Disclaimer

- The material provided in this document is not my original work and is a summary of some one else's work(s).
- A simple Google search of the title of the document will direct you to the original source of the material.
- I do not guarantee the accuracy, completeness, timeliness, validity, non-omission, merchantability or fitness of the contents of this document for any particular purpose.
- Downloaded from najeebkhan.github.io

Theory and Implementation of Hidden Markov Models

Outline

- Introduction
- Discrete Time Markov Processes
- Extensions to Hidden Markov Models
- The Three Basic Problems of HMMs
- Types of HMMs
- Continuous Observation Densities in HMMs
- Autoregressive HMMs
- Variants on HMM Structures
- Inclusion of Explicit State Duration Density in HMMs
- Optimization Criterion– ML, MMI, and MDI
- Comparisons of HMMs
- Implementation Issues for HMMs
- Improving the Effectiveness of Model Estimates
- Model Clustering and Splitting
- HMM System for Isolated Word Recognition

Implementation Issues for HMMs

- In this section we deal with several practical implementation issues including
  - Scaling
  - Multiple observation sequences
  - Initial parameter estimates
  - Missing data
  - Choice of model size and type

Scaling

- $\alpha_t(i)$ consists of the sum of a large number of terms, each of the form

$$\left(\prod_{s=1}^{t-1} a_{q_s q_{s+1}} \prod_{s=1}^{t} b_{q_s}(\mathbf{O}_s) \right)$$

- Since each a and b term is less than 1, it can be seen that as t starts to get big, each term of $\alpha_t(i)$ starts to head exponentially to zero
- For sufficiently large t the dynamic range of the $\alpha_t(i)$ computation will exceed the precision range of essentially any machine
- Hence the only reasonable way of performing the computation is by incorporating a scaling procedure

Scaling

- The basic scaling procedure which is used is to multiply $\alpha_t(i)$ by a scaling coefficient that is independent of i
- The goal is to keep the scaled $\alpha_t(i)$ within the dynamic range of the computer for $1 \leq t \leq T$
- A similar scaling is done to the $\beta_t(i)$ coefficients
- At the end of the computation, the scaling coefficients are canceled out exactly

Scaling

- Consider the re-estimation formula for the state transition coefficients a_{ij}

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, \beta_{t+1}(j)}.$$

- For a fixed t, we first compute

$$\alpha_t(i) = \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\, a_{ij} b_j(O_t). \qquad c_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)}$$

- Then

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\, a_{ij} b_j(O_t)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \hat{\alpha}_{t-1}(j)\, a_{ij} b_j(O_t)}$$

Scaling

$$\hat{\alpha}_{t-1}(j) = \left(\prod_{\tau=1}^{t-1} c_\tau\right)\alpha_{t-1}(j)$$

$$\hat{\alpha}_t(i) = \frac{\sum_{j=1}^{N} \alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1} c_\tau\right) a_{ij} b_j(O_t)}{\sum_{i=1}^{N}\sum_{j=1}^{N} \alpha_{t-1}(j)\left(\prod_{\tau=1}^{t-1} c_\tau\right) a_{ij} b_j(O_t)} = \frac{\alpha_t(i)}{\sum_{i=1}^{N} \alpha_t(i)}$$

- Each $\alpha_t(i)$ is effectively scaled by the sum over all states of $\alpha_t(i)$
- The same scale factors are used for betas as are used for the alphas

$$\hat{\beta}_t(i) = c_t \beta_t(i)$$

Scaling

- Each scale factor effectively restores the magnitude of the $\alpha_t(i)$ terms to 1
- The magnitudes of the α and β terms are comparable
- Using the same scaling factors on the β's as was used on the α's is an effective way of keeping the computation within reasonable bounds

Scaling

- In terms of the scaled variables we see that the re-estimation formula becomes

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i)\, a_{ij} b_j(O_{t+1})\, \hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \hat{\alpha}_t(i)\, a_{ij} b_j(O_{t+1})\, \hat{\beta}_{t+1}(j)}$$

$$\hat{\alpha}_t(i) = \left[\prod_{s=1}^{t} c_s \right] \alpha_t(i) = C_t \alpha_t(i)$$

$$\hat{\beta}_{t+1}(j) = \left[\prod_{s=t+1}^{T} c_s \right] \beta_{t+1}(j) = D_{t+1} \beta_{t+1}(j).$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, D_{t+1} \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} C_t \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, D_{t+1} \beta_{t+1}(j)}$$

$$C_t D_{t+1} = \prod_{s=1}^{t} c_s \prod_{s=t+1}^{T} c_s = \prod_{s=1}^{T} c_s = C_T$$

Scaling

- In terms of the scaled variables we see that the re-estimation formula becomes

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} \hat{\alpha}_t(i)\, a_{ij} b_j(O_{t+1})\, \hat{\beta}_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \hat{\alpha}_t(i)\, a_{ij} b_j(O_{t+1})\, \hat{\beta}_{t+1}(j)}$$

$$\hat{\alpha}_t(i) = \left[\prod_{s=1}^{t} c_s\right] \alpha_t(i) = C_t \alpha_t(i)$$

$$\hat{\beta}_{t+1}(j) = \left[\prod_{s=t+1}^{T} c_s\right] \beta_{t+1}(j) = D_{t+1} \beta_{t+1}(j).$$

$$\overline{a_{ij}} = \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, D_{t+1} \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} C_t \alpha_t(i)\, a_{ij} b_j(O_{t+1})\, D_{t+1} \beta_{t+1}(j)}$$

Independent of t

$$C_t D_{t+1} = \prod_{s=1}^{t} c_s \prod_{s=t+1}^{T} c_s = \prod_{s=1}^{T} c_s = C_T$$

