an extremely impoverished representation of the outputs as a function of the latent variables since this transformation is restricted to be global and linear: a single matrix captures the state to output mapping and it is applied uniformly regardless of location in the state space. But it is somewhat easier to capture state dynamics since the state is a multidimensional vector of continuous variables on which a matrix "flow" is acting; this enforces some continuity of the latent variables across time. *Constrained hidden Markov models* (Roweis, 2000) address the modeling of state dynamics by building some topology into the hidden state representation. The essential idea is to constrain the transition parameters of a conventional HMM so that the discrete-valued hidden state evolves in a structured way.[5] In particular, below we consider parameter restrictions which constrain the state to evolve as a discretized version of a continuous multivariate variable, i.e. so that it inscribes only connected paths in some space. This lends a physical interpretation to the discrete state trajectories in an HMM.

An illustrative game

Consider playing the following game: divide a sheet of paper into several contiguous, non-overlapping regions which between them cover it entirely. In each region inscribe a symbol, allowing symbols to be *repeated* in different regions. Place a pencil on the sheet and move it

around, reading out (in order) the symbols in the regions through which it passes. Add some *noise* to the observation process so that some fraction of the time incorrect symbols are reported in the list instead of the correct ones. The game is to reconstruct the configuration of regions on the sheet from only such an ordered list(s) of noisy symbols. Of course, the absolute scale, rotation and reflection of the sheet can never be recovered, but learning the essential *topology* may be possible.[6] Figure 1.14 illustrates this setup.

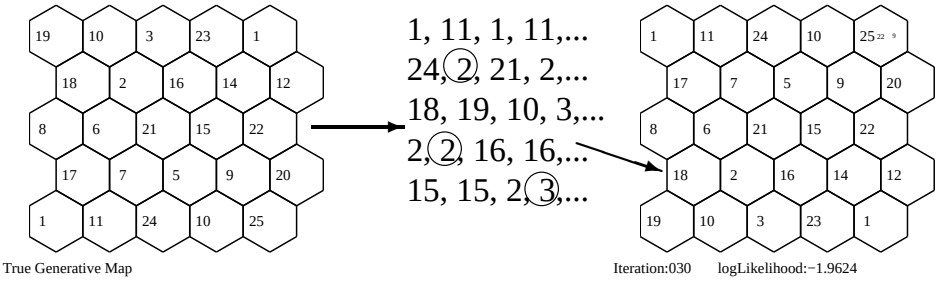

Figure 1.14. **(left)** True map which generates symbol sequences by random movement between connected cells. **(centre)** An example noisy output sequence with noisy symbols circled. **(right)** Learned map after training on 3 sequences (with 15% noise probability) each 200 symbols long. Each cell actually contains an entire distribution over all observed symbols, though in this case only the upper right cell has significant probability mass on more than one symbol. Only the top three symbols of each cell's histogram are show, with *font size proportional to the square root of probability* (to make ink roughly proportional).

Without noise or repeated symbols, the game is easy (non-probabilistic methods can solve it) but in their presence it is not. One way of mitigating the noise problem is to do statistical averaging. For example, one could attempt to use the average separation *in time* of each pair of symbols to define a dissimilarity between them. It then would be possible to use methods like multi-dimensional scaling or a sort of *Kohonen mapping though time*[7] to explicitly construct a configuration of points obeying those distance relations. However, such methods still cannot deal with many-to-one state to output mappings (repeated numbers in the sheet) because by their nature they assign a unique spatial location to each symbol.

Playing this game is analogous to doing unsupervised learning on structured sequences. (The game can also be played with continuous outputs, although often high-dimensional data can be effectively clustered around a manageable number of prototypes; thus a vector time-series can be converted into a sequence of symbols.) Constrained HMMs incorporate latent variables with topology yet retain powerful nonlinear

output mappings and can deal with the difficulties of noise and many-to-one mappings mentioned above; so they can "win" our game (see fig. 1.14). The key insight is that the game generates sequences exactly according to a hidden Markov process whose transition matrix allows only transitions between neighbouring cells and whose output distributions have most of their probability on a single symbol with a small amount on all other symbols to account for noise.

