CLEMSON UNIVERSITY

Lecture 2:

Algorithms and Applications

Outline

- State Sequence Decoding
  - Example: dice & coins
- Observation Sequence Evaluation
  - Example: spoken digit recognition
- HMM architectures
- Other applications of HMMs
- HMM tools

State Sequence Decoding

- The aim of decoding is to **discover** the hidden state sequence that most likely describes a given observation sequence.

- One solution to this problem is to use the **Viterbi** algorithm, which finds the **single best** state sequence for an observation sequence.

- **Parameter δ:** The probability of the most probable state path for the partial observation sequence:

$$\delta_t(i) = \max_{q_1, q_2, \ldots, q_{t-1}} p(q_1 q_2 \ldots q_t = s_i, o_1, o_2, \ldots, o_t | \lambda)$$

Viterbi Algorithm:

1. Initialization:

$$\delta_1(i) = \pi_i b_i(o_1),\ 1 \le i \le N$$

$$\psi_1(i) = 0$$

2. Recursion:

$$\delta_t(j) = \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}]\, b_j(o_t),\ 2 \le t \le T,\ 1 \le j \le N$$

$$\psi_t(j) = arg \max_{1 \le i \le N} [\delta_{t-1}(i) a_{ij}], 2 \le t \le T,\ 1 \le j \le N$$

3. Termination:

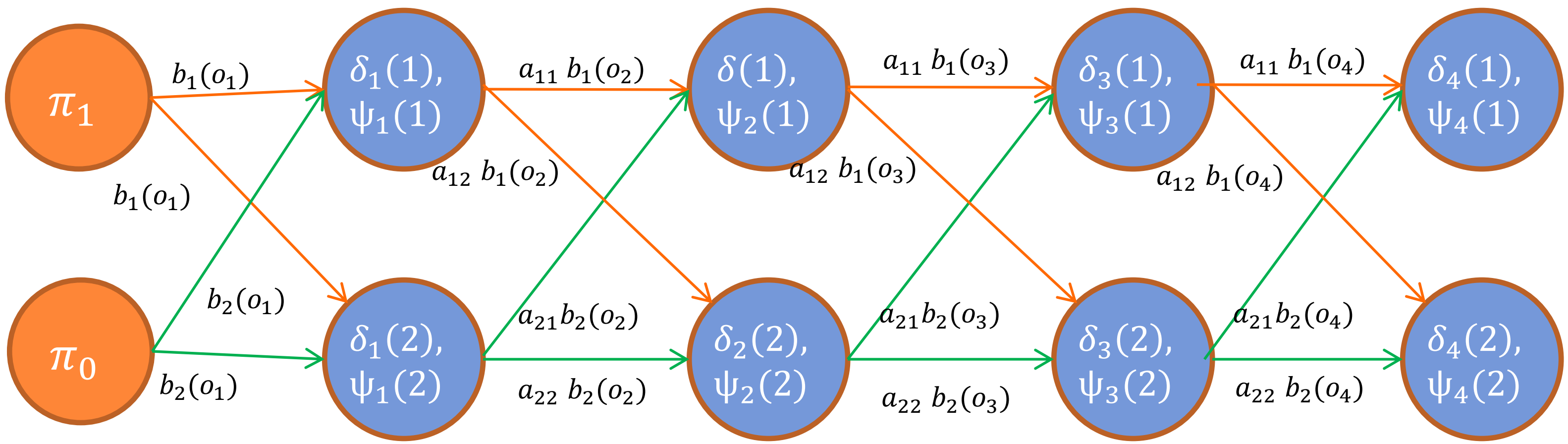
$$P^* = \max_{1 \le i \le N} [\delta_T(i)]$$

$$q_T^* = arg \max_{1 \le i \le N} [\delta_T(i)]$$

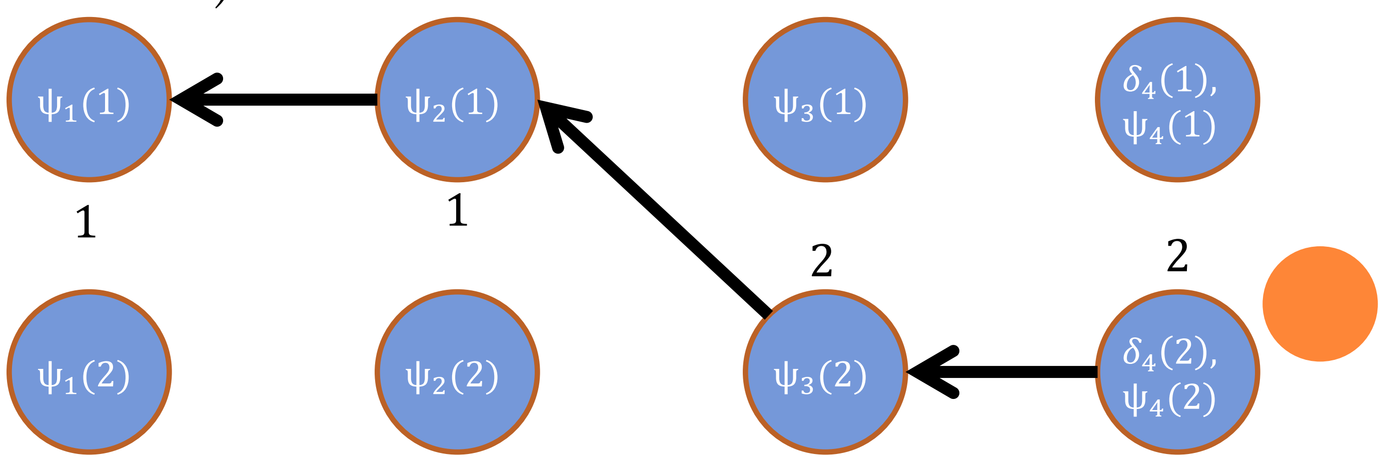
4. Optimal state sequence backtracking:

$$q_t^* = \psi_{t+1}\left(q_{t+1}^*\right),\ t = T-1, T-2, ..., 1$$

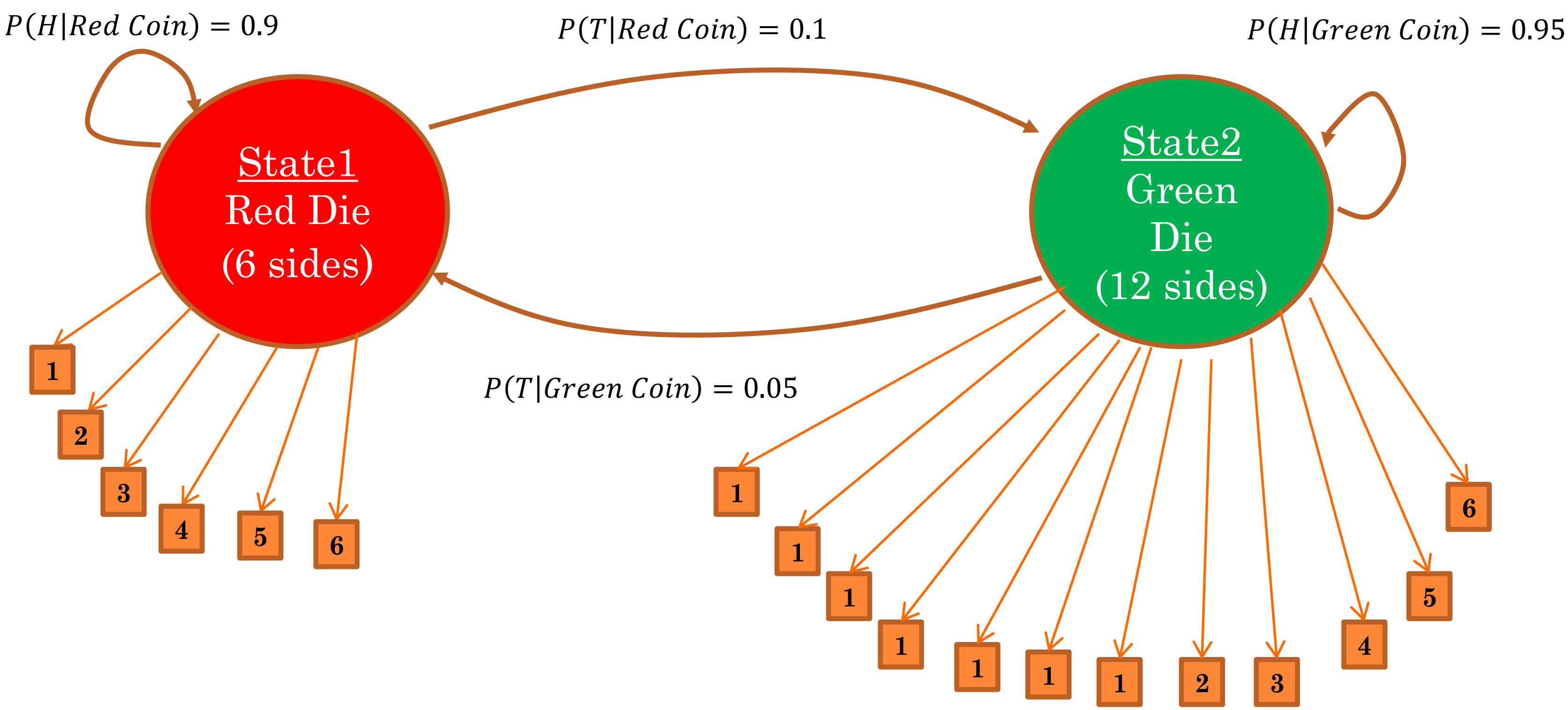
First pass:



Second pass (back track):



Dice Experiment – State Sequence Decoding

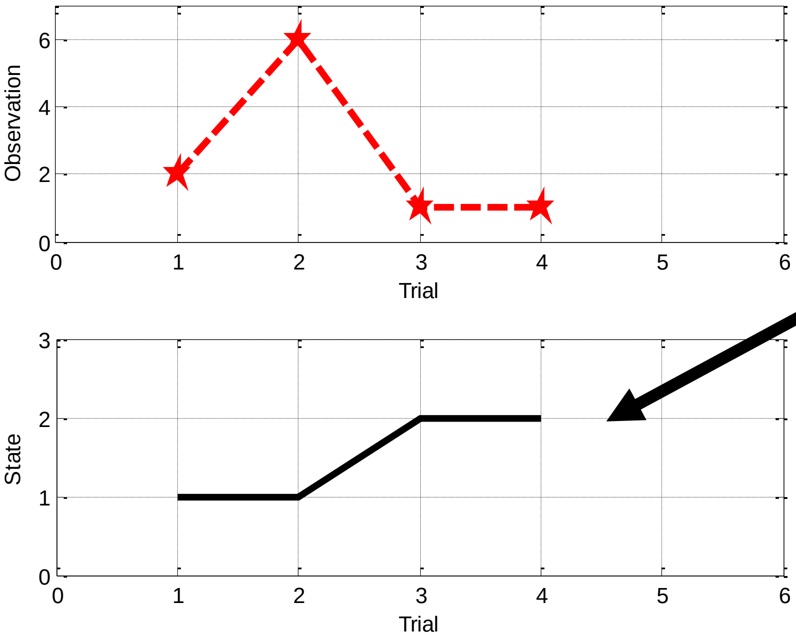


$$A = \begin{bmatrix} 0.9 & 0.1 \\ 0.05 & 0.95 \end{bmatrix} \qquad \pi = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad B = \begin{bmatrix} \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} & \frac{1}{6} \\ \frac{7}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} & \frac{1}{12} \end{bmatrix}$$

http://www.mathworks.com/help/stats/hidden-markov-models-hmm.html

- MATLAB Viterbi algorithm:

obs = [2 6 1 1]; %Die outcomes
states = [1 1 2 2]; %True state sequence




```
likelystates = hmmviterbi(obs,A,B)

likelystates =

   1    1    2    2
```

(From the dice experiment)

Observation Sequence Evaluation

- Imagine first we have L number of HMM models. This problem could be viewed as one of evaluating **how well a model predicts** a given observation sequence $O = o_1, \ldots, o_T$; and thus allows us to **choose** the most appropriate model λ_l $(1 \leq l \leq L)$ from a set, i.e.,

$$P(O|\lambda_l) = P(o_1, \ldots, o_T|\lambda_l)\ ?$$

- Remember that an observation sequence O depends on the state sequence $Q = q_1, \dots, q_T$ of a HMM λ_l. So,

$$P(O|Q, \lambda_l) = \prod_{t=1}^{T} P(o_t|q_t, \lambda_l) = b_{q_1}(o_1) \times b_{q_2}(o_2) \times \cdots \times b_{q_T}(o_T)$$

- For state sequence Q of the observation sequence O we have:

$$P(Q|\lambda_l) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T}$$

- Finally, we can come up with the final evaluation of the observation sequence as:

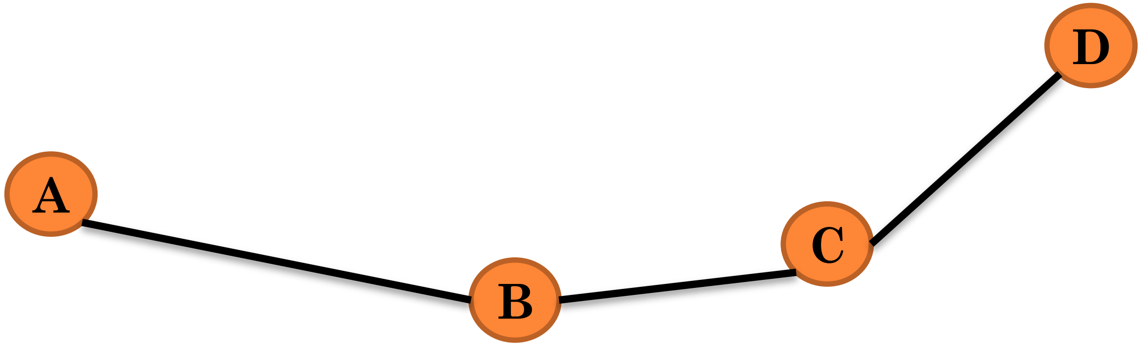
$$P(O|\lambda_l) = \sum_{Q} P(O|Q, \lambda_l) P(Q|\lambda_l)$$

$$= \sum_{q_1,\dots,q_T} \pi_{q_1} b_{q_1}(o_1)\, a_{q_1 q_2} b_{q_2}(o_2) \dots a_{q_{T-1} q_T} b_{q_T}(o_T)$$

Observation Sequence Evaluation

- There is an **issue** with the last expression: We would have to consider ALL possible state sequences for the observations evaluation (brute force).

- **Solution:** acknowledge there is redundancy in calculations → Forward – Backward Algorithm



F-B Analogy: Obtain distance from city A to other three distant cities B, C, D.

Observation Sequence Evaluation

- Forward - Backward actually are two similar algorithms which compute the same thing ($P(O|\lambda_l)$); it depends where calculations start. **Either** of the two can be use for the observation sequence evaluation.