Scaling

- C_T cancels out in both the numerator and denominator and the exact re-estimation equation is therefore realized
- It should be obvious that the above scaling procedure applies equally well to re-estimation of the π or B coefficients
- If scaling is not performed at some instant t, the scaling coefficients c_t, are set to 1 at that time

Scaling

- To find P(OIλ) we cannot merely sum up the $\alpha_t(i)$ terms since these are scaled already
- However, we can use the property that

$$\prod_{t=1}^{T} c_t \sum_{i=1}^{N} \alpha_T(i) = C_T \sum_{i=1}^{N} \alpha_T(i) = 1$$

$$\prod_{t=1}^{T} c_t \cdot P(O|\lambda) = 1$$

$$P(O|\lambda) = \frac{1}{\prod_{t=1}^{T} c_t}$$

$$\log [P(O|\lambda)] = -\sum_{t=1}^{T} \log c_t$$

Scaling

- The log of P can be computed, but not P since it would be out of the dynamic range of the machine anyway
- When using the Viterbi algorithm to give the maximum likelihood state sequence, no scaling is required if we use logarithms

Multiple Observation Sequences

- The left-right model imposes constraints on the state transition matrix, and the initial state probabilities
- The major problem with left-right models is that one cannot use a single observation sequence to train the model (i.e. for re-estimation of model parameters)
- This is because the transient nature of the states within the model only allow a small number of observations for any state (until a transition is made to a successor state)

Multiple Observation Sequences

- In order to have sufficient data to make reliable estimates of all model parameters, one has to use multiple observation sequences
- We denote the set of K observation sequences as

$$\boldsymbol{O} = [\boldsymbol{O}^{(1)}, \boldsymbol{O}^{(2)}, \cdots, \boldsymbol{O}^{(k)}]$$

- The kth observation sequence is given by

$$\boldsymbol{O}^{(k)} = [O_1^{(k)}\ O_2^{(k)}\ \cdots\ O_{T_k}^{(k)}]$$

Multiple Observation Sequences

- We assume each observation sequence is independent of every other observation sequence, and our goal is to adjust the parameters of the model λ to maximize

$$P(O|\lambda) = \prod_{k=1}^{K} P(\mathbf{O}^{(k)}|\lambda) = \prod_{k=1}^{K} P_k.$$

- The re-estimation formulas for multiple observation sequences are modified by adding together the individual frequencies of occurrence for each sequence

Multiple Observation Sequences

$$\overline{a_{ij}} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, a_{ij} b_j(O_{t+1}^{(k)})\, \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(i)}$$

$$\overline{b_j}(\ell) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{\substack{t=1 \\ \text{s.t. } O_t = v_\ell}}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(i)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} \alpha_t^k(i)\, \beta_t^k(i)}$$

Multiple Observation Sequences

- Each observation sequence has its own scaling factor
- The key idea is to remove the scaling factor from each term before summing
- This can be accomplished by writing the re-estimation equations in terms of the scaled variables

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k - 1} \hat{\alpha}_t^k(i)\, a_{ij} b_j(O_{t+1}^{(k)})\, \hat{\beta}_{t+1}^k(j)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k - 1} \hat{\alpha}_t^k(i)\, \hat{\beta}_t^k(i)}$$

Initial Estimates of HMM Parameters

- In theory, the re-estimation equations should give values of the HMM parameters which correspond to a local maximum of the like1ihood function
- How do we choose initial estimates of the HMM parameters so that the local maximum is the global maximum of the likelihood function?
- Experience has shown that either random or uniform initial estimates of the π and A parameters is adequate for giving useful re-estimates of these parameters in almost all cases

Initial Estimates of HMM Parameters

- However, for the B parameters, experience has shown that good initial estimates are required
- Such initial estimates can be obtained in a number of ways, including
  - Manual segmentation of the observation sequence into states with averaging of observations within states
  - Maximum likelihood segmentation of observations with averaging
  - Segmental k-means segmentation with clustering

Effects of Insufficient Training Data

- There is often an insufficient number of occurrences of different model events to give good estimates of the model parameters
- If the training sequence is so small that it does not contain any occurances of $q_t=j$ and $o_t=v_k$ then $b_k(j)=0$
- The resultant model produces a zero probability for any observation sequence that actually includes $q_t=j$ and $o_t=v_k$

Effects of Insufficient Training Data

- The solutions to this problem are
  - Increase the size of the training set
  - Reduce the size of the model
  - Interpolate one set of parameter estimates with another set of parameter
- The simplest way is to use a floor

$$b_j(k) = \begin{cases} b_j(k), & \text{if } b_j(k) \geq \delta_b \\ \delta_b, & \text{otherwise} \end{cases}$$

$$U_{jk}(r,\ r) = \begin{cases} U_{jk}(r,r), & \text{if } U_{jk}(r,r) \geq \delta_u \\ \delta_u, & \text{otherwise} \end{cases}$$

Effects of Insufficient Training Data

- The remaining issues in implementing HMMs
  - Choice of model: Ergodic, left-right, or some other form
  - Choice of model size
  - Choice of observation symbols(D,C, single, mixture)
- There is no theoretically correct way of making these choices

Model Clustering and Splitting

- One of the basic assumptions in statistical modeling is that the variability in the observations from an information source can be modeled by the assumed statistical distribution
- Because of variability in the production or processing it is often expedient to consider using more than a single HMM to characterize the source