Model definition: state topologies from cell packings

Defining a constrained HMM involves identifying each state of the underlying (hidden) Markov chain with a spatial cell in a fictitious *topology space*. This requires selecting a *dimensionality* d for the topology space and choosing a *packing* (such as hexagonal or cubic) which fills the space. The number of cells in the packing is equal to the number of states M in the original Markov model. Cells are taken to be all of equal size and (since the scale of the topology space is completely arbitrary) of unit volume. Thus, the packing covers a volume M in topology space with a side length ℓ of roughly $\ell = M^{1/d}$. The dimensionality and packing together define a vector-valued function $\boldsymbol{x}(m), m = 1 \ldots M$ which gives the location of cell m in the packing. (For example, a cubic packing of d dimensional space defines $\boldsymbol{x}(m{+}1)$ to be $\left[m, m/\ell, m/\ell^2, \ldots, m/\ell^{d-1}\right] \mathrm{mod} \ell$.) (here a mod b denotes the remainder after dividing a by b). State m in the Markov model is assigned to to cell m in the packing, thus giving it a location $\boldsymbol{x}(m)$ in the topology space. Finally, we must choose a *neighbourhood rule* in the topology space which defines the neighbours of cell m; for example, all "connected" cells, all face neighbours, or all those within a certain radius. (For cubic packings, there are 3^d-1 connected neighbours and $2d$ face neighbours in a d dimensional topology space.) The neighbourhood rule also defines the boundary conditions of the space – e.g. periodic boundary conditions would make cells on opposite extreme faces of the space neighbours with each other.

The transition matrix of the HMM is now *preprogrammed* to only allow transitions between neighbours. All other transition probabilities are set to zero, making the transition matrix very sparse. (We set all permitted transitions to be equally likely.) Now, *all valid state sequences in the underlying Markov model represent connected ("city block") paths through the topology space.* Figure 1.15 illustrates this for a three-dimensional model.

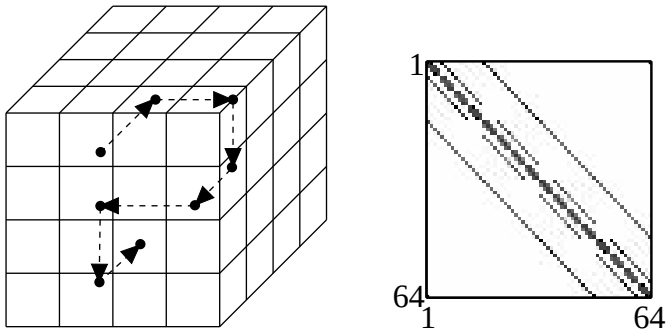

Figure 1.15. **(left)** Physical depiction of the topology space for a constrained HMM with d=3,ℓ=4 and M=64 showing an example state trajectory. **(right)** Corresponding transition matrix structure for the 64-state HMM computed using face-centred cubic packing. The gaps in the inner bands are due to edge effects.

State inference and learning

The constrained HMM has exactly the same inference procedures as a regular HMM: the *forward-backward algorithm* for computing state occupation probabilities and the *Viterbi decoder* for finding the single best state sequence. Once these discrete state inferences have been performed, they can be transformed using the state position function $\boldsymbol{x}(m)$ to yield probability distributions over the topology space (in the case of forward-backward) or paths through the topology space (in the case of Viterbi decoding). This transformation makes the outputs of state decodings in constrained HMMs comparable to the outputs of inference procedures for continuous state dynamical systems such as Kalman smoothing.

The learning procedure for constrained HMMs is also almost identical to that for HMMs. In particular, the EM algorithm (Baum-Welch) is used to update model parameters. The crucial difference is that the transition probabilities which are precomputed by the topology and packing are never updated during learning. In fact, having a preprogrammed and fixed transition matrix makes learning much easier in some cases. Not only do the transition probabilities not have to be learned, but their structure constrains the hidden state sequences in such a way as to make the learning of the output parameters much more efficient when the underlying data really does come from a spatially structured generative model. Notice that in this case, each part of state space had only a single output (except for noise) so the final learned output distributions became essentially minimum entropy. But constrained HMMs can in

principle model stochastic or multimodal output processes since each state stores an entire private distribution over outputs.

