THE BAUM-WELCH ALGORITHM

Ramasubramanian H. Sundaram

Department for Electrical and Computer Engineering
Mississippi State University, Mississippi State, MS 39762
sundaram@isip.msstate.edu

ABSTRACT

In a typical speech recognition system it is assumed that the sequence of observed speech vectors corresponding to a word sequence is generated by a parametric model. If a Markov model is used then the problem of finding the conditional probability of acoustic evidence given the word is replaced by estimation of model parameters of the Markov model. Given a set of training examples the parameters of the model can be estimated by a robust and efficient reestimation procedure. This process is called Acoustic Training and one such procedure used to reestimate the parameters of the model is Baum-Welch Reestimation algorithm. This paper will focus on the details of this algorithm and its practical implications.

1. INTRODUCTION

A speech recognizer maps the input speech vectors with the word sequence that needs to be recognized. An inherent problem with this is that the mapping may not be one-to-one since different word sequences can have similar speech vectors. Hence the problem is approached with a statistical outlook involving probabilities[1]. Given the input acoustic vector the recognizer chooses the most likely word sequence. If the speech vectors or observations is represented by $O = o_1, o_2, \ldots, o_t$ where o_t is the speech vector observed at time t. The output of the speech recognizer will be

$$\hat{W} = \arg max P(w_i/O) \quad (1)$$

where w_i is the ith vocabulary word.

Using Bayes rule this probability is computed as,

$$P(w_i/O) = \frac{P(O/w_i)P(w_i)}{P(O)} \quad (2)$$

If the probability $p(w_i)$ is known then the most probable spoken word depends on the likelihood P(O/wi). Since the dimensionality of the observation is large, direct computation of $P(O/w_i)$ is impractical. However, if the word production is assumed to be from a parametric model like an Markov model, then the estimation of $P(O/w_i)$ is replaced by a simpler problem of estimating the parameters of the model.

2. HIDDEN MARKOV MODELS

2.1. Overview

In a typical speech recognition system, the words in the word sequence to be recognized are modelled using a parametric model called Hidden Markov Models (HMMs). It is assumed that the sequence of observed speech vectors corresponding to each word is generated by the model corresponding to that word[2].

A Markov model is a finite state machine which changes state once every time unit and at each time *t* a state *j* is entered and speech vector o_t is generated from the probability density function $b_j(o_t)$. Also, the manner in which the state transitions occur with the model is also probabilistic and is governed by a state-transition matrix. If in a Markov model the state sequence that produced the observation sequence is not known deterministically, then the Markov model is called Hidden Markov Model. Thus HMM's are double embedded stochastic process with an underlying stochastic process that is not directly observable but can be observed through another set of stochastic processes that produce the sequence of observations. Thus the elements of a HMM includes

N (the number of states in the model), M (the number of distinct observation symbols per state), A (state-transition probability distribution), B (observation sequence probability distribution) and π (initial state distribution).

2.2. Problems for HMM's:

Given the general form and elements of a HMM it can be easily seen that for the model to be useful for applications certain problems related to it need to be solved. In simplistic terms the three problems are

1. given the observation sequence O = (o_1,o_2,..., o_T) and a model $\lambda = (A, B, \pi)$, how do we efficiently compute P(O/λ), the probability of the observation sequence given the model.

2. Given the observation sequence O = (o_1,o_2,..., o_T) and the model λ, how do we choose a corresponding state sequence q = (q_1q_2...q_T) that is optimal in some sense.

3. How do we adjust the model parameters λ to maximize P(O/ λ)?

The third problem is what we are interested in and where we optimize the model parameters so as to describe how the observed sequence is best represented. This is also called the **Training Sequence Problem** since it is used to train the model. This is important for most applications as it allows to optimally adapt model parameters to observed data and in the process create best models for the recognizing the real data.

3. MATHEMATICAL SOLUTION

An attempt to solve the above problems in its direct way will not be computationally feasible even for relatively small values of states and observations in an HMM and. For example, for N = 5 and T = 100, we need 10^{72} computations to find the probability of observation sequence given the model[2]. A more efficient procedure called forward-backward procedure is used.

3.1. The forward-backward procedure

Before going into the details of how the model parameters are updated it is important to know how certain probabilities are estimated using the forward-backward procedure. This makes the problem computationally less expensive.

