

## ML for Sequential Data (HMM)

Prepared by:

*Smitkumar Khanpara (2019201021)*

*Anurag Pateriya (2019201057)*

### Contents

<b>1</b>	<b>Sequential Data</b>	<b>2</b>
<b>2</b>	<b>Forecasting (AutoRegressive models)</b>	<b>3</b>
<b>3</b>	<b>Basics of Markov Model</b>	<b>5</b>
<b>4</b>	<b>Hidden Markov model</b>	<b>6</b>
<b>5</b>	<b>HMM Parameters</b>	<b>8</b>
<b>6</b>	<b>Problems of HMM</b>	<b>10</b>
<b>7</b>	<b>References</b>	<b>13</b>

# 1 Sequential Data

- Data/Output Paradigms:

- In machine learning, the data type of individual variable is a very important part. Based on the data type of variables we use different machine learning techniques, one can choose supervised learning techniques and if the dependent variable is absent than based on the data types of independent variables one can choose unsupervised learning techniques.

Table 1: Data/output Paradigm

Output Data	Type of Data	Task	Machine Learning Algorithm
0.15, 0.2, 0.35, ...	Numerical	Prediction	Regression
A, D, E, F	Categorical	Classification	Support Vector Machine
The quick brown fox jumps over the lazy dog.	??	??	??

- From the table-1 we can see that data given in the first two rows are self-explanatory of task need to be performed. What if the output is the sequence as given in the third row? We will see machine learning techniques for sequential data in the next topics.

- Sequential Data: Sequential data is a kind of data in which elements appear in a certain order, and they are not independent of each other. So we can assume that time series is a kind of sequential data, because of the order matters. A time series is a sequence taken at successive equally spaced points in time and it is not the only case of sequential data. Other examples are given below.

1. Audio data: The audio waveform is the one kind of sequential data. The image represents amplitude vs. time graph for the audio sequence.
2. Video: It is a sequence of frames per second.
3. Text: It is a sequence of words in the well-defined format as a sentence.
4. Biological: DNA is a sequence of adenine (A), guanine (G), cytosine (C), and thymine (T).

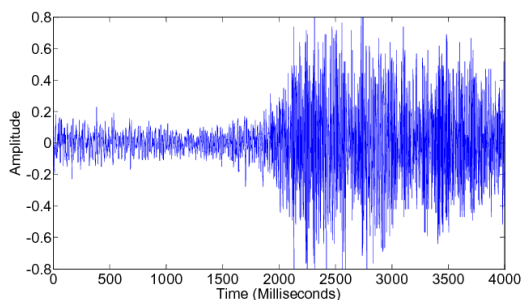


Figure 1: Audio data



Figure 2: Video Data

The quick brown fox jumps right over the lazy dog. the quick brown fox jumps right over the lazy dog.

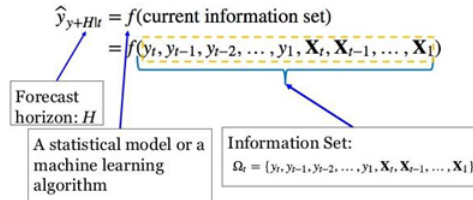
Figure 3: Text Data



Figure 4: DNA Sequence

## 2 Forecasting (AutoRegressive models)

- AutoRegressive model is a time series model that uses observations from previous time steps as input to a regression equation to predict the value at the next time step. It is a very simple idea that can result in accurate forecasts on a range of time series problems.



- This above example is an example of self supervised learning, so we are using the data itself to estimate the model. There is no separate target output label.

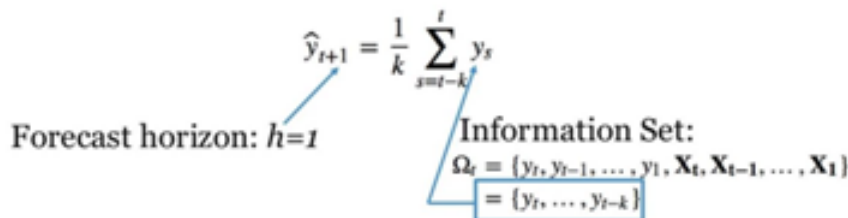
### 1. Forecasting: Simplest model

This is way simplest model in which to predict value for a particular timestamp then the best predictor is the most recent value that you have. This is also known as AR (1) process which is autoregressive with time stamp 1.

$$\hat{y}_{t+1} = y_t$$

### 2. Forecasting: AR(k) model

This is a more sophisticated model than AR(1). In AR(k) model we aggregate most recent k values by taking a mean average. So, the statistical machine learning model we are using here is quite simple.