Recovery of mouth movements from speech audio

We have applied the constrained HMM approach described above to the problem of recovering mouth movements from the acoustic waveform in human speech. Data containing simultaneous audio and articulator movement information was obtained from the University of Wisconsin X-ray microbeam database (Westbury, 1994). Eight separate points (four on the tongue, one on each lip and two on the jaw) located in the midsaggital plane of the speaker's head were tracked while subjects read various words, sentences, paragraphs and lists of numbers. The x and y coordinates (to within about $\pm$1mm) of each point were sampled at 146Hz by an X-ray system which located gold beads attached to the feature points on the mouth, producing a 16-dimensional vector every 6.9ms. The audio was sampled at 22kHz with roughly 14 bits of amplitude resolution but in the presence of machine noise.

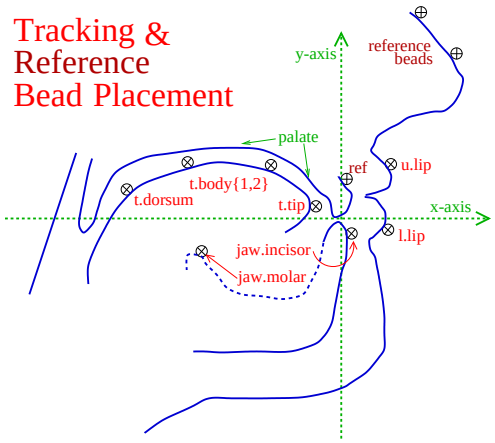

Figure 1.16. Midsaggital locations of tracking beads in the University of Wisconsin X-ray microbeam articulatory dataset.

How do these data relate to the general task introduced above? These data are well suited to the constrained HMM architecture. They come from a system whose state variables are known, because of physical constraints, to move in connected paths in a low degree-of-freedom space. In other words the (normally hidden) articulators (movable structures of the mouth), whose positions represent the underlying state of the speech production system,[8] move slowly and smoothly. The observed speech signal—the system's output—can be characterized by a sequence

of short-time spectral feature vectors, often known as a *spectrogram*. In the experiments reported here, we have characterized the audio signal using 12 line spectral frequencies (LSFs) measured every 6.9ms (to coincide with the articulatory sampling rate) over a 25ms window. These LSF vectors characterize only the *spectral shape* of the speech waveform over a short time but not its energy. Average energy (also over a 25ms window every 6.9ms) was measured as a separate one dimensional signal. Unlike the movements of the articulators, the audio spectrum/energy can exhibit quite abrupt changes, indicating that the mapping between articulator positions and spectral shape is not smooth. (Compare the sampling rate of 22kHz for the acoustic signal with 146Hz for the articulators.) Furthermore, the mapping is many to one: *different* articulator configurations can produce very similar spectra (see below).

The unsupervised learning task, then, is to explain the complicated sequences of observed spectral features (LSFs) and energies as the outputs of a system with a low-dimensional state vector that changes slowly and smoothly. In other words, can we learn the parameters[9] of a constrained HMM such that connected paths through the topology space (state space) generate the acoustic training data with high likelihood? Once this unsupervised learning task has been performed, we can (as shown below) relate the learned trajectories in the topology space to the true (measured) articulator movements.

While many models of the speech production process predict the many-to-one and non-smooth properties of the articulatory to acoustic mapping, it is useful to confirm these features by looking at real data. Figure 1.17 shows the experimentally observed distribution of articulator configurations used to produce similar sounds. It was computed as follows. All the acoustic and articulatory data for a single speaker are collected together. Starting with some sample called the *key sample*, we find the 1000 samples "nearest" to this key by two measures: articulatory distance, defined using the Mahalanobis norm between two position vectors under the global covariance of all positions for the appropriate speaker, and spectral shape distance, again defined using the Mahalanobis norm but now between two line spectral frequency vectors using the global LSF covariance of the speaker's audio data. In other words, I find the 1000 samples that "look most like" the key sample in mouth shape and that "sound most like" the key sample in spectral shape. we then plot the tongue bead positions of the key sample (as a thick cross), and the 1000 nearest samples by mouth shape (as a thick ellipse) and spectral shape (as dots). The points of primary interest are the dots; they show the distribution of tongue positions used to generate very similar sounds. (The thick ellipses are shown only as a control

to ensure that many nearby points to the key sample *do* exist in the dataset.) Spread or multimodality in the dots indicates that many *different* articulatory configurations are used to generate the *same* sound.

