A Sub-optimal Viterbi-like Search for Linear Dynamic Models Classification

Dimitris Oikonomidis, Vassilis Diakoloukas and Vassilis Digalakis

Department of Electronics & Computer Engineering
Technical University of Crete, GR-73100 Chania, Greece
{doikon,vdiak,vas}@telecom.tuc.gr

Abstract

This paper describes a Viterbi-like decoding algorithm applied on segment-models based on linear dynamic systems (LDMs). LDMs are a promising acoustic modeling scheme which can alleviate several of the limitations of the popular Hidden Markov Models (HMMs). There are several implementations of LDMs that can be found in the literature. For our decoding experiments we consider general identifiable forms of LDMs which allow increased state space dimensionality and relax most of the constraints found in other approaches. Results on the AURORA2 database show that our decoding scheme significantly outperforms standard HMMs, particularly under significant noise levels.

Index Terms: linear dynamic models, speech recognition, acoustic modeling

1. Introduction

Hidden Markov models (HMMs) have proved to be a very successful approach to automatic speech recognition. Their strength is mainly attributed to their tractable mathematical framework and the well developed algorithms for training and recognition that are available. However, there are still many limitations, derived from fundamental assumptions which are inappropriate from a speech-modelling viewpoint and reduce the potential for further improvement. For instance, the independence assumption does not take into account the fact that a speech signal is produced by a continuously moving physical system such as the vocal tract.

The segment-based models [4] represent sequences of several consecutive acoustic feature vectors and can therefore explicitly characterize the relationship between these vectors in the time-domain. Various approaches introduced in the past fall into the family of the segment-base modeling schemes. One such approach is based on Linear Dynamic Models (LDMs). Although LDMs have been applied in the past under certain forms for speech recognition, their strength was limited since several modeling restrictions in the construction of their parameter form were applied to ensure system’s stability[2],[3],[5].

A more general identifiable form of LDM, was first applied in speech recognition in [1]. The system is a multivariate state-space linear dynamic model in a canonical form that ensures stability, controllability and identifiability [6]. Based on this general form and an appropriate element-wise Maximum Likelihood (ML) reestimation algorithm it is possible to train a model which will accurately capture the time-domain variability, both within a segment or across segments.

An acoustic model is useful in real-life applications only if it is accompanied with appropriate and efficient decoding algorithms. In this paper we go a step forward and present an efficient decoding algorithm for the aforementioned general Linear Dynamic Model. Most of the decoding approaches in speech recognition can be categorized into one of the following categories. The first category includes all the hypothesis search techniques on a tree basis like the A* search algorithm [3] or the split and merge algorithm [2]. Such algorithms are considered sub-optimal since they evaluate a relatively small portion of the potentially complete paths. Alternatively, we implement a Viterbi-like search algorithm on our segment-based model. When applied on HMMs, Viterbi is based on the memoryless property of the model dynamics, thus the most likely sequence can be obtained. However this is not true for segment-models. Thus, a sub-optimal Viterbi-like search algorithm is proposed. Viterbi-like algorithms have been also used for the decoding of other modeling schemes such as the trajectory HMMs[8].

The remaining of the paper is organized as follows. Section 2 defines the Linear Dynamic Model that we use. In section 3 the Viterbi-like decoding algorithm is presented. The experimental setup and results are shown in section 4. Concluding remarks are presented in the final section.

2. The General Identifiable Linear Dynamic Models

The LDM is described from the following pair of equations

\[ x_{k+1} = Fx_k + w_k \]
\[ y_k = Hx_k + v_k \]

where the state \( x_k \) at time \( k \) is a \((n \times 1)\) vector, the observation \( y_k \) is \((m \times 1)\) and \( w_k, v_k \) are uncorrelated, zero-mean Gaussian vectors with covariances

\[ E\{w_k w_k^T\} = P_{\delta_{kl}} \]
\[ E\{v_k v_k^T\} = R_{\delta_{kl}} \]

In the above equation \( \delta_{kl} \) denotes the Kronecker delta and \( T \) denotes the transpose of a matrix. The initial state \( x_0 \) is Gaussian with known mean and covariance \( \mu_0, \Sigma_0 \). Equation (1) describes the state dynamics, while (2) shows a prediction of the observation based on the state estimation.