Model Clustering and Splitting

- There are two motivations behind this multiple HMM approach
  - Lumping together all the variability from inhomogeneous data sources leads to unnecessarily complex models, often yielding lower model accuracy
  - Some of the variability or rather the inhomogeneity in the source data, may be known a priori, thus warranting separate modeling of the source data sets

Model Clustering and Splitting

- Several generalized clustering algorithms exist all of which are suitable for the purpose of separating inconsistent training
- Each divided subgroup becomes more homogeneous and therefore is better modeled by a single HMM
- The nearest neighbor rule required for these clustering algorithms is simply to assign an observation sequence O to cluster i if

$$p(O \mid \lambda_i) = \max P(O \mid \lambda_j)$$

Model Clustering and Splitting

- An alternative to model clustering is to arbitrarily subdivide a given speech source into a large number of subclasses
- And then consider a generalized procedure for model merging based on source likelihood consideration
- For large vocabulary speech recognition we often try to build specialized units for recognition
- Triphones: 10,000

Model Clustering and Splitting

- The problem is how to determine which pairs of units should be merged
- Consider two models λ_a and λ_b corresponding to training observation sets O_a and O_b and the merged model λ_{a+b} corresponding to the merged observation sets $\{O_a, O_b\}$
- We can then compute the change in entropy (loss of information) resulting from the merged model as

$$\begin{aligned}\Delta H_{ab} &= H_a + H_b - H_{a+b} \\ &= -P(\mathbf{O}_a|\lambda_a)\log P(\mathbf{O}_a|\lambda_a) - P(\mathbf{O}_b|\lambda_b)\log P(\mathbf{O}_b|\lambda_b) \\ &\quad + P(\{\mathbf{O}_a, \mathbf{O}_b\}|\lambda_{a+b})\log P(\{\mathbf{O}_a, \mathbf{O}_b\}|\lambda_{a+b}).\end{aligned}$$

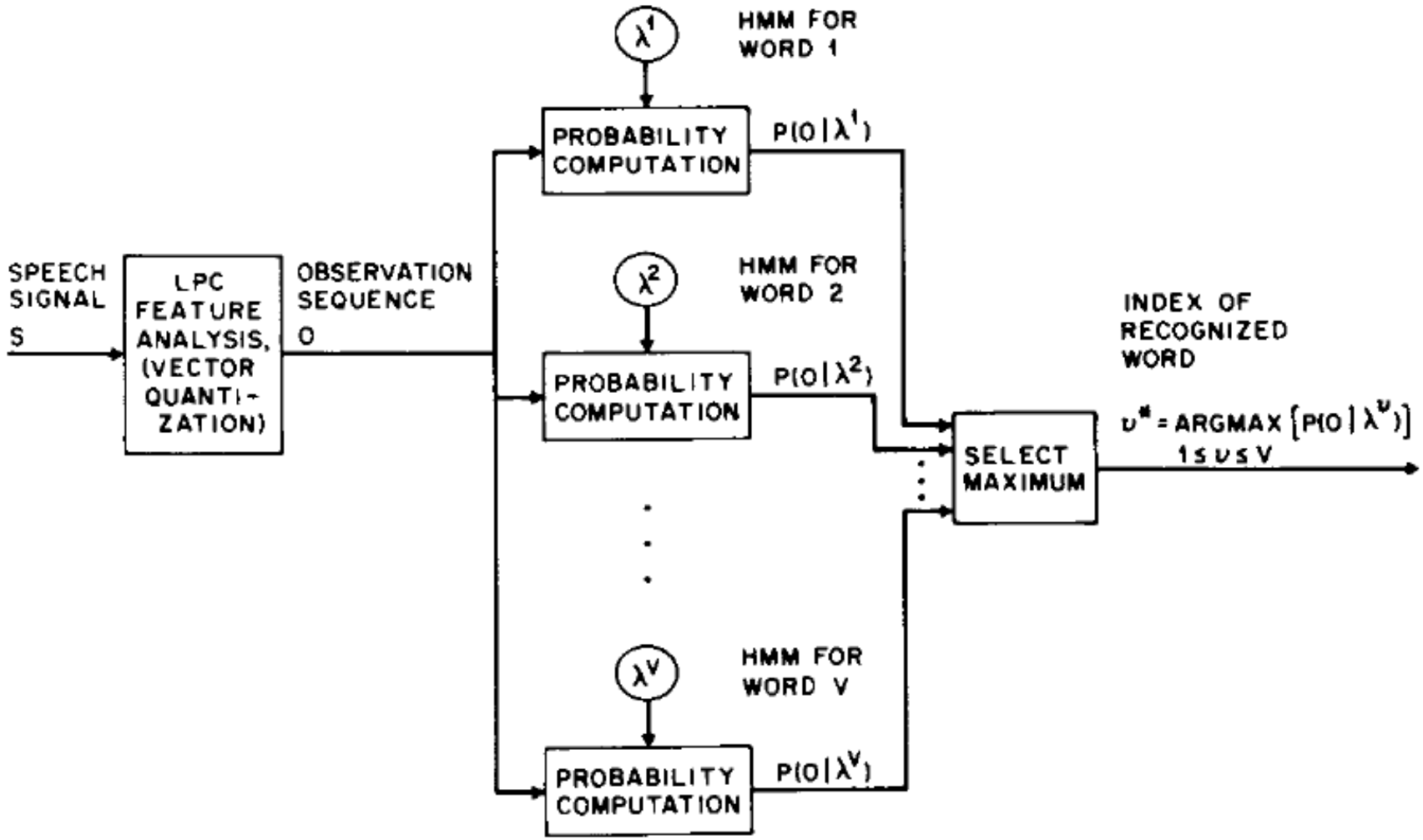
HMM System for Isolated Word Recognition