- In the Forward case, we have a parameter α which represents the probability of the **partial** observation sequence $o_1, \dots, o_t$ and state s_i at time t, i.e.,

$$\alpha_t(i) = P(o_1, \dots, o_t, q_t = s_i | \lambda_l)$$

- Forward Algorithm:

1. Initialization:

$$\alpha_1(i) = \pi_i b_i(o_1), \; 1 \le i \le N$$

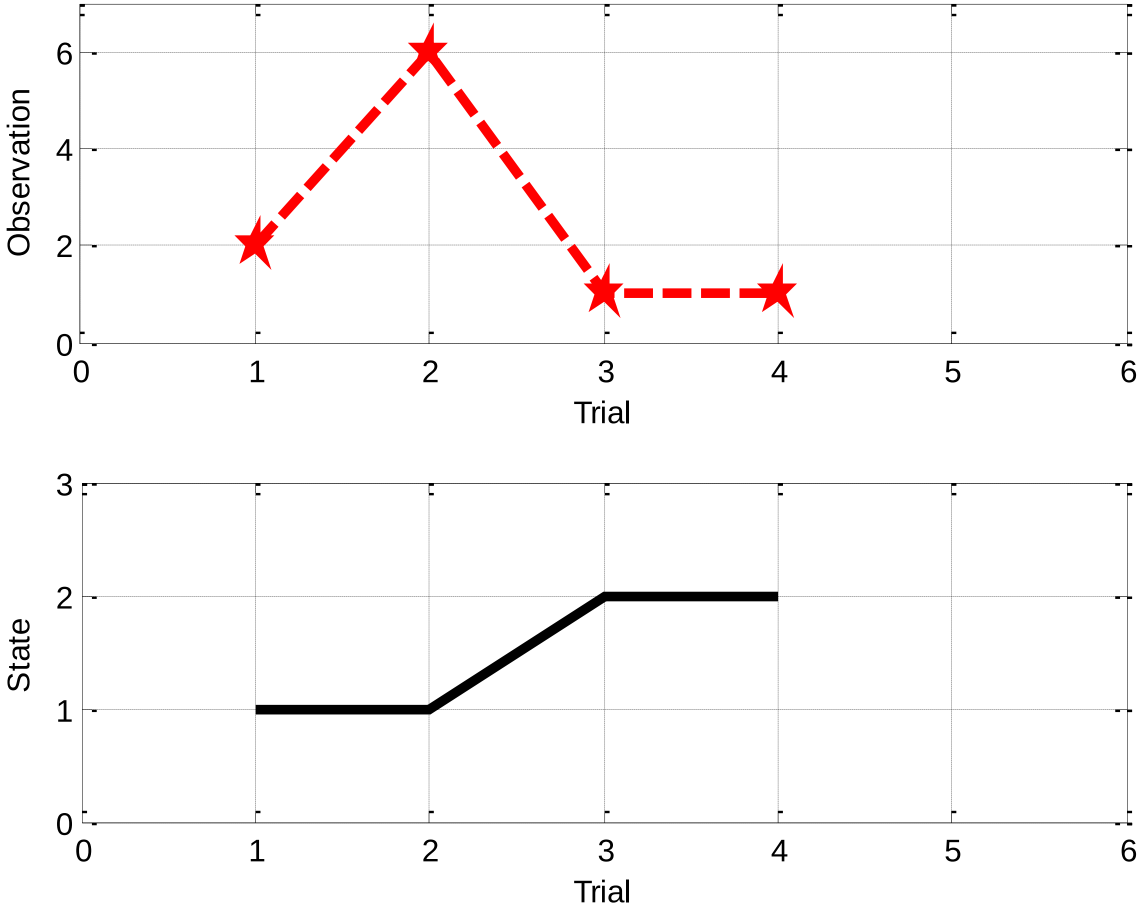
2. Recursion:

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

$$1 \le t \le T-1, \; 1 \le j \le N$$

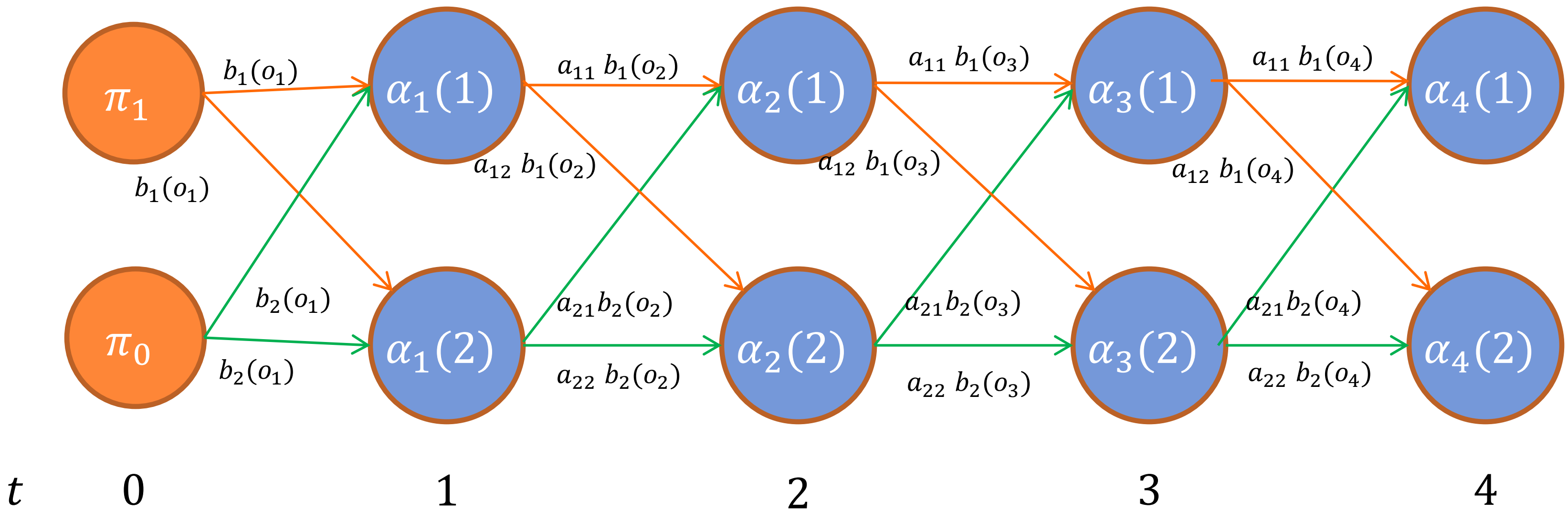
3. Termination:

$$p(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$



$O = \{2, 6,1, 1, \}$

$\mathrm{Q} = \{1, 1,2, 2\}$



$$P(o_1, o_2, o_3, o_4 \mid \lambda) = \alpha_4(1) + \alpha_4(2)$$

- Using MATLAB:

```
obs = [2 6 1 1 ];
states = [1 1 2 2];
[PSTATES, LOGPSEQ, FORWARD, BACKWARD, S]= hmmdecode(obs,A,B);

f = FORWARD.*repmat(cumprod(S),size(FORWARD,1),1);
```

f =

1.0000	0.1500	0.0226	0.0034	0.0005	0.0001
0.0000	0.0083	0.0019	0.0024	0.0015	0.0009