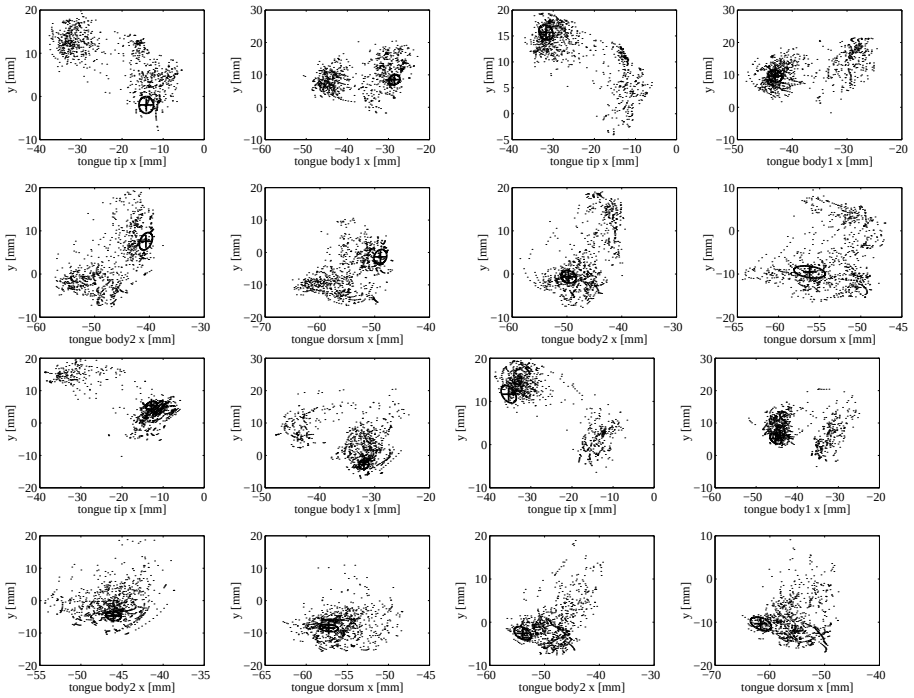

Figure 1.17. Inverse mapping from acoustics to articulation is ill-posed in real speech production data. Each group of four articulator-space plots shows the 1000 samples in the entire dataset which are "nearest" to one key sample (thick cross). The dots are the 1000 nearest samples using an acoustic measure based on line spectral frequencies. Spread or multimodality in the dots indicates that many different articulatory configurations are used to generate very similar sounds. Only the positions of the four tongue beads have been plotted. Four examples (with different key samples) are shown, one each group of four panels. The thick ellipses (shown as a control) are the two-standard deviation contour of the 1000 nearest samples using an articulatory position distance metric.

Why not do direct supervised learning from short-time spectral features (LSFs) to the articulator positions? The ill-posed nature of the inverse problem as shown in figure 1.17 makes this impossible. To illustrate this difficulty, we have attempted to recover the articulator positions from the acoustic feature vectors using Kalman smoothing on a LDS. In this case, since we have access to both the hidden states (ar-

ticulator positions) and the system outputs (LSFs) we can compute the optimal parameters of the model directly. (In particular, the state transition matrix is obtained by regression from articulator positions and velocities at time t onto positions at time $t+1$; the output matrix by regression from articulator positions and velocities onto LSF vectors; and the noise covariances from the residuals of these regressions.) Figure 1.18b shows the results of such smoothing; the recovery is quite poor, even when the test utterance is included in the training set used to estimate model parameters.