The parametric structure of our multivariate state-space model has the following identifiable canonical form for the case in which \( x_k \) is a \( 5 \times 1 \) vector and the observation vector \( y_k \) is a...
3 × 1 vector.

\[
F = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 \\
\times & \times & \times & \times & \times \\
\times & \times & \times & \times & \times \\
0 & 0 & 0 & 1 & \times \\
\times & \times & \times & \times & \times
\end{bmatrix}
\]

(5)

\[
H = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0
\end{bmatrix}
\]

(6)

The number of rows with ‘\times’ s in \(F\) represents the free parameters of the matrix and equals the size of the output vector \(m\). The ones in matrix \(H\) are equal to the number of the rows in \(F\) that are filled with free parameters, and their position is related to the location of these rows in \(F\).

To construct the form of the state transition matrix \(F\) we follow the process described in [6]. First, we set its elements along the superdiagonal equal to one and the remaining elements are zeroed. Then, we choose arbitrarily the \(m\) row numbers \(r_1\) to be filled with free parameters, where \(i = 1, \ldots, m\). There is only one constraint, that \(r_0 = n\), where \(m\) denotes the dimension of the observation and \(n\) the dimension of the state vector. In addition, we set \(r_0 = 0\).

The observation matrix \(H\) is then constructed as follows. First, we define \(H\) to be \(m \times n\) in size and filled with zeros. Then we set each row \(i = 1, \ldots, m\) of the \(H\) matrix to have a one in column \(c_i = r_{i-1} + 1\). For instance, for the example shown in (5) and (6) we get:

- \(r_1 = 2 \Rightarrow c_1 = r_{1-1} + 1 = r_0 + 1 = 1\)
- \(r_2 = 3 \Rightarrow c_2 = r_{2-1} + 1 = r_1 + 1 = 3\)
- \(r_3 = 5 \Rightarrow c_3 = r_{3-1} + 1 = r_2 + 1 = 4\)

Hence, the observation matrix \(H\) will have ones in columns 1, 3 and 4 for its rows 1, 2 and 3, respectively.

3. Sub-optimal Viterbi-like search

The Viterbi algorithm [7], originally devised as an error-correcting scheme for convolution codes, is a well known algorithm used in HMMs to find the best state sequence. It can be seen as a modified forward algorithm, where instead of summing up the probabilities from all the different paths coming in the same destination state, we pick only the best path. The efficient computation of the optimal path using dynamic programming techniques and a trellis structure, in a time-synchronous fashion, has made Viterbi algorithm the predominant search strategy for continuous speech recognition.

With the addition of beam search, where at each time, only nodes attaining a score close to the current highest score are kept in the beam while the rest are discarded (pruned), time-synchronous Viterbi beam search can handle tasks of all sizes.

In LDMs however, the “memoryless” property of HMMs, where the probability of being in state \(j\) at time \(t\) depends only upon the state at time \(t−1\), does not hold. Here the process evolves dynamically and the state at time \(t\) depends upon all the previous time slices. Equations (1), (2) represent a standard Kalman filter and their calculation involves a Kalman forward recursion. Nevertheless, in LDMs, the analogue of HMMs’ state is the segment. It is not the best state sequence what we search, but the best segmentation among all possible segmentations. This can be seen in Figure 1, where an LDM with 4 segments is presented. Each segment \(s_i\) has a different set of parameters \(\{F_i, H_i, P_i, R_i\}\). The typical approach, also used in our previous work [1], is to derive this segmentation by performing a forced alignment with an HMM model. After obtaining this “optimal” segmentation, we could use the aforementioned set of parameters with the corresponding observations, to compute the log-likelihood of the whole model.

In general, the state of the LDM is the augmented vector \([x_k, s_k]\). Hence, an exact Viterbi search algorithm should retain, at every point \(k\) in time, all possible histories leading to a different \([x_k, s_k]\) combination. In practice, this means that an exact search should retain at each point in time \(k\), all possible segment alignments, since each one of them will produce a different state \(x_k\).