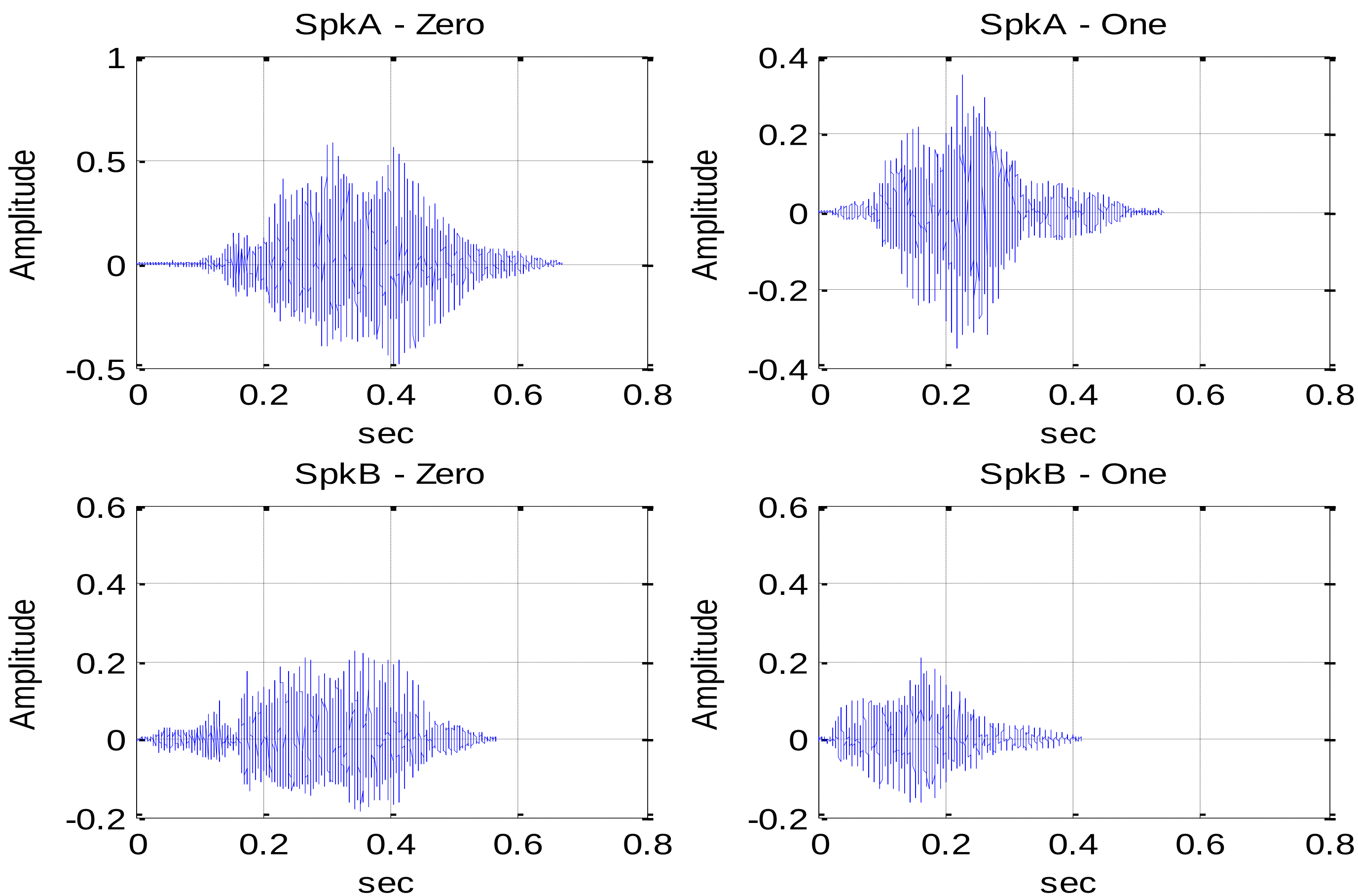
$P(o_1, o_2, o_3\,, o_4\,|\lambda) = 0.0001 + 0.0009 = 0.001$

$\log[P(o_1, o_2, o_3\,, o_4\,|\lambda)] = \log(0.0001 + 0.0009) = -3$

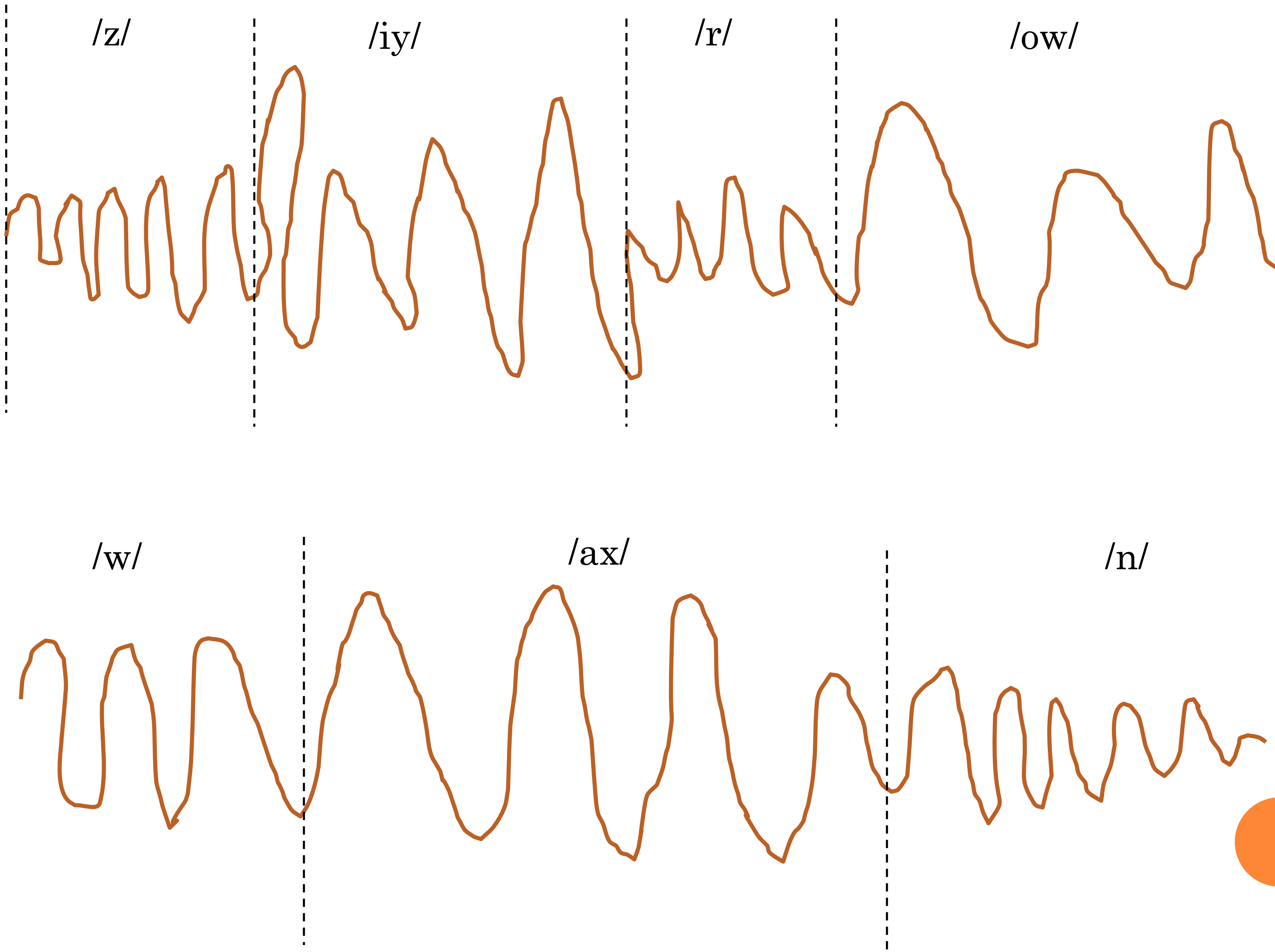
This number would be significant if we can compare it with different HMMs.