Consider the variable $\alpha_t(i)$ defined as

$$\alpha_t(i) = Pr((O_1, O_2, \ldots, O_t, i_t = q_i)/\lambda) \quad (3)$$

the probability of the partial observation sequence (until time t) and the state q_i at time t, given the model λ. This can be solved inductively as

$$\alpha_1(i) = \pi_i b_i(O_1) \text{ for } 1 \le i \le N \quad (4)$$

where π_i is the initial state distribution for the i^{th} state and $b_i(O_1)$ is the output probability distribution for state *i*. Then for subsequent time instants *t=1,2,..., T-1*

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}) \quad (5)$$

where a_{ij} is the state-transition probability for transition from state *i* to state *j*. And the probability of the observation given the model is given by

$$Pr(O/\lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad (6)$$

Equation (4) initiates the forward probabilities with the joint probability of being in the state q_i at T=1 and the initial observation O_1. Equation (5) represents how the state q_j is reached at time t+1 from the N possible states at time t. Since $\alpha_t(i)$ is the probability of the joint event that O_1O_2...O_t are observed and the state stops at q_i at time t, the product $\alpha_t(i)\, a_{ij}$ is then the probability of the joint event O_1O_2...O_t are observed and state q_j is reached at time t+1 via state q_i. Summing this over all possible N states at time *t* results in the probability of being at state q_j at time t+1. Finally, the probability of the output given the model is the sum of the terminal forward probabilities in the last time frame. The implementation of the recursive computation of the forward variable leads to a lattice structure as

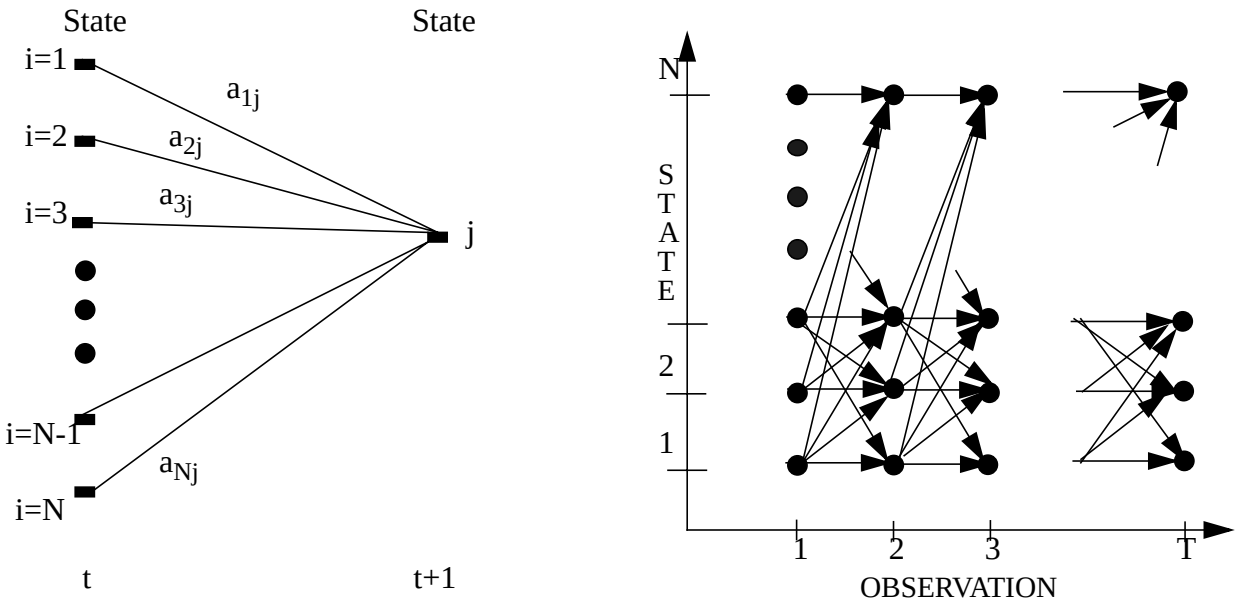

Figure 1: Lattice structure for the recursive computation of the forward variable

shown in Figure 1.

By using the forward method we obtain a computational savings of 69 orders of magnitude than with direct calculation. The key to this is that since there are only N states, all possible paths will merge into these N states no matter how long the observation sequence is. At each time instant the calculation involves N previous values calculated in the previous time instant because each of the N states is reached from the same N states at the previous time instant. This helps in avoiding redundant calculations and hence the computational savings.

Similarly, we can consider the backward variable as

$$\beta_t(i) = Pr(O_{t+1}, O_{t+2}, \ldots, O_T / i_t = q_i, \lambda) \quad (7)$$

which is the probability of the partial observation sequence from t+1 to the end given state q_i at time *t* and the model. The backward variable can also be solved inductively as

$$\beta_T(i) = 1 \text{ for } 1 \le i \le N \quad (8)$$

and

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(O_{t+1}(j)) \quad (9)$$

As for forward variable, the backward variable is also computed recursively and follows a lattice structure.

4. BAUM-WELCH REESTIMATION

The forward-backward procedure defined in the previous section helps in reestimating the model parameters. There is no analytical method to solve for the model parameter set that maximizes the probability of the observation sequence in a closed form. So we use an iterative procedure to reestimate the models so that the likelihood of the observation sequence given the model increases. The Baum-Welch method is one such method for reestimating the parameters[3].

4.1. Reestimation

Let $\xi_t(i, j)$ denote the probability of being in state *i* at time *t* and in state *j* at time t+1, given the model and the observation i.e.