- Assume we have a vocabulary of V words to be recognized and that each word is to be modeled by a distinct HMM
- Assume that for each word in the vocabulary we have a training set of K occurrences of each spoken word
- Each occurrence of the word constitutes an observation sequence

HMM System for Isolated Word Recognition

- For isolated word speech recognition, we must perform the following
  - For each word v in the vocabulary, we must build an HMM
  - For each unknown word which is to be recognized, the processing of Figure must be carried out

HMM System for Isolated Word Recognition



HMM System for Isolated Word Recognition

- The probability computation step is generally performed using the Viterbi algorithm
- It requires on the order of V . N^2 .T computations
- For modest vocabulary sizes, e.g., V = IOO words, with an N = 5 state model, and T = 40 observations for the unknown word, a total of IO^5 computations are required for recognition

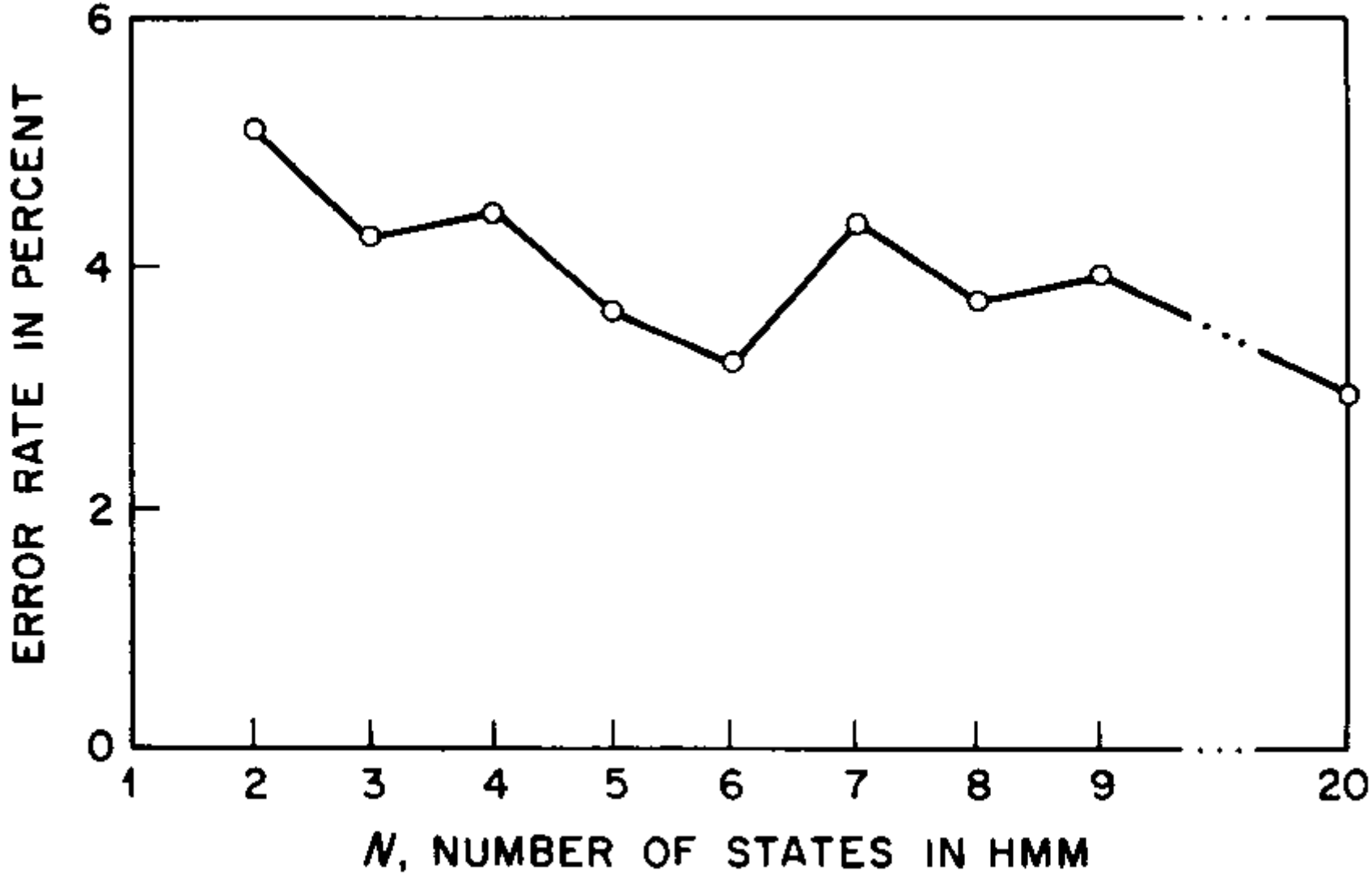
Choice of Model Parameters

- For isolated word recognition with a distinct HMM designed for each word in the vocabulary, a left-right model is more appropriate than an ergodic model
  - We can associate time with model states in a fairly straight forward manner
  - We can envision the physical meaning of the model states as distinct sounds of the word being modeled