### 3. Forecasting: ARMA (Auto Regressive Moving Average)

- In the statistical analysis of time series, ARMA models provide a parsimonious description of a stationary stochastic process in terms of two polynomials, one for the autoregression (AR) and the second for the moving average (MA).
- The AR part involves regressing the variable on its own lagged (i.e., past) values.
- The MA part involves modeling the error term as a linear combination of error terms occurring contemporaneously and at various times in the past.
- The model is usually referred to as the ARMA(p,q) model where p is the order of the AR part and q is the order of the MA part (as defined below).
- Every stationary ARMA model specifies  $Y_t$  as a weighted sum of past error terms.
- AR(p) :

$$Y_t = \delta + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \phi_3 Y_{t-3} + \dots + \phi_p Y_{t-p} + a$$

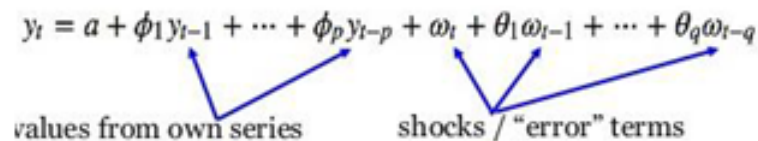
- MA(q) :

$$Y_t = \mu + \omega_t + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + \dots + \theta_q \omega_{t-q}$$

- ARMA(p,q) :

$$Y_t = a + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \phi_3 Y_{t-3} + \dots + \phi_p Y_{t-p} + \omega_t + \theta_1 \omega_{t-1} + \theta_2 \omega_{t-2} + \dots + \theta_q \omega_{t-q}$$

$$y_t = a + \phi_1 y_{t-1} + \dots + \phi_p y_{t-p} + \omega_t + \theta_1 \omega_{t-1} + \dots + \theta_q \omega_{t-q}$$



values from own series                  shocks / "error" terms

$$\omega_t \sim N(0, \sigma_\omega^2) \quad \forall t$$

- Fitting Model:

- Choosing p and q

Finding appropriate values of p and q in the ARMA(p,q) model can be facilitated by plotting the partial auto-correlation functions for an estimate of p, and likewise using the auto-correlation functions for an estimate of q. Further information can be gleaned by considering the same functions for the residuals of a model fitted with an initial selection of p and q.

- Estimating coefficients

ARMA models in general can be, after choosing p and q, fitted by least squares regression to find the values of the parameters which minimize the error term. It is generally considered good practice to find the smallest values of p and q which provide an acceptable fit to the data. For a pure AR model the Yule-Walker equations may be used to provide a fit.

### 3 Basics of Markov Model

- Markov Processes:

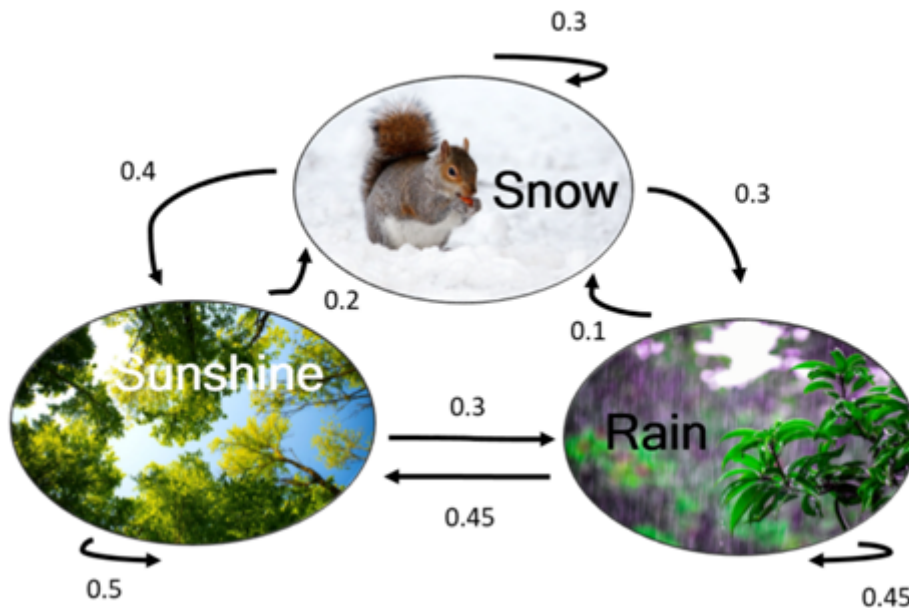
A Markov process is a stochastic process that satisfies the Markov property (sometimes characterized as "memorylessness"). In simpler terms, it is a process for which predictions can be made regarding future outcomes based solely on its present state and—most importantly—such predictions are just as good as the ones that could be made knowing the process's full history. In other words, conditional on the present state of the system, its future and past states are independent.