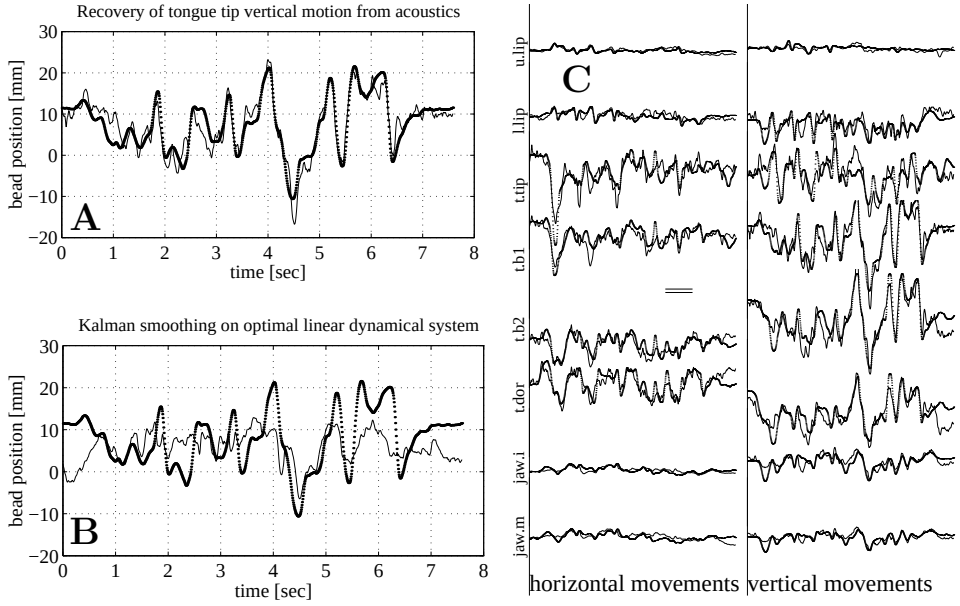

Figure 1.18. **(A)** Recovered articulator movements using state inference on a constrained HMM. A four-dimensional model with 4096 states (d=3,ℓ=6,M=216) was trained on data (all beads) from a single speaker but not including the test utterance shown. Dots show the actual measured articulator movements for a single bead coordinate versus time; the thin lines are estimated movements from the corresponding acoustics. **(B)** Unsuccessful recovery of articulator movements using Kalman smoothing on a global LDS model. All the (speaker-dependent) parameters of the underlying linear dynamical system are known; they have been set to their optimal values using the true movement information from the training data. Furthermore, for this example, the test utterance shown was included in the training data used to estimate model parameters. **(C)** All 16 bead coordinates; all vertical axes are the same scale. Bead names are shown on the left. Horizontal movements are plotted in the left-hand column and vertical movements in the right-hand column. The separation between the two horizontal lines near the centre of the right panel indicates the machine measurement error.

Constrained HMMs can be applied to this recovery problem, as previously reported (Roweis and Alwan, 1997). (Earlier results used a small subset of the same database that was not continuous speech and did not provide the hard experimental verification (fig. 1.17) of the many-to-one problem.)

The basic idea is to train (unsupervised) on sequences of acoustic-spectral features and then map the topology space state trajectories onto the measured articulatory movements. Figure 1.18 shows movement recovery using state inference in a four-dimensional model with 4096 states (d=4,ℓ=8,M=4096) trained on data (all beads) from a single speaker. (Naive unsupervised learning runs into severe local minima problems. To avoid these, in the simulations shown above, models were trained by slowly annealing two learning parameters[10]: a term ϵ^β was used in place of the zeros in the sparse transition matrix, and γ_t^β was used in place of $\gamma_t = p(m_t|observations)$ during inference of state occupation probabilities. Inverse temperature β was raised from 0 to 1.) To infer a continuous state trajectory from an utterance after learning, we first do Viterbi decoding on the acoustics to generate a discrete state sequence m_t and then interpolate smoothly between the positions $\boldsymbol{x}(m_t)$ of each state.

After unsupervised learning, a single linear fit is performed between these continuous state trajectories and actual articulator movements on the training data. (The model cannot discover the units system or axes used to represent the articulatory data.) To recover articulator movements from a previously unseen test utterance, we infer a continuous state trajectory as above and then apply the single linear mapping (learned only once from the training data).

Constrained HMMs for Articulatory Data: Conclusions, extensions and related work

By enforcing a simple constraint on the transition parameters of a standard HMM, a link can be forged between discrete state dynamics and the motion of a real-valued state vector in a continuous space. For complex time-series generated by systems whose underlying latent variables do in fact change slowly and smoothly, such constrained HMMs provide a powerful unsupervised learning paradigm. They can model state to output mappings that are highly nonlinear, many to one and not smooth. Furthermore, they rely only on well understood learning and inference procedures that come with convergence guarantees.

