INFERENCE ON MARKOV CHAINS

A Taghavi, J Kurra, S Sahu

Abstract

Statistical inference is widely used to make propositions about a population, using data drawn from the population with some form of sampling. Any statistical inference requires some assumptions. In this project, the Markov Chain has been used to model the set of assumptions. A widely popular statistical Markov model - the hidden Markov Model has been discussed. Subsequently, the three main algorithms for performing inference in Markov Chains have been examined. Finally, examples