Choice of Model Parameters

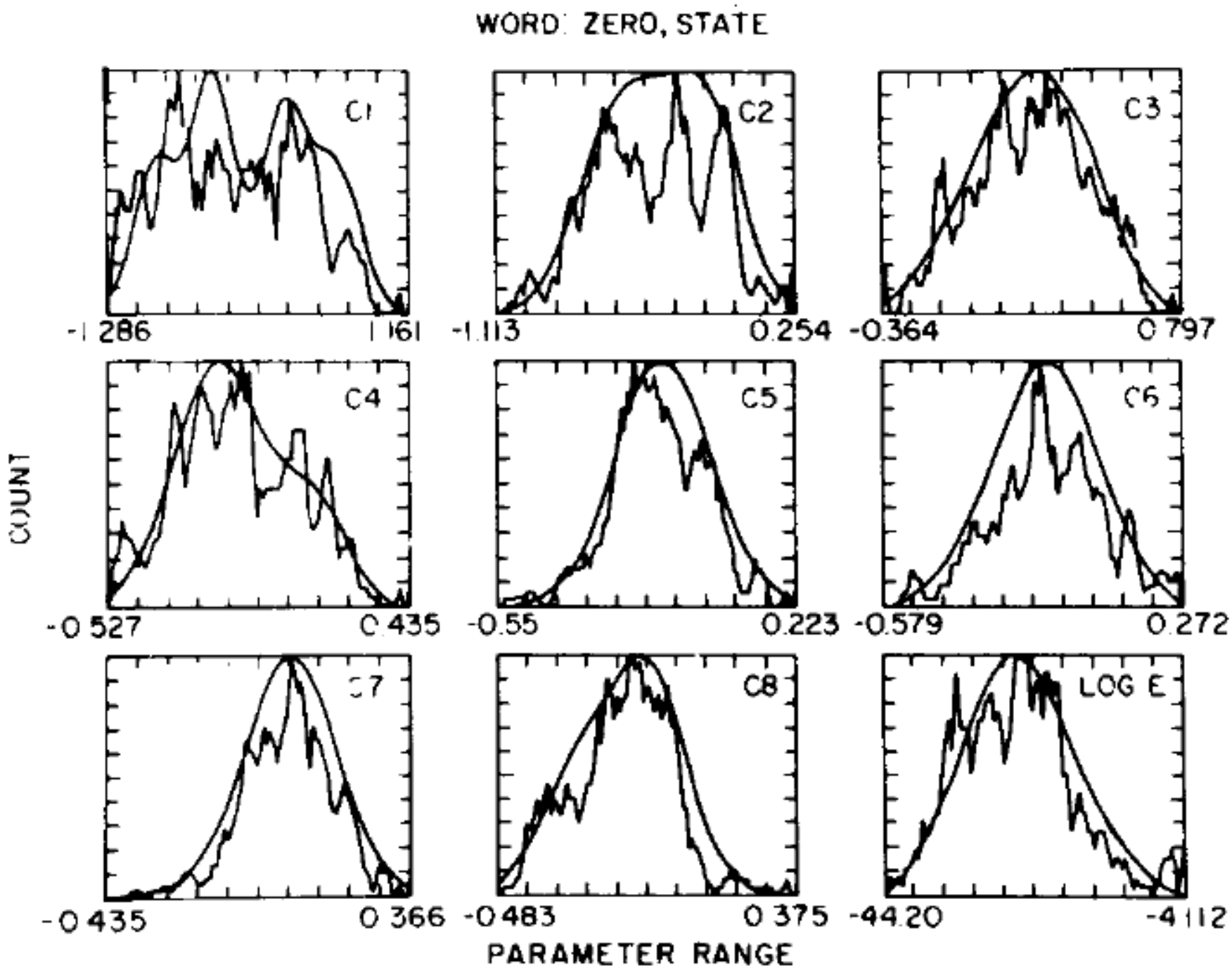
- How many states?
- Let the number of states correspond roughly to the number of sounds (phonemes) within the word hence models with number of states from 2 to 10 would be appropriate
- Let the number of states correspond roughly to the average number of observations in a spoken version of the word, the so-called Bakis model
- We discuss the first approach, with fixed number of states in each HMM

Choice of Model Parameters



Choice of Model Parameters

- Observation vector its representation?
- LPC derived weighted cepstral coefficients and weighted cepstral derivatives can be used as the observation vectors for continuous models
- For discrete symbol models we use a codebook to generate the discrete symbols
- For the continuous models we use as many as M = 64~256 mixtures per state
- It is preferable to use diagonal covariance matrices with several mixtures, rather than fewer mixtures with full covariance matrices


WORD: ZERO, STATE
C1
-1.286
1.161
C2
-1.113
0.254
C3
-0.364
0.797
C4
-0.527
0.435
C5
-0.55
0.223
C6
-0.579
0.272
C7
-0.435
0.366
C8
-0.483
0.375
LOG E
-44.20
-4.112
COUNT
PARAMETER RANGE

Choice of Model Parameters

- Another experimentally verified fact about the HMM is that it is important to limit some of the parameter estimates in order to prevent them from becoming too small