Results on synthetic and real data show that these models can successfully capture the low-dimensional structure present in complex vector

time-series. In particular, we have shown that a speaker dependent constrained HMM can accurately recover articulator movements from continuous speech to within the measurement error of the data. This acoustic to articulatory inversion problem has a long history in speech processing (see e.g. (Schroeter and Sondhi, 1994) and references therein). Many previous approaches have attempted to exploit the smoothness of articulatory movements for inversion or modeling: Hogden *et.al* (e.g. (Nix and Hogden, 1999)) provided early inspiration for these ideas, but do not address the many-to-one problem; Simon Blackburn (Blackburn and Young, 1996) has investigated a *forward* mapping from articulation to acoustics but does not explicitly attempt inversion; early work at Waterloo (Ramsay and Deng, 1994) suggested similar constraints for improving speech recognition systems but did look at real articulatory data, more recent work at Rutgers (Chennoukh et al., 1997) developed a very similar system much further with good success. Perpiñán (Carreira-Perpinan, 2000), considers a related problem in sequence learning using EPG speech data as an example.

While in this section we have described only "diffusion" type dynamics (transitions to all neighbours are equally likely) it is also possible to consider *directed flows* which give certain neighbours of a state lower (or zero) probability. The left-to-right HMMs mentioned earlier are an example of this for one-dimensional topologies. For higher dimensions, flows can be derived from discretization of matrix (linear) dynamics or from other physical/structural constraints. It is also possible to have many connected local flow regimes (either diffusive or directed) rather than one global regime as discussed above; this gives rise to *mixtures* of constrained HMMs which have block-structured rather than banded transition matrices. Smyth (Smyth, 1997) has considered such models in the case of one-dimensional topologies and directed flows; we have applied these to learning character sequences from English text. Another application I have investigated is map learning from multiple sensor readings. An explorer (robot) navigates in an unknown environment and records at each time many local measurements such as altitude, pressure, temperature, humidity, etc. We wish to reconstruct from only these sequences of readings the topographic maps (in each sensor variable) of the area as well as the trajectory of the explorer. A final application is tracking (inferring movements) of articulated bodies using video measurements of feature positions.

GLOSSARY

HMM Hidden Markov Model.

spectrogram Log periodogram showing the absolute value of short-time Fourier transforms of the original signal.

inference The process of estimating the conditional distribution of hidden (latent) variables in a model given the observed values and the model parameters.

generative model A stochastic machine which outputs data in the form of the data we are trying to analyze, whose behaviour is controlled by a set of tuneable parameters.

hidden (latent) variables Quantities such as hidden factors, states in a Markov chain or cluster assignments which are used in statistical models to explain complex variability in observed data but are not measured directly.

4. Summary

In this chapter, we have explored the use of inference in probabilistic generative models as a powerful signal processing tool for speech and audio. The basic paradigm explored was to design a simple model for the data we observe in which the key quantities that we would eventually like to compute appear as hidden (latent) variables. By executing probabilistic inference in such models, we automatically estimating the hidden quantities and thus perform our desired computation. In a sense, the rules of probability derive for us, automatically, the optimal signal processing algorithm for our desired outputs given our inputs under the model assumptions. Crucially, even though the generative model may be quite simple and may not capture all of the variability present in the data, the results of inference can still be extremely informative.

We gave several examples showing how inference in very simple generative models can be used to perform surprisingly complex speech processing tasks including denoising, source separation, pitch tracking, timescale modification and estimation of articulatory movements from audio.

Acknowledgments

STR is funded in part by NSERC Canada, by the Premier's Research Excellence Award of Ontario and by the Canada Research Chairs Program. Thanks to Lawrence Saul and Chris Harvey for helpful discussions about the MAXVQ model. A shortened version of the first section of this chapter was presented at a special session of Eurospeech 2003 (Geneva) organized by Mazin Rahim and Rob Shapire and. Work on the Segmental HMM was done in collaboration with Kannan Achan at the University of Toronto. John Hopfield first proposed the idea of a segmental model constrained to have boundaries at zero crossings.