This segmentation can be achieved with the LDM alone, using a sub-optimal Viterbi-like algorithm as shown in Figure 2. In each iteration, the Kalman forward recursion takes a state vector \(x_{k−1}\), an error covariance matrix \(\Sigma_{k−1}\) and an observation \(y_k\) and produces a new estimate for \(x_k\) and \(\Sigma_k\). The parameters of this Kalman iteration are the set \(\{F_i, H_i, P_i, R_i\}\) (denoted in the Figure as \(F_i\) for brevity), according to which segment we are in. During this process, the log-likelihood is also computed as

\[
L(y_k, \theta) = - \left\{ \log [\Sigma_{k−1}(\theta)] + e_k^2(\theta)\Sigma_{k−1}^{-1}(\theta)e_k(\theta) \right\} + C
\]

where \(\theta\) denotes the set of parameters that characterize the segment, \(e_k^2(\theta)\), \(\Sigma_{k−1}(\theta)\) is the prediction error and its covariance obtained from the Kalman filter equations and \(C\) is a constant.

Obviously, the number of paths grows exponentially. However, we can apply a Viterbi-like criterion and retain at each node, only the path with the highest score from all the incoming paths. The computation grows according to the (time) depth we expand the nodes, before we take a decision. For depth \(d = 1\), two paths arrive at each node and we need to do two Kalman iterations, for \(d = 2\) there are four paths and we need to do eight Kalman iterations. For example, consider the node denoted as \(A\) in Figure 2. If the depth equals 1, the incoming paths are the \(B \rightarrow A\) and \(C \rightarrow A\). This case is equivalent to the hypothesis that the augmented vector \([x_k, s_k]\) can be adequately approximated by the segment \(s_i\) for the purpose of the search. For the same node \(A\) and depth \(d = 2\) the incoming paths are: \(D \rightarrow B \rightarrow A\), \(E \rightarrow B \rightarrow A\), \(E \rightarrow C \rightarrow A\) and \(F \rightarrow C \rightarrow A\). Note that in this case, \(D \rightarrow B\) and \(E \rightarrow B\) cannot merge in one node because each path has a different state \(x_k\), so both must be expanded to \(A\).

In general for depth \(d\), there are \(2^d\) paths arriving at each node, so we need to do \(2^d\) Kalman filter propagations spanning
Figure 2: Trellis structure of an LDM with 4 segments.

A length of $d$ frames each, in order to compute the highest score. The complexity is $O(d^2dNT)$, where $N$ is the number of LDM model segments, $d$ is the depth we expand the nodes and $T$ is the number of observations. The algorithm is described below:

1. Initialization (for $t = 1$ and $\forall 1 \leq j \leq N$):
   
   $\delta_1(s_j) = L(y_1, s_j, x_0^*)$
   
   $\psi_1(s_j) = 0$

2. Recursive step (for $t = 2, ..., T$ and $\forall 1 \leq j \leq N$):

   $\delta_t(s_j) = \max_i \left\{ \sum_{k=t-d+1}^t L(y_k, s_j, x_k^*) + \delta_{t-1}(s_i) \right\}$
   
   $\psi_t(s_j) = \arg \max_i \left\{ \sum_{k=t-d+1}^t L(y_k, s_j, x_k^*) + \delta_{t-1}(s_i) \right\}$

3. Termination (for $t = T$):

   $P^* = \max_i \{ \delta_T(s_i) \}$
   
   $s_T^* = \arg \max_i \{ \delta_T(s_i) \}$

4. Backtrace (for $t = T-1, ..., 1$):

   $s_t^* = \psi_{t+1}(s_{t+1}^*)$}

In the first step we compute the score $\delta_1(s_j)$ for each node $s_j$. The log-likelihood $L(y_k, s_j, x_{k-1}^*)$, which is estimated as in equation (7), is a function of the observation $y_k$ at time $k$, of the set of parameters of the current segment ($s_j$) and the output state estimation of the previous Kalman forward iteration coming from node $s_i$ ($x_{k-1}^*$). In the initialization step where $t = 1$ the previous output state is the initial state $x_0$ which is estimated during training from the observations. At every recursive step, we expand the paths of each node up to depth $d$, take the highest score of all the paths arriving in a certain node and record this path as the best one in the backtrace table.

Figure 3: Trajectories of the true and predicted observations.