Choice of Model Parameters

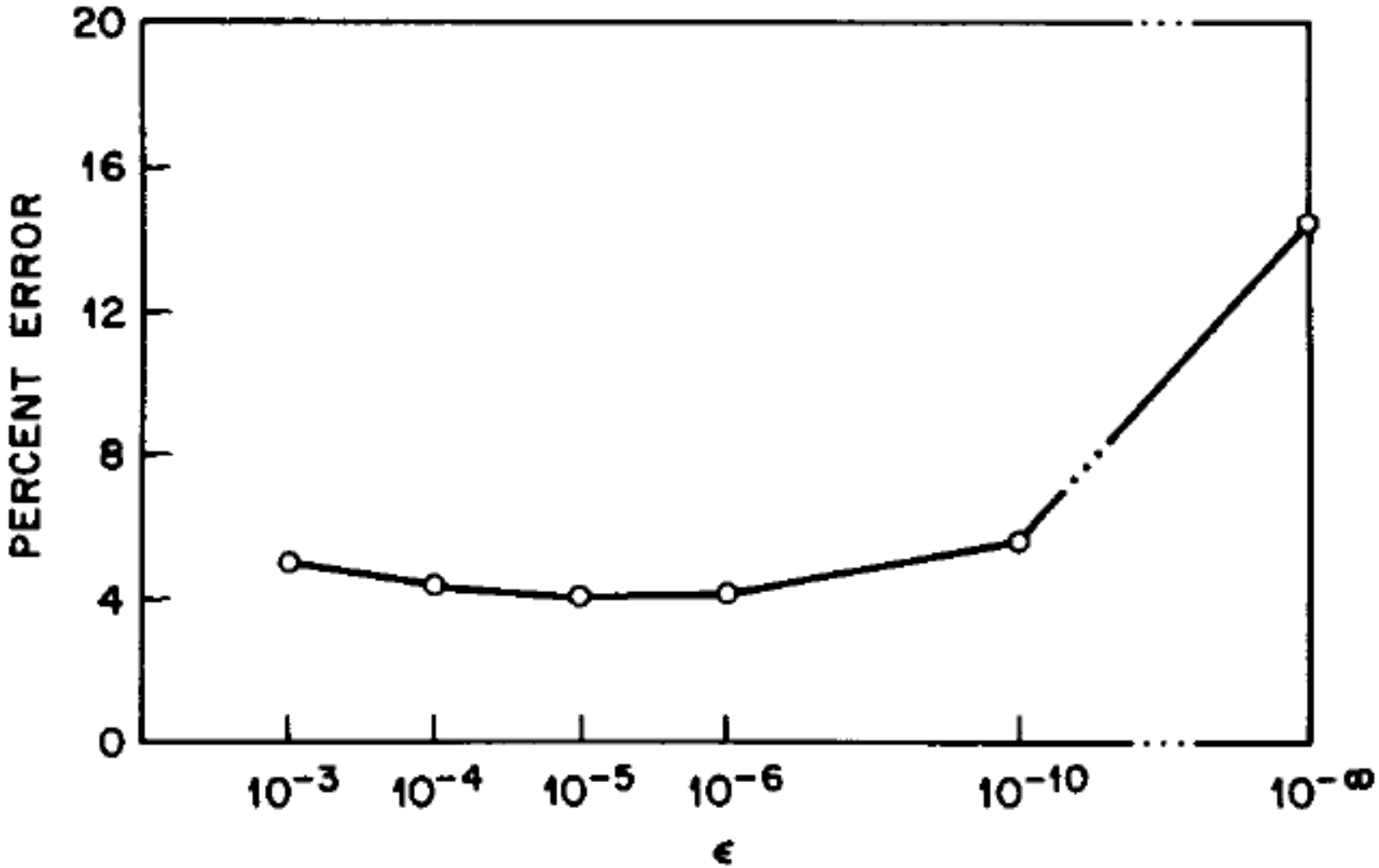