Notes

1. An equivalent operation can be performed in the frequency domain by making a conventional spectrogram of the original signal $y(t)$ and modulating the magnitude of each short time DFT while preserving its phase: $s^w(\tau) = \mathcal{F}^{-1}\left\{\alpha^w \|\mathcal{F}\{y^w(\tau)\}\| \angle\mathcal{F}\{y^w(\tau)\}\right\}$ where $s^w(\tau)$ and $y^w(\tau)$ are the w^{th} windows (blocks) of the recovered and original signals, α_i^w is the masking signal for subband i in window w, and $\mathcal{F}[\cdot]$ is the DFT.

2. This can be demonstrated artificially by taking several isolated sources or noises and mixing them in a controlled way. Since the original components are known, an "optimal" set of masking signals can be computed. For example, we might set $\alpha_i(t)$ equal to the ratio of energy from one source in band i around times $t \pm \tau$ to the sum of energies from all sources in the same band at that time (as recommended by the Wiener filter) or to a binary version which thresholds this ratio. Constructing masks in this way is, of course, not possible when we are confronted with an unknown mixture or corrupted signal, but it can be useful for generating labeled training data for use by a statistical learning system, as discussed below.

3. Many variations on this basic theme are possible: if the final observation is obtained by stochastically selecting one of the proposed output vectors, then this becomes a "mixture of mixtures" which reduces to a large flat mixture model or quantizer with a number of codebook entries equal to the product of the codebook sizes of the constituent quantizers. In Zemel's Cooperative Vector Quantization (CVQ) model (Hinton and Zemel, 1994), the proposals are combined *linearly* (either with the same coefficients across all dimensions or with different coefficients on each dimension) to produce the final output for each case. Zemel has also proposed a different model, called Multiple-Cause Vector Quantization (MCVQ) (Ross and Zemel, 2003); in MCVQ each component (dimension) of the observation vector also stochastically selects which vector quantizer is to provide its value. Each observation is then a noisy composite of the proposed values from each vector quantizer. This is like a mixture of mixtures but where each dimension makes a separate choice about which mixture to select. MAXVQ has similarities to each of MCVQ and CVQ. Unlike MCVQ, in MAXVQ the composite is not made by having each output dimension select a quantizer. Similar to CVQ there is a single fixed function which is applied to the proposed vectors from each quantizer to generate the final output. However, in CVQ this function implements a weighted sum, while in MAXVQ it implements an elementwise maximum.)

4. Much of the work described in this section was performed in collaboration with Kannan Achan at the University of Toronto.

5. A standard trick in traditional speech applications of HMMs is to use "left-to-right" transition matrices which are a special case of the type of constraints investigated in this section. However, left-to-right (Bakis) HMMs force state trajectories that are inherently one-dimensional and uni-directional whereas here we also consider higher dimensional topology and free omni-directional motion.

6. The observed symbol sequence must be "informative enough" to reveal the map structure (this can be quantified using the idea of *persistent excitation* from control theory).

7. Consider a network of units which compete to explain input data points. Each unit has a position in the output space as well as a position in a lower dimensional topology space. The winning unit has its position in output space updated towards the data point; but also the recent (in time) winners have their positions in topology space updated towards the topology space location of the current winner. Such a rule works well, and yields topological maps in which *nearby units code for data that typically occur close together in time*. However it cannot learn many-to-one maps in which more than one unit at different topology locations have the same (or very similar) outputs.

8. Articulator positions do not provide complete state information. For example, the excitation signal (voiced or unvoiced) is not captured by the bead locations. They do, however, provide much important information; other state information is easily accessible directly from acoustics.

9. Model structure (dimensionality and number of states) is set using cross validation.

10. An easier way (which we have used previously) to find good minima is to initialize the models using the articulatory data themselves. This does not provide as impressive

"structure discovery" as annealing but still yields a system capable of inverting acoustics into articulatory movements on previously unseen test data. First, a constrained HMM is trained on just the articulatory movements; this works easily because of the natural geometric (physical) constraints. Next, we take the distribution of acoustic features (LSFs) over all times (in the training data) when Viterbi decoding places the model in a particular state and use those LSF distributions to initialize an equivalent acoustic constrained HMM. This new model is then retrained until convergence using Baum-Welch.