For a markov process, the basic assumption is that next state depends only on previous state:

$$\mathbb{P}(X_t = j | X_1 = i_1, \dots, X_{t-1} = i_{t-1}) = \mathbb{P}(X_t = j | X_{t-1} = i_{t-1})$$

- Markov chains:

- A Markov chain is a model that tells us something about the probabilities of sequences of random variables, states, each of which can take on values from some set. These sets can be words, or tags, or symbols representing anything, like the weather.
- A Markov chain makes a very strong assumption that if we want to predict the future in the sequence, all that matters is the current state. The states before the current state have no impact on the future except via the current state.



- Formally, a Markov chain is specified by the following components:

$Q = q_1 q_2 \dots q_N$	a set of $N$ <b>states</b>
$A = a_{11} a_{12} \dots a_{n1} \dots a_{nn}$	a <b>transition probability matrix</b> $A$ , each $a_{ij}$ representing the probability of moving from state $i$ to state $j$ , s.t. $\sum_{j=1}^n a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, \dots, \pi_N$	an <b>initial probability distribution</b> over states. $\pi_i$ is the probability that the Markov chain will start in state $i$ . Some states $j$ may have $\pi_j = 0$ , meaning that they cannot be initial states. Also, $\sum_{i=1}^n \pi_i = 1$

## 4 Hidden Markov model

- Hidden Markov Model (HMM) is a parameterized distribution for sequences of observations. An intuitive way to explain HMM is to go through an example. Suppose that Taylor hears (a.k.a. observes) a sequence of  $T$  sounds  $O_1, O_2, \dots, O_T$  and he wants to reason something about this sequence. He makes the assumption that the sequence of sounds that he heard depends on a sequence of  $T$  words  $S_1, S_2, \dots, S_T$ , which he never gets to see and which is why they are called the hidden states. HMM gives Taylor a method which, under certain assumptions, allows him to assign appropriate probabilities to sound sequences  $O$ 's and word sequences  $S$ 's and to make reasonable deductions about them.
- A hidden Markov model (HMM) allows us to talk about both observed events Hidden Markov model (like words that we see in the input) and hidden events (like part-of-speech tags) that we think of as causal factors in our probabilistic model. An HMM is specified by the following components:

$Q = q_1 q_2 \dots q_N$	a set of $N$ states
$A = a_{11} \dots a_{ij} \dots a_{NN}$	a <b>transition probability matrix</b> $A$ , each $a_{ij}$ representing the probability of moving from state $i$ to state $j$ , s.t. $\sum_{j=1}^N a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_T$	a sequence of $T$ <b>observations</b> , each one drawn from a vocabulary $V = v_1, v_2, \dots, v_V$
$B = b_i(o_t)$	a sequence of <b>observation likelihoods</b> , also called <b>emission probabilities</b> , each expressing the probability of an observation $o_t$ being generated from a state $i$
$\pi = \pi_1, \pi_2, \dots, \pi_N$	an <b>initial probability distribution</b> over states. $\pi_i$ is the probability that the Markov chain will start in state $i$ . Some states $j$ may have $\pi_j = 0$ , meaning that they cannot be initial states. Also, $\sum_{i=1}^n \pi_i = 1$

- A first-order hidden Markov model instantiates two simplifying assumptions.
  - First, as with a first-order Markov chain, the probability of a particular state depends only on the previous state:

$$\text{MarkovAssumption} : P(q_i | q_1 \dots q_{i-1}) = P(q_i | q_{i-1})$$

- Second, the probability of an output observation  $o_i$  depends only on the state that produced the observation  $q_i$  and not on any other states or any other observations:

$$\text{OutputIndependence} : P(o_i | q_1 \dots q_i, \dots, q_T, o_1, \dots, o_i, \dots, o_T) = P(o_i | q_i)$$

- State for Markov model
  - The state could be a data point  $x_i$  (Markov Chain classifier)
  - The state could be an object (object ranking)
  - The state could be the destination of a link (internet search engines)