Digit Recognition Example – Observation Sequence Evaluation

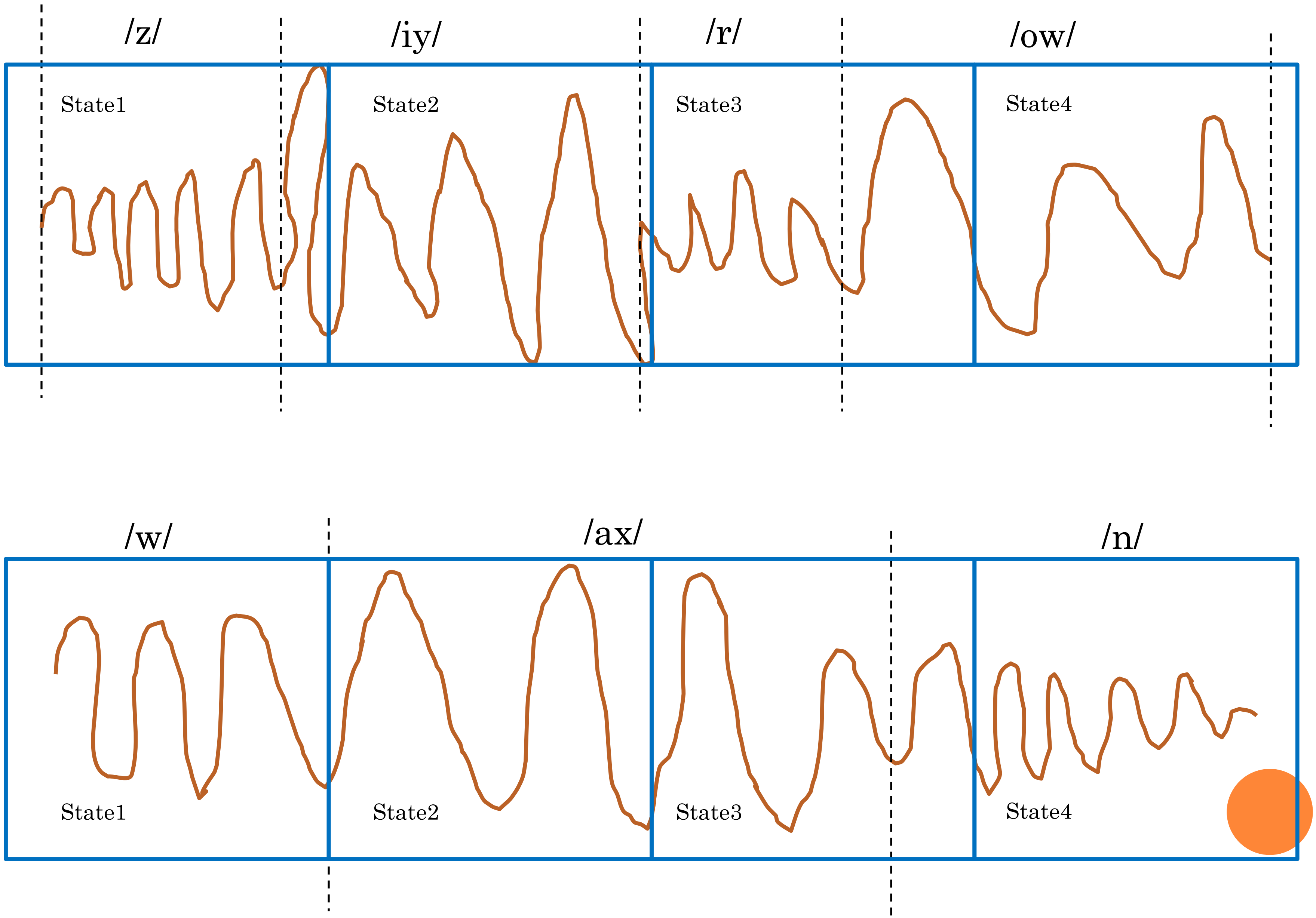
- Recognize digits ‘Zero’ and ‘One’ from two speakers:



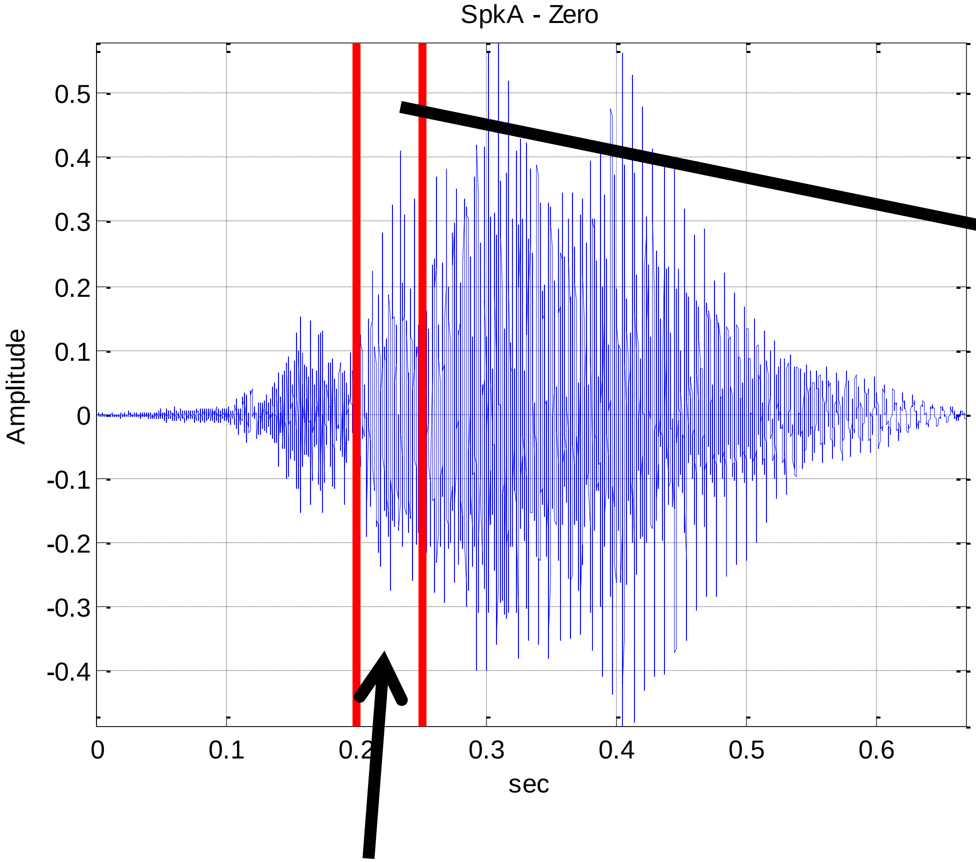
Phonemes:



States: Abstract representation of the sounds



- Feature extraction of speech signals:



$$Feature = \begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \ldots \\ C_{12} \\ \Delta \\ \ldots \\ \Delta\Delta \\ \ldots \end{bmatrix}$$

39-element feature vector per frame

Divide the speech signal into frames using a 30ms window.