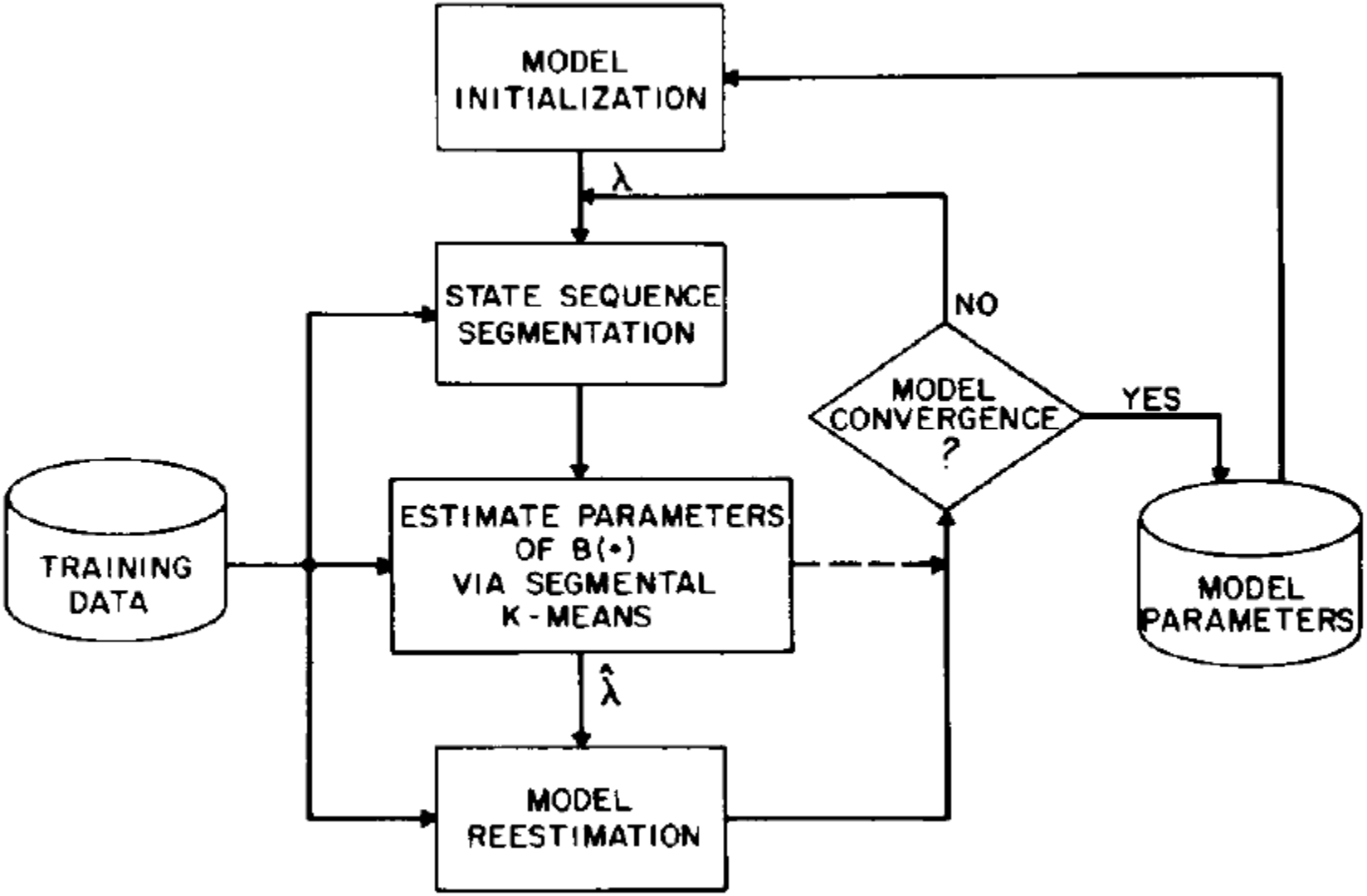
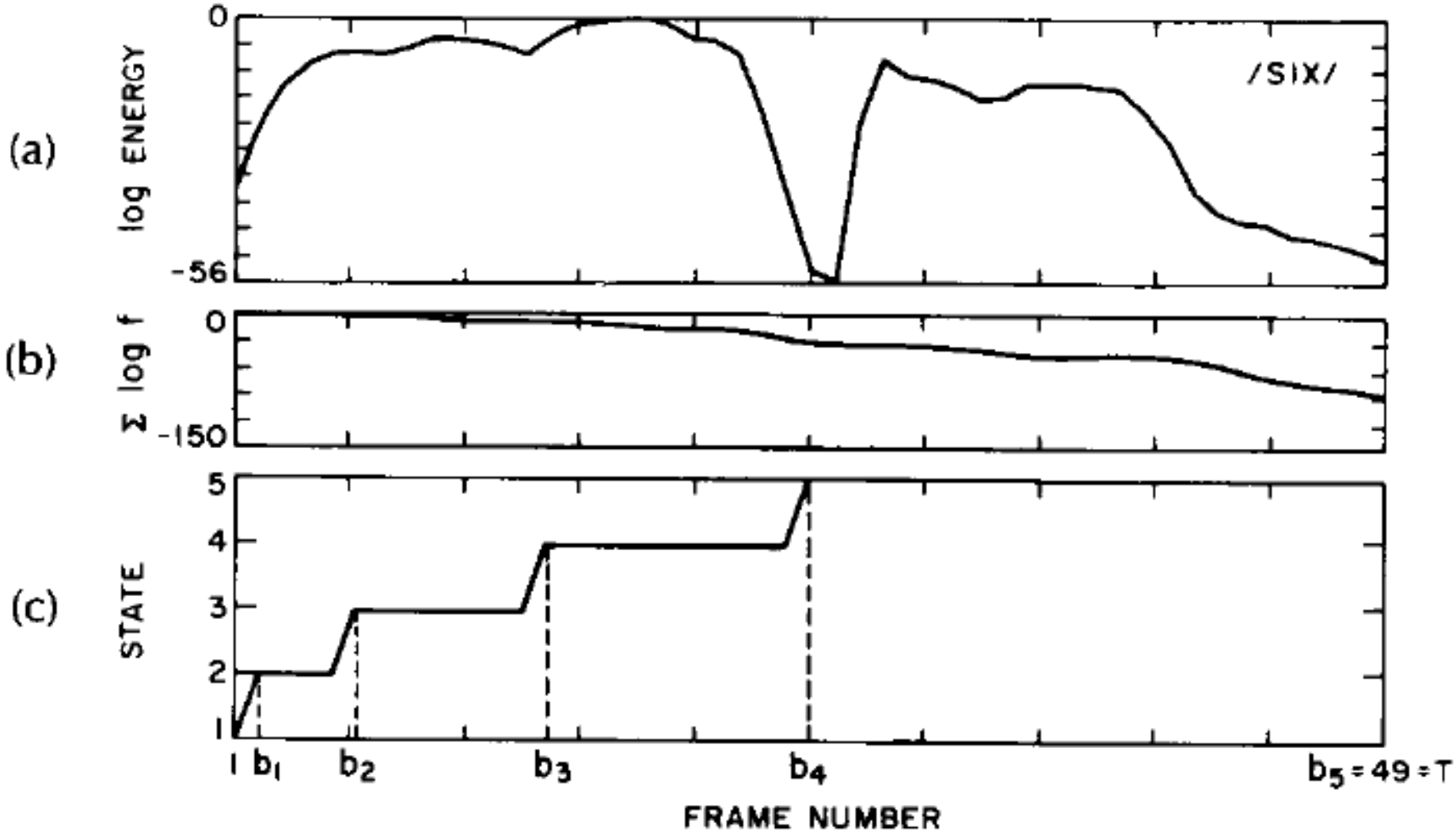