- State transition matrices:

A stationary Markov chain with  $N$  states is described by an  $N \times N$  transition matrix with constraints:

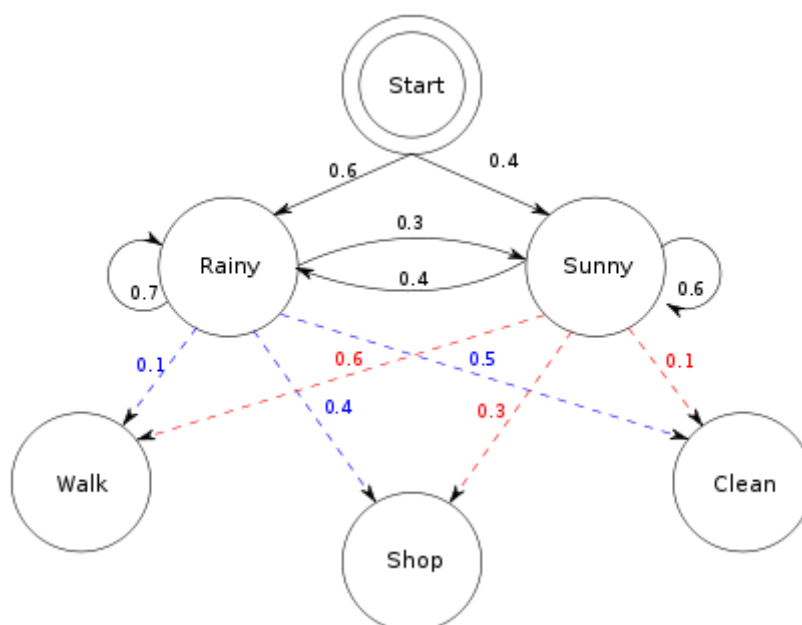
$$Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}$$

$$q_{ij} \triangleq P(x_{t+1} = i \mid x_t = j) \quad q_{ij} \geq 0 \quad \sum_{i=1}^N q_{ij} = 1 \quad \text{for all } j$$

Notice that the sum of each row equals 1 (think why). Such a matrix is called a Stochastic Matrix. The  $(i,j)$  is defined as  $p_{i,j}$  -the transition probability between  $i$  and  $j$ .

- Example:

- Consider two friends, Alice and Bob, who live far apart from each other and who talk together daily over the telephone about what they did that day.
- Bob is only interested in three activities: walking in the park, shopping, and cleaning his apartment. The choice of what to do is determined exclusively by the weather on a given day.
- Alice has no definite information about the weather, but she knows general trends. Based on what Bob tells her he did each day, Alice tries to guess what the weather must have been like.
- Alice believes that the weather operates as a discrete Markov chain. There are two states, "Rainy" and "Sunny", but she cannot observe them directly, that is, they are hidden from her.
- On each day, there is a certain chance that Bob will perform one of the following activities, depending on the weather: "walk", "shop", or "clean". Since Bob tells Alice about his activities, those are the observations. The entire system is that of a hidden Markov model (HMM).



- Applications:

HMMs can be applied in many fields where the goal is to recover a data sequence that is not immediately observable (but other data that depend on the sequence are). Applications include:

1. Computational finance
2. Single-molecule kinetic analysis
3. Cryptanalysis
4. Speech recognition, including Siri
5. Speech synthesis
6. Part-of-speech tagging
7. Document separation in scanning solutions
8. Machine translation
9. Partial discharge
10. Gene prediction

## 5 HMM Parameters

- A HMM consists of a number of states. Each state  $j$  has an associated observation probability distribution  $b_j(o_t)$  which determines the probability of generating observation  $o_t$  at time  $t$  and each pair of states  $i$  and  $j$  has an associated transition probability  $a_{ij}$ . In HTK the entry state 1 and the exit state  $N$  of an  $N$  state HMM are non-emitting.