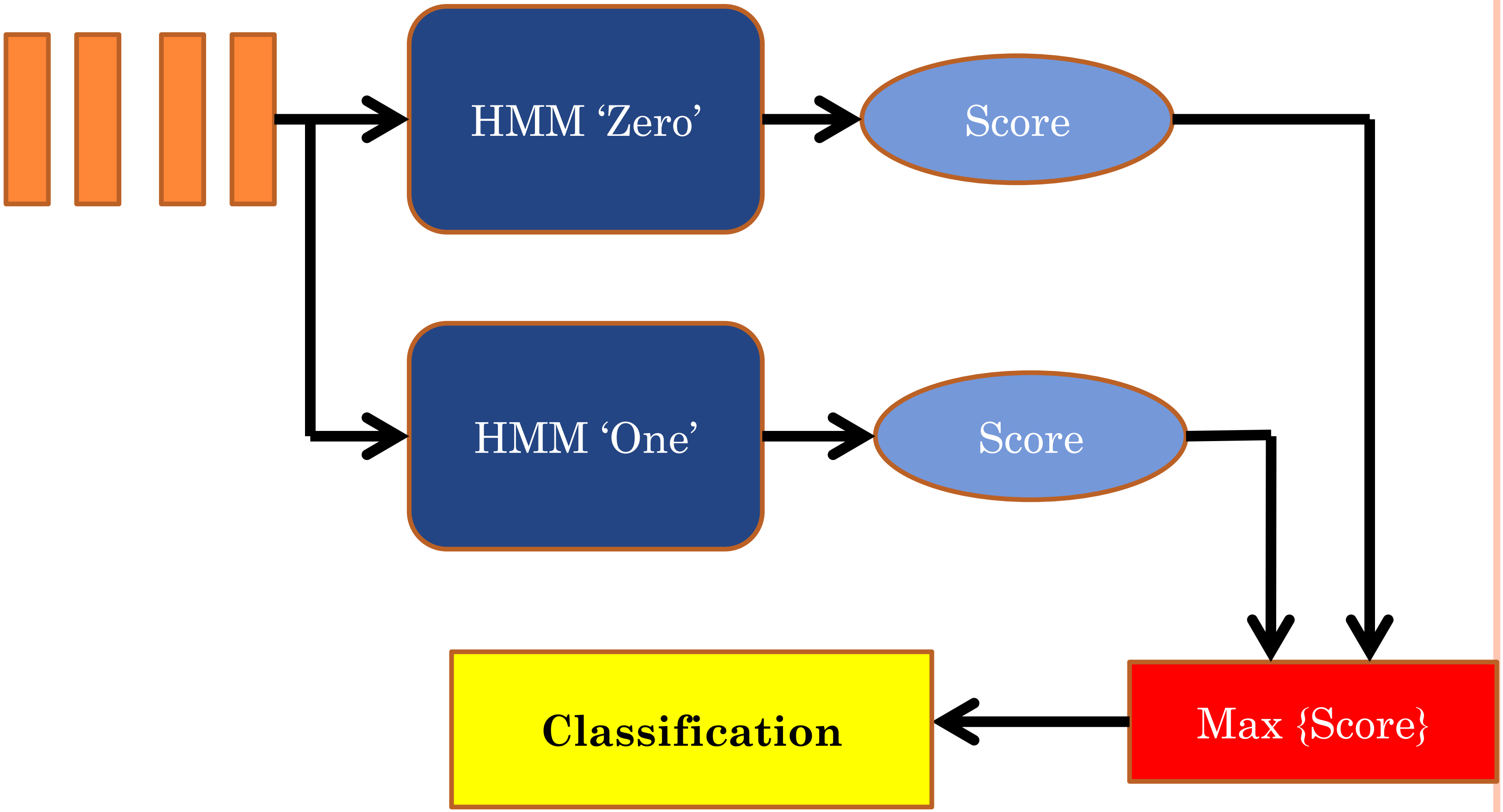
Observation Sequence Evaluation

- For a word (zero/one) it has several feature vectors:

$$\underbrace{\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \ldots \\ C_{12} \\ \Delta \\ \ldots \\ \Delta\Delta \\ \ldots \end{bmatrix} \begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \ldots \\ C_{12} \\ \Delta \\ \ldots \\ \Delta\Delta \\ \ldots \end{bmatrix} \quad \ldots \quad \begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \ldots \\ C_{12} \\ \Delta \\ \ldots \\ \Delta\Delta \\ \ldots \end{bmatrix}}_{}$$

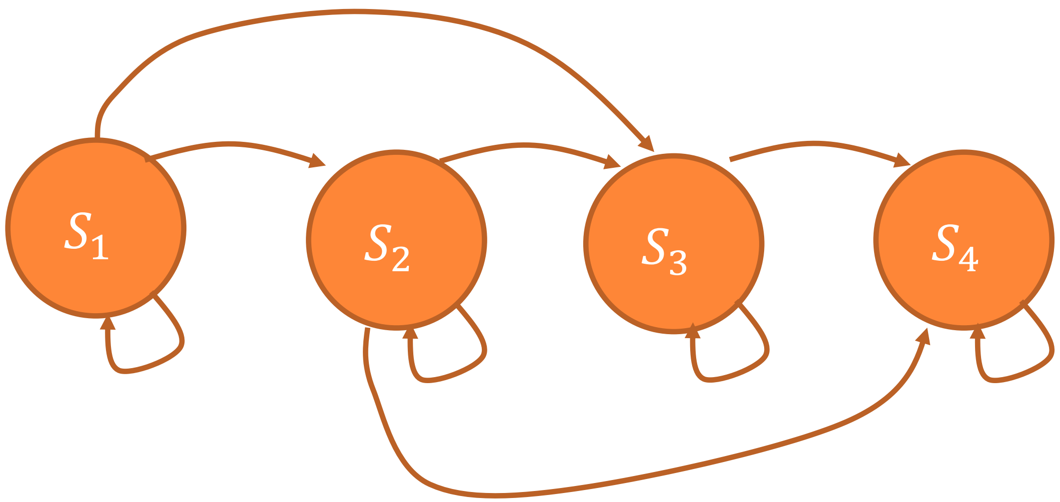
For a HMM these are the observations!

- Basic idea:



- HMM architecture:

Left-Right Architecture



- Create a HMM for word Zero and One.
- Both HMM have the same number of states (4).
- Model the emission probabilities with 2 Gaussian Mixtures.
- States will be an abstract representation of the features.
- 3 utterances of each word (both speakers) will be used as training data.
- 2 utterances of each word (both speakers) will be used as test data.

- Build the training data for each word by concatenation:
  - zero_data:

$$
\boxed{\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix} \cdots \begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}}
\boxed{\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix} \cdots \begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}}
\boxed{\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix} \cdots \begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}}
$$

  - one_data:

$$
\boxed{\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix} \cdots \begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}}
\boxed{\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix} \cdots \begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}}
\boxed{\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}\begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix} \cdots \begin{bmatrix} Log\ Energy \\ C_0 \\ C_1 \\ \cdots \\ C_{12} \\ \Delta \\ \cdots \\ \Delta\Delta \\ \cdots \end{bmatrix}}
$$

- We start by estimating the transition matrix for both HMMs (HMM Toolbox, Kevin Murphy, 1998)

```
M = 2; %mixtures
Q = 4; %states
O = size(one_data,2); %dimension
T = size(one_data,1);
nex = 1;
data = zeros(O,T,nex);
data(:,:,nex) = one_data';

prior0 = normalise(rand(Q,1));
transmat0 = mk_stochastic(rand(Q,Q));

[mu0, Sigma0] = mixgauss_init(Q*M,reshape(data, [O T*nex]) , 'diag');
mu0 = reshape(mu0, [O Q M]);
Sigma0 = reshape(Sigma0, [O O Q M]);
mixmat0 = mk_stochastic(rand(Q,M));

[LL, prior1, transmat1, mu1, Sigma1, mixmat1] = mhmm_em(data, prior0, transmat0, mu0, Sigma0, mixmat0, 'max_iter',
20);

oneA = transmat1;
omu = mu1;
oSigma = Sigma1;
oprior = prior1;
omixmat = mixmat1;
```

Parameters

- Transition probability matrices:

zA =

0.9414	0.0293	0.0293	0
0	0.7966	0.1017	0.1017
0	0	0.9016	0.0984
0	0	0	1.0000

oA =

0.8388	0.0806	0.0806	0
0	0.9212	0.0394	0.0394
0	0	0.9076	0.0924
0	0	0	1.0000

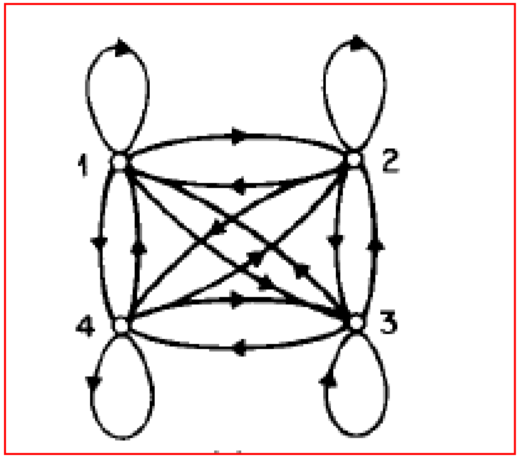
- Now, we can evaluate the test data by feeding to each of the HMM models, compute the log likelihood score, and assigned to a HMM based on the max of score.

```
LogLikScore = zeros(4,2);

LogLikScore(1,1) = mhmm_logprob(ts_zeromfcc1', zprior, zA, zmu, zSigma, zmixmat);
LogLikScore(1,2) = mhmm_logprob(ts_zeromfcc1', oprior, oA, omu, oSigma, omixmat);

LogLikScore(2,1) = mhmm_logprob(ts_zeromfcc2', zprior, zA, zmu, zSigma, zmixmat);
LogLikScore(2,2) = mhmm_logprob(ts_zeromfcc2', oprior, oA, omu, oSigma, omixmat);

LogLikScore(3,1) = mhmm_logprob(ts_onemfcc1', zprior, zA, zmu, zSigma, zmixmat);
LogLikScore(3,2) = mhmm_logprob(ts_onemfcc1', oprior, oA, omu, oSigma, omixmat);

LogLikScore(4,1) = mhmm_logprob(ts_onemfcc2', zprior, zA, zmu, zSigma, zmixmat);
LogLikScore(4,2) = mhmm_logprob(ts_onemfcc2', oprior, oA, omu, oSigma, omixmat);

Results = { };
for i=1:size(LogLikScore,1)
   if(LogLikScore(i,1)>LogLikScore(i,2))
      Results = [Results;{'Zero'}];
   else
      Results = [Results;{'One'}];
   end
end
```