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j / O, \lambda) \quad (10)$$

From the definition of forward and backward variables $\xi_t(i, j)$ can be expressed in the form

$$\xi_t(i, j) = \frac{P(q_t = i, q_{t+1} = j, O/\lambda)}{P(O/\lambda)} \quad (11)$$

$$= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(O/\lambda)} \quad (12)$$

$$= \frac{\alpha_T(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_T(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \quad (13)$$

Also, let $\gamma_t(i)$ be defined as the probability of being in state *i* at time t, given the entire observation sequence and the model. This can be related to $\xi_t(i, j)$ by summing over as

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i, j) \quad (14)$$

If we sum $\gamma_t(i)$ over the time index t, it can be interpreted as the expected number of times that state *i* is visited or in other words, expected number of transitions made from state i. In the same manner, summation of $\xi(i, j)$ over *t* is the expected number of transitions from state *i* to state j. Using the concept of event occurrences we can reestimate the parameters of the HMM namely π (the initial state transition probabilities), A (state transition probabilities) and B (output probability distribution). This can be represented as

$$\bar{\pi}_j = \gamma_1(i) \quad (15)$$

= number of times in state *i* at time *t* = 1

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (16)$$

= expected number of transitions from state *i* to state j / expected number of transitions from state i

and,

$$\bar{b}_j(k) = \frac{\sum_{t=1}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad (17)$$

= expected number of times in state *j* and observing symbol v_k / expected number of times in state j

If we define the current model as $\lambda = (A, B, \pi)$ and use (15), (16) and (17) to compute reestimated parameters $\bar{\lambda} = (\bar{A}, \bar{B}, \bar{\pi})$ then it can be shown[1] that either the initial model λ defines a critical point of the likelihood function in which case $\bar{\lambda} = \lambda$ or the model $\bar{\lambda}$ is more likely than the model λ in the sense that $P(O/\bar{\lambda}) > P(O/\lambda)$. If we above procedure is done over several iterations then we can improve the probability of the observation sequence being observed from the model.

4.2. Practical Issues:

Though we have reduced the computations involved in reestimation process using the forward-backward procedure the computations become exhaustive as the input utterances on which the model is trained becomes long. This is because as the utterance length increases the number of models that are present in that utterance also increases leading to a situation where the number of forward and backward probabilities that need to be calculated is really large.

For a speech recognition system used a typical left-right model topology this hurdle can be overcome by employing the fact that not all states can be reached

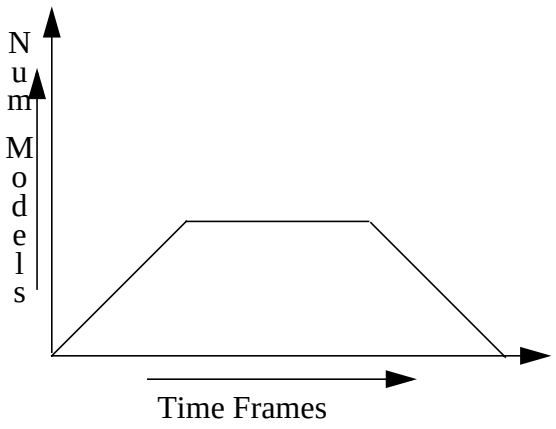

Figure 2: Illustration of number of models hypothesized against the progression of time.

at all times.If the input utterance is known that one need not hypothesize all the states at the first time instant. This is because it is clearly known that the only state that correspond to the input vector is the first state of the first model. So windowing of the models is done during every time frame to choose only those models that are likely to be hypothesized in this time frame. Figure 2 shows how the number of models hypothesized varies at various instants. It can be seen that at the beginning only a few models are hypothesized but as the time progresses the number of models also increases. It saturates and remains constant and as the time progresses to the last frame the number of models start to decreased.

Another way of saving on computations is by pruning off those models that have low probabilities. If the backward probability for a model is very low and falls below a threshold then the corresponding forward probability for that model is not calculated. In other words the model is not hypothesized in this time frame and is said to be pruned. This method of pruning based on the backward probabilities is call **Beta-pruning.**

5. CONCLUSION

The Baum-Welch method of training differs from the Viterbi training in the sense that the Baum-Welch algorithm assumes that any state can occur at any time with some probability and updates the model parameters based on these probabilities rather than choose a single best state sequence and update the model parameters based on the state sequence. Usually continuous distribution gaussian mixtures are used to model the input vector and the means and covariances are updated after every pass of training. The iteration is done over 4 or 5 passes for the models to be sufficiently trained. Also there should be sufficient data for all the models to get trained properly

6. REFERENCES

[1] Frederick Jelinek, “Statistical Methods for Speech Recognition,” The MIT Press, Cambridge, England,1997

[2] L.R.Rabiner, B.H. Juang, “An Introduction to Hidden Markov Models,” IEEE ASSP Magazine, vol. 3, February 1986

[3] L.R.Rabiner, B.H.Juang,“Fundamentals of Speech Recognition,” Prentice Hall Signal Processing Series, 1993