Achan, Kannan, Roweis, Sam, and Frey, Brendan (2004). A segmental hmm for speech waveforms. Technical Report UTML-TR-2004-001, University of Toronto.

Blackburn, S. and Young, S. (1996). Pseudo-articulatory speech synthesis for recognition using automatic feature extraction from x-ray data. In *ICSLP 1996 v.2*, volume 2, pages 969–972.

Brown, Guy J. and Cooke, Martin P. (1994). Computational auditory scene analysis. *Computer Speech and Language*, 8.

Carreira-Perpinan, Miguel (2000). Reconstruction of sequential data with probabilistic models and continuity constraints. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12.

Cauwenberghs, Gert (1999). Monaural separation of independent acoustical components. In *IEEE Symposium on Circuit and Systems (ISCAS'99)*. IEEE.

Chennoukh, S., Sinder, D., Richard, G., and Flanagan, J. (1997). Voice mimic system using an articulatory codebook for estimation of vocal tract shape. In *Eurospeech 1997*, Rhodes, Greece.

Ephraim, Y., Malah, D., and Juang, B.H. (1989). On the application of hidden markov models for enhancing noisy speech. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37.

Gales, M. and Young, S. (1996). Robust continuous speech recognition using parallel model combination. *IEEE Transactions on Speech and Audio Processing*, 4(5):352–359.

Green, P., Barker, J., Cooke, M.P., and Josifovski, L. (2001). Handling missing and unreliable information in speech recognition. In *AISTATS*.

Hinton, Geoffrey and Zemel, Richard (1994). Autoencoders, minimum description length, and helmholtz free energy. In *Advances in Neural Information Processing Systems (NIPS)*, volume 6. MIT Press.

Jojic, Nebojsa and Frey, Brendan (2000). Topographic transformation as a discrete latent variable. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12. MIT Press.

Logan, Beth and Moreno, Pedro (1998). Factorial hmms for acoustic modeling. In *ICASSP 1998*. IEEE.

Nix, D. and Hogden, J. (1999). Maximum likelihood continuity mapping: An alternative to HMMs. In *Advances in Neural Information Processing Systems (NIPS)*, volume 11. MIT Press.

Plante, F, Ainsworth, W.A., and Meyer, G.F (1995). A pitch extraction reference database. In *Eurospeech 1995*.

Ramsay, G. and Deng, L. (1994). A stochastic framework for articulatory speech recognition. *Journal of the Acoustical Society of America*, 95(5):2873.

Reyes, Manuel, Raj, B., and Ellis, Dan (2003). Multi-channel source separation by factorial hmms. In *ICASSP 2003*. IEEE.

Ross, David and Zemel, Richard (2003). Multiple cause vector quantization. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15. MIT Press.

Roweis, Sam (2000). Constrained hidden markov models. In *Advances in Neural Information Processing Systems (NIPS)*, volume 12. MIT Press.

Roweis, Sam (2001). One microphone source separation. In *Advances in Neural Information Processing Systems (NIPS)*, volume 13. MIT Press.

Roweis, Sam and Alwan, Abeer (1997). Towards articulatory speech recognition. In *Eurospeech 1997*, volume 3, pages 1227–1230, Rhodes, Greece.

S., Roucos. and Wilgus., A. M. (1985). High quality time-scale modification for speech. In *ICASSP 1985*. IEEE.

Schroeter, J. and Sondhi, M. (1994). Techniques for estimating vocal tract shapes from the speech signal. *IEEE Transactions on Speech and Audio Processing*, 2(1 p2):133–150.

Smyth, P. (1997). Clustering sequences with hidden Markov models. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 9, pages 648–654. MIT Press.

Varga, A.P. and Moore, R.K. (1990). Hidden markov model decomposition of speech and noise. In *ICASSP 1990*, pages 845–848. IEEE.

Wan, Eric.A. and Nelson, Alex.T. (1998). Removal of noise from speech using the dual ekf algorithm. In *ICASSP 1998*. IEEE.

Westbury, J.R. (1994). X-ray microbeam speech production database user's handbook. Technical report, University of Wisconsin, Madison.