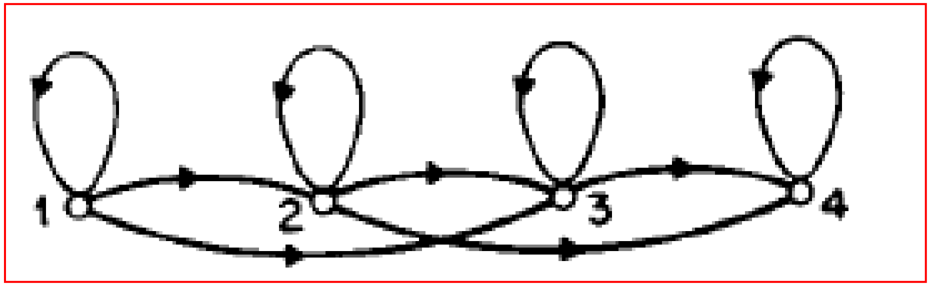
```
LogLikScore =

   1.0e+03 *

   -2.4663      -Inf
   -2.2151      -Inf
   -3.1509    0.6508
   -2.4131    0.4971
```

```
Results =

    'Zero'
    'Zero'
    'One'
    'One'
```

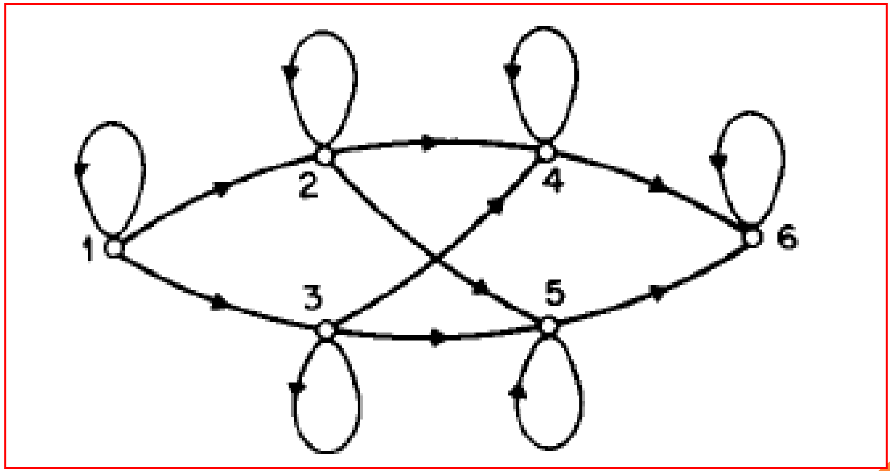
HMM Architectures



Ergodic HMM



Left-Right HMM

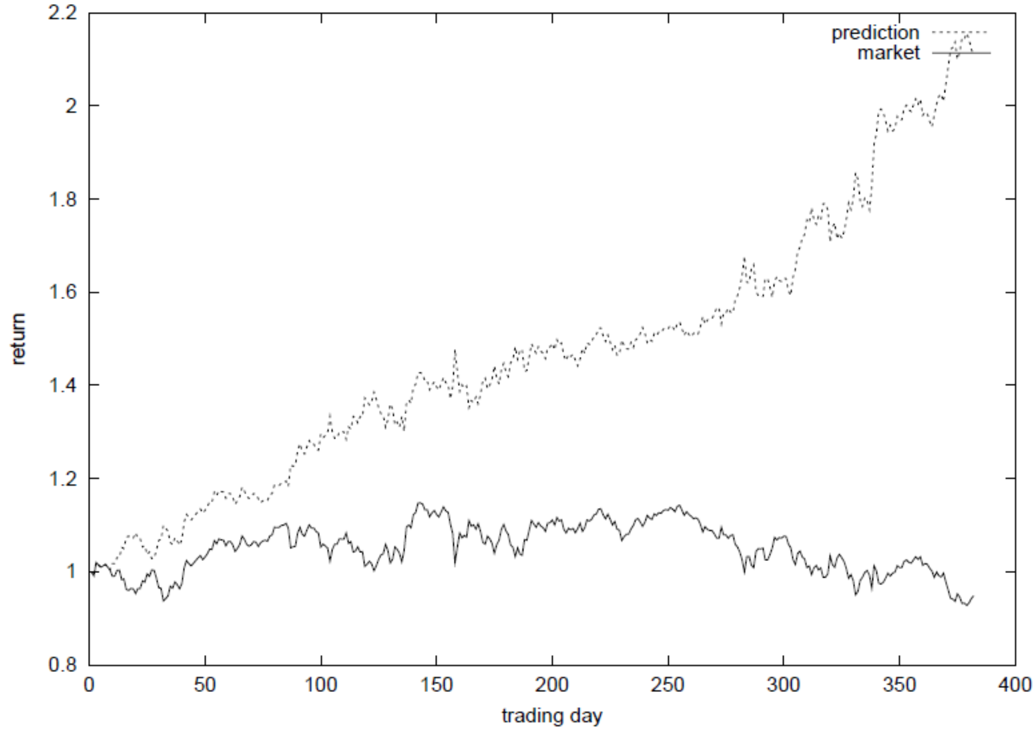


Parallel HMM

Other Applications of HMMs

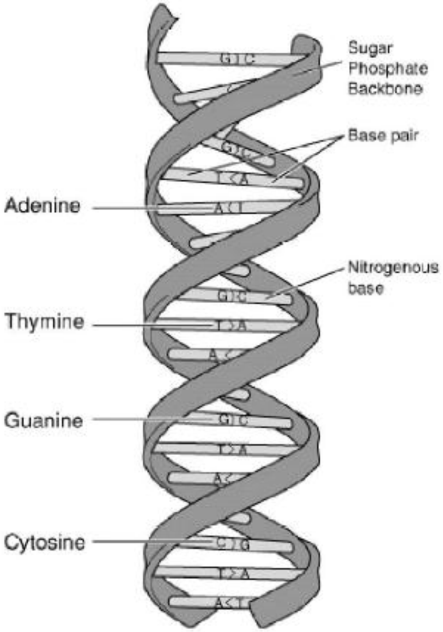
- Due to the powerfulness that Markov models provide, it can be used anywhere sequential information exists:
  - Finance
  - Biology
  - Tracking systems
  - Speech processing
  - Image processing
  - Communication systems
  - Many more…

- Model non-stationary and non-linearity of financial data to predict the direction of the time series.



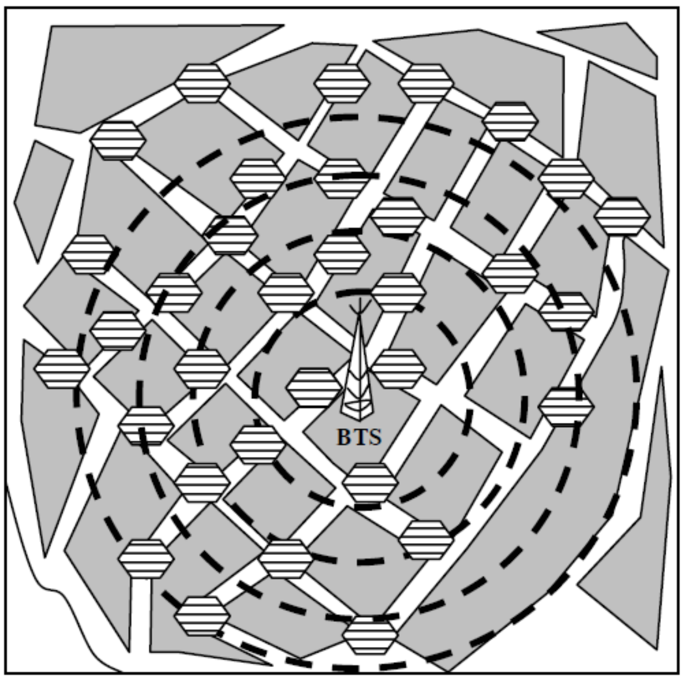
Zhang, Y.; Prediction of Financial Time Series with Hidden Markov Models; Shandong University, China, 2001.