Fig. 17. Average word error rate as a function of the minimum discrete density value ϵ.

Choice of Model Parameters

- Good initial estimates of the parameters of the $b_i(O_t)$ densities were essential for rapid and proper convergence of the re-estimation formulas
- A procedure for providing good initial estimates of these parameters was devised and is shown in figure


MODEL INITIALIZATION
λ
STATE SEQUENCE SEGMENTATION
NO
MODEL CONVERGENCE ?
YES
TRAINING DATA
ESTIMATE PARAMETERS OF B(•) VIA SEGMENTAL K-MEANS
MODEL PARAMETERS
λ̂
MODEL REESTIMATION


(a)
log ENERGY
0
-56
/SIX/
(b)
Σ log f
0
-150
(c)
STATE
5
4
3
2
1
1 b1
b2
b3
b4
b5 = 49 = T
FRAME NUMBER

Incorporation of State Duration into the HMM

- The cost of including duration density was rather high; namely a D^2-fold increase in computation and a D-fold increase in storage
- An alternative procedure was proposed in which the state duration probability $p_i(d)$ was measured directly from the segmented training sequences used in the segmental K-means procedure

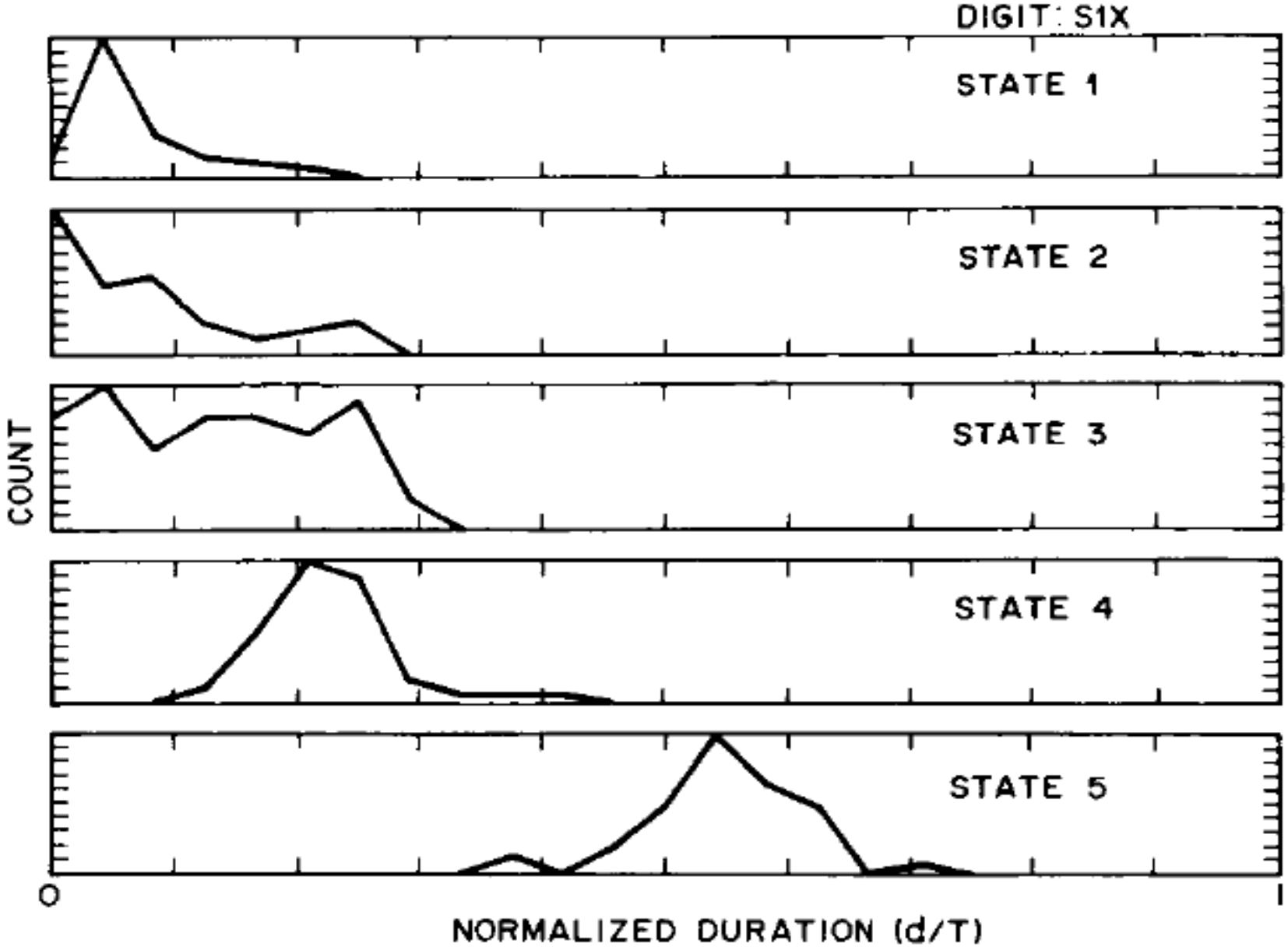


Fig. 20. Histograms of the normalized duration density for the five states of the digit "six."

Incorporation of State Duration into the HMM

- First the normal Viterbi algorithm is used to give the best segmentation of the observation sequence of the unknown word into states via a backtracking procedure
- The duration of each state is then measured from the state segmentation
- A postprocessor then increments the log-likelihood score of the Viterbi algorithm, by the quantity

$$\log \hat{P}(q, O|\lambda) = \log P(q, O|\lambda) + \alpha_d \sum_{j=1}^{N} \log [p_j(d_j)]$$

HMM Isolated-Digit Performance

Table 1 Average Digit Error Rates for Several Recognizers and Evaluation Sets

Recognizer Type	Evaluation Set			
	Original Training	TS2	TS3	TS4
LPC/DTW	0.1	0.2	2.0	1.1
LPC/DTW/VQ	—	3.5	—	—
HMM/VQ	—	3.7	—	—
HMM/CD	0	0.2	1.3	1.8
HMM/AR	0.3	1.8	3.4	4.1

Phoneme Alignment