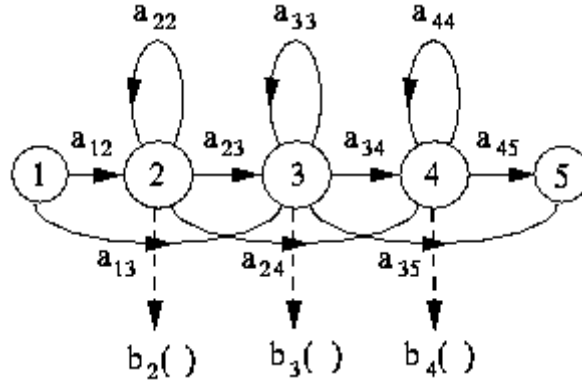


Figure 5: Simple Left-Right HMM

- Fig. 5 shows a simple left-right HMM with five states in total. Three of these are emitting states and have output probability distributions associated with them. The transition matrix for this model will have 5 rows and 5 columns. Each row will sum to one except for the final row which is always all zero since no transitions are allowed out of the final state.

HTK is principally concerned with continuous density models in which each observation probability distribution is represented by a mixture Gaussian density. In this case, for state  $j$  the probability  $b_j(o_t)$  of generating observation  $o_t$  is given by

$$b_j(o_t) = \prod_{s=1}^S \left[ \sum_{m=1}^{M_{js}} c_{j_{sm}} N(o_{st}; \mu_{j_{sm}}, \Sigma_{j_{sm}}) \right]^{y_s}$$

where  $M_{js}$  is the number of mixture components in state  $j$  for stream  $s$ ,  $c_{j_{sm}}$  is the weight of the  $m$ 'th component and  $N(o; \mu, \Sigma)$  is a multivariate Gaussian with mean vector  $\mu$  and covariance matrix  $\Sigma$ , that is

$$N(o; \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{-\frac{1}{2}(o-\mu)^T \Sigma^{-1} (o-\mu)}$$

where  $n$  is the dimensionality of  $o$ . The exponent  $y_s$  is a stream weight and its default value is one. Other values can be used to emphasise particular streams. However, none of the standard HTK tools manipulate it.

- HTK also supports discrete probability distributions in which case

$$b_j(o_t) = \prod_{s=1}^S \{P_{js}[v_s(o_{st})]\}^{y_s}$$

where  $[v_s(o_{st})]$  is the output of the vector quantiser for stream  $s$  given input vector  $o_{st}$  and  $P_{js}[v]$  is the probability of state  $j$  generating symbol  $v$  in stream  $s$ .

- In addition to the above, any model or state can have an associated vector of duration parameters  $d_k$ . Also, it is necessary to specify the kind of the observation vectors, and the width of the observation vector in each stream. Thus, the total information needed to define a single HMM is as follows

- type of observation vector
- number and width of each data stream



- optional model duration parameter vector
- number of states
- for each emitting state and each stream
  - \* mixture component weights or discrete probabilities
  - \* if continuous density, then means and covariances
  - \* optional stream weight vector
  - \* optional duration parameter vector
- transition matrix

## 6 Problems of HMM

- Evaluation:  
How likely is a given sequence of expressions, given how our model works ?
- Decoding:
  - What is the most likely sequence of moods, given the sequence of expressions and how our model works ?
  - What is the probability that the mood at frame 4 was ‘happy’ ?
- Learning:  
Given many sequences of expressions, how do we determine the parameters of the model? (Transition Matrix, Observation Probabilities)

### 1. The Evaluation Problem and the Forward Algorithm

- The Evaluation Problem and the Forward Algorithm We have a model  $\lambda = (A, B, \pi)$  and a sequence of observations  $O = o_1, o_2, \dots, o_T$ , and  $p\{O|\lambda\}$  must be found. We can calculate this quantity using simple probabilistic arguments. But this calculation involves number of operations in the order of  $N^T$ . This is very large even if the length of the sequence,  $T$  is moderate. Therefore we have to look for an other method for this calculation. Fortunately there exists one which has a considerably low complexity and makes use an auxiliary variable,  $\alpha_t(i)$  called forward variable.
- The forward variable is defined as the probability of the partial observation sequence  $o_1, o_2, \dots, o_T$ , when it terminates at the state  $i$ . Mathematically,

$$\alpha_t(i) = p\{o_1, o_2, \dots, o_t, q_t = i|\lambda\}$$

Then it is easy to see that following recursive relationship holds.

$$\alpha_{t+1}(j) = b_j(o_{t+1}) \sum_{i=1}^N \alpha_t(i) a_{ij}, 1 \leq j \leq N, 1 \leq t \leq T - 1$$

where,

$$\alpha_t(j) = \pi_j b_j(o_t), 1 \leq j \leq N$$

Using this recursion we can calculate

$$\alpha_T(i), 1 \leq i \leq N$$

and then the required probability is given by,

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

The complexity of this method, known as the forward algorithm is proportional to  $N^2T$ , which is linear wrt  $T$  whereas the direct calculation mentioned earlier, had an exponential complexity.