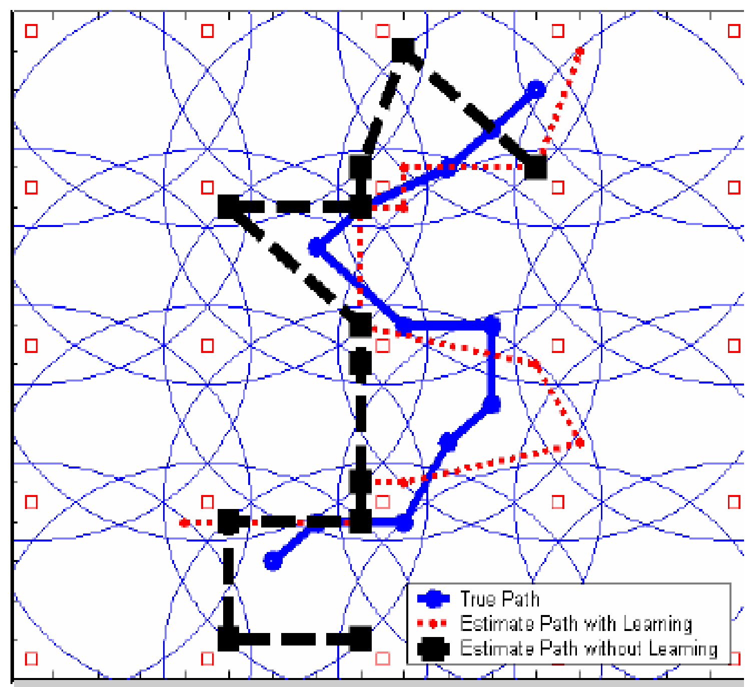
- DNA is composed if 4 bases (A, G, T, C) which pair together form nucleotides. Markov models can compute likelihoods of an DNA sequence.



Nelson, R.; Foo, S.; Weatherspoon, M.; , "Using Hidden Markov Modeling in DNA Sequencing," *System Theory, 2008. SSST 2008. 40th Southeastern Symposium on* , vol., no., pp.215-217, 16-18 March 2008

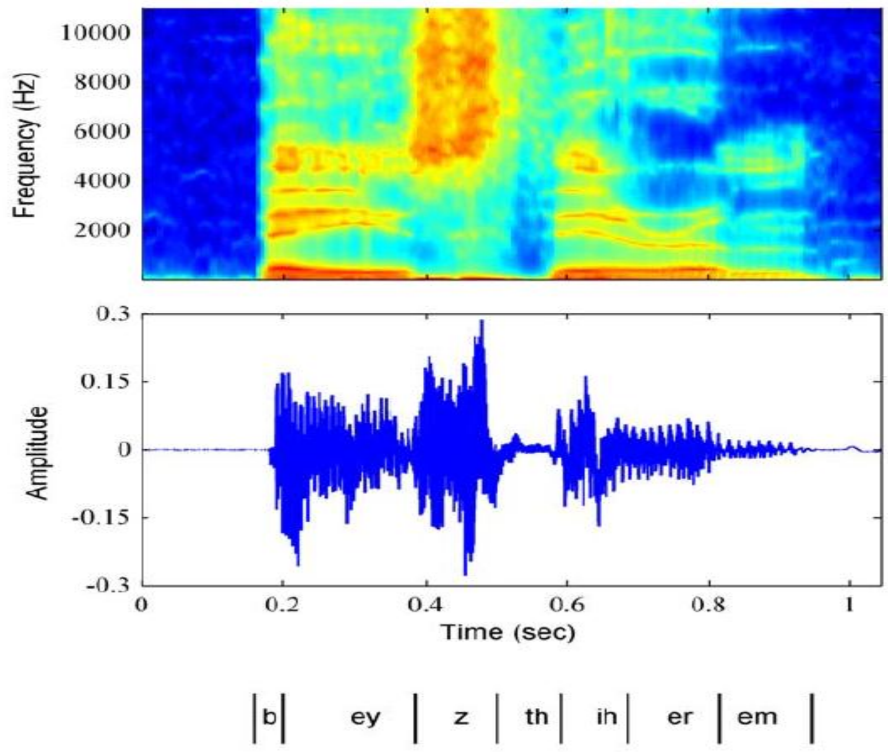
- Markov models can be used to estimate the position in a tracking system.





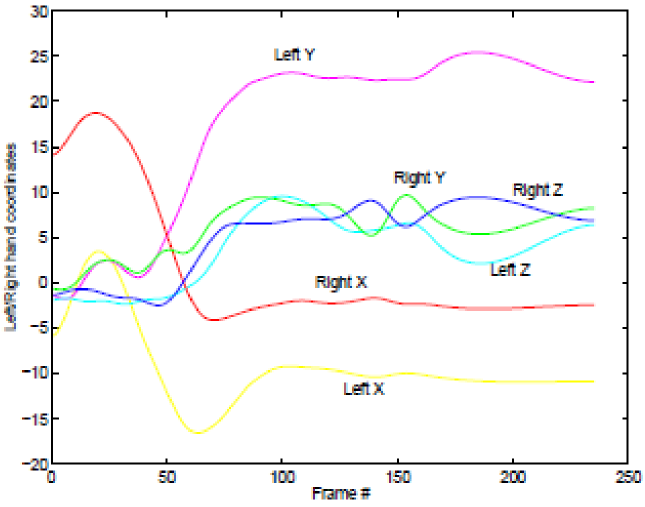
Hieu and Thanh, Ground Mobile Target Tracking by Hidden Markov Models

- Speech recognition has been the most exploited area for use of Markov models.



Christopher M. Bishop. **"Pattern recognition and machine learning. Springer"**, 1st ed. 2006. corr. 2nd printing edition, October 2006.

- Human action recognition can be modeled with Markov models



cobra

Brand, M.; Oliver, N.; Pentland, A.; , "Coupled hidden Markov models for complex action recognition," *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on* , vol., no., pp.994-999, 17-19 Jun 1997

HMM Tools

- Hidden Markov Toolkit (HTK) - Cambridge University :
  - http://htk.eng.cam.ac.uk/

- MATLAB functions:
  - *hmmtrain, hmmgenerate, hmmdecode, hmmestimate, hmmviterbi*

- HMM Matlab Toolbox (Kevin Murphy, 1998):
  - http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm.html

- MATLAB functions for training and evaluating HMMs and GMMs (Ron Weiss, Columbia University):
  - https://github.com/ronw/matlab_hmm

- Sage (open-source mathematics software): http://www.sagemath.org/
  - HMM: statistics package
  - Online Sage Notebook (Gmail account)
  - http://www.sagemath.org/doc/reference/index.html
  - http://www.sagemath.org/doc/reference/sage/stats/hmm/hmm.html

References

- Rabiner, L.R.; , **"A tutorial on hidden Markov models and selected applications in speech recognition,"** *Proceedings of the IEEE* , vol.77, no.2, pp.257-286, Feb 1989
- John R. Deller, John, and John H. L. Hansen. **"Discrete-Time Processing of Speech Signals".** Prentice Hall, New Jersey, 1987.
- Barbara Resch (modified Erhard and Car Line Rank and Mathew Magimai-doss); **"Hidden Markov Models A Tutorial for the Course Computational Intelligence."**
- Henry Stark and John W. Woods. **"Probability and Random Processes with Applications to Signal Processing (3rd Edition)."** Prentice Hall, 3 edition, August 2001.
- HTKBook: http://htk.eng.cam.ac.uk/docs/docs.shtml