Figure 3 shows that our Viterbi-like algorithm can better track observation trajectories compared to the standard HMM alignments. The blue lines are the trajectories of two MFCCs, $C_1$ and $C_3$ of a test utterance. The red lines show the predicted observation vectors using our model and the HMM alignments, while the green lines are the prediction with our Viterbi-like search algorithm. Clearly, the use of a better segmentation allows the model to track better the true trajectory.

4. Experiments

We have performed a series of word-classification experiments in order to validate our LDM system for speech recognition and evaluate the estimation algorithm. Specifically, we used the AURORA2 speech database[9], which is a connected digit corpus based on TIDIGITS, downsamped to 8KHz and with several types of noise artificially added at several SNRs. The front-end uses a total of 12 Mel-warped cepstral coefficients plus energy and $c_0$.

We used 11 word-models corresponding to the words in the AURORA2 corpus (digits 1 to 9, zero and oh). Each word-model has a number of time-invariant regions (segments) ranging from 2 to 8, depending on the phonetic transcription of each word. Table 1 shows the number of regions for each word-model that we considered.

<table>
<thead>
<tr>
<th></th>
<th>one</th>
<th>two</th>
<th>three</th>
<th>four</th>
<th>five</th>
<th>six</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>seven</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>2</td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Number of regions for each word-model
The first issue in implementing the dynamical system is the dimensionality of the state-space. Based on the general canonical forms of the LDM that we examine, the size of the state-vector can be equal or larger than the size of the observation vector. When the state and observation vectors are at equal size, the observation matrix becomes the identity matrix and the observation vector is just a noisy version of the state vector.

Another important issue is the initialization of system parameters. The noise covariance matrices are initialized randomly, while the initial state-transition matrices, and the covariance of the initial state $x_0$ are directly estimated from the observations. The initialization process is straightforward for equally sized states and observations. However, when the state dimension is larger then the observation dimension it is necessary to concatenate the observation vectors with more features and artificially construct observation vectors equally sized to the states.

For our experiments additional MFCC coefficients were added to the original observation vector and the enlarged observations were used to obtain the initial statistics for the estimation of the model parameters.

As far as the classification is concerned, we obtain true word-boundaries produced by an HMM and keep them fixed. Decoding is performed using our novel viterbi-like sub-optimal search over all possible segmentations of each word-model.

For our experiments, we used a clean training set consisting of 104 gender-balanced speakers and 8444 sentences. The trained LDM had equally-sized state and observation vectors. The evaluation was done on a separate test set defined as the AURORA2-A test set, with subway additive noise at several SNRs, which consisted of 1000 sentences from the training speakers.

Table 2: Word-classification performance of the LDM system

<table>
<thead>
<tr>
<th>AURORA2</th>
<th>HMM</th>
<th>LDM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subway</td>
<td>Mfcc,E</td>
<td>HMM-align</td>
</tr>
<tr>
<td>clean</td>
<td>97.19%</td>
<td>97.85%</td>
</tr>
<tr>
<td>SNR20</td>
<td>90.91%</td>
<td>92.53%</td>
</tr>
<tr>
<td>SNR15</td>
<td>80.09%</td>
<td>85.93%</td>
</tr>
<tr>
<td>SNR10</td>
<td>57.68%</td>
<td>71.30%</td>
</tr>
<tr>
<td>SNR5</td>
<td>36.01%</td>
<td>46.72%</td>
</tr>
</tbody>
</table>

Table 3: Word-classification performance for LDMs with different state-size setups

<table>
<thead>
<tr>
<th>AURORA2</th>
<th>State Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subway</td>
<td>14</td>
</tr>
<tr>
<td>clean</td>
<td>97.73%</td>
</tr>
<tr>
<td>SNR20</td>
<td>93.52%</td>
</tr>
<tr>
<td>SNR15</td>
<td>89.68%</td>
</tr>
<tr>
<td>SNR10</td>
<td>77.21%</td>
</tr>
<tr>
<td>SNR5</td>
<td>53.66%</td>
</tr>
</tbody>
</table>

dimension may result in more accurate but condition-specific models.

5. Conclusions

In this paper, we presented a new sub-optimal Viterbi-like decoding algorithm suitable for Linear Dynamic Models. Our experiments showed that it significantly outperforms the standard HMM segmentation, especially when large levels of noise are present. This algorithm could be used as the base for the development of a decoder for LDMs.

6. References