In a similar way we can define the backward variable  $\beta_t(i)$  as the probability of the partial observation sequence  $o_1, o_2, \dots, o_T$ , given that the current state is  $i$ . Mathematically,

$$\beta_t(i) = p\{o_{t+1}, o_{t+2}, \dots, o_T | q_t = i, \lambda\}$$

As in the case of  $\alpha_t(i)$  there is a recursive relationship which can be used to calculate  $\beta_t(i)$  efficiently.

$$\beta_t(i) = \sum_{j=1}^N \beta_{t+1}(j) a_{ij} b_j(o_{t+1}), 1 \leq i \leq N, 1 \leq t \leq T - 1$$

where,

$$\beta_T(i) = 1, 1 \leq i \leq N$$

Further we can see that,

$$\alpha_t(i)\beta_t(i) = p\{O, q_t = i|\lambda\}, 1 \leq i \leq N, 1 \leq t \leq T$$

Therefore this gives another way to calculate  $p\{O|\lambda\}$ , by using both forward and backward variables as given in below equation.

$$p\{O|\lambda\} = \sum_{i=1}^N p\{O, q_t = i|\lambda\} = \sum_{i=1}^N \alpha_t(i)\beta_t(i)$$

Above equation is very useful, specially in deriving the formulas required for gradient based training.

## 2. The Decoding Problem and the Viterbi Algorithm

- In this case We want to find the most likely state sequence for a given sequence of observations,  $O = o_1, o_2, \dots, o_T$  and a model,  $\lambda = (A, B, \pi)$ .
- The solution to this problem depends upon the way "most likely state sequence" is defined. One approach is to find the most likely state  $q_t$  at  $t=t$  and to concatenate all such 'q<sub>t</sub>'s. But some times this method does not give a physically meaningful state sequence. Therefore we would go for another method which has no such problems.
- In this method, commonly known as Viterbi algorithm, the whole state sequence with the maximum likelihood is found. In order to facilitate the computation we define an auxiliary variable,

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

which gives the highest probability that partial observation sequence and state sequence up to  $t=t$  can have, when the current state is  $i$ .

It is easy to observe that the following recursive relationship holds.

$$\delta_{t+1}(j) = b_j(o_{t+1}) \left[ \max_{1 \leq i \leq N} \delta_t(i) a_{ij} \right], 1 \leq i \leq N, 1 \leq t \leq T - 1$$

where,

$$\delta_t(j) = \pi_j b_j(o_t), 1 \leq j \leq N$$

So the procedure to find the most likely state sequence starts from calculation of  $\delta_T(j)$ ,  $1 \leq j \leq N$  using recursion in equation of  $\delta_{t+1}(j)$ , while always keeping a pointer to the "winning state" in the maximum finding operation. Finally the state  $j^*$ , is found where

$$j^* = \arg \max_{1 \leq j \leq N} \delta_T(j)$$

and starting from this state, the sequence of states is back-tracked as the pointer in each state indicates. This gives the required set of states.

This whole algorithm can be interpreted as a search in a graph whose nodes are formed by the states of the HMM in each of the time instant  $t$ ,  $1 \leq t \leq T$

### 3. The Learning Problem

- Generally, the learning problem is how to adjust the HMM parameters, so that the given set of observations (called the training set) is represented by the model in the best way for the intended application.
- Thus it would be clear that the "quantity" we wish to optimize during the learning process can be different from application to application.
- In other words there may be several optimization criteria for learning, out of which a suitable one is selected depending on the application.
- There are two main optimization criteria found in ASR literature;
  - (a) Maximum Likelihood (ML)
  - (b) Maximum Mutual Information (MMI)

## 7 References

- Speech and Language Processing. Daniel Jurafsky James H. Martin.  
<https://web.stanford.edu/~jurafsky/slp3/A.pdf>
- Three basic problems of HMMs  
<http://jedlik.phy.bme.hu/~gerjanos/HMM/node6.html>.
- Hidden Markov model - Wikipedia  
[https://en.wikipedia.org/wiki/Hidden\\_Markov\\_model](https://en.wikipedia.org/wiki/Hidden_Markov_model).
- Autoregressive–moving-average model - Wikipedia  
[https://en.wikipedia.org/wiki/Autoregressive-moving-average\\_model](https://en.wikipedia.org/wiki/Autoregressive-moving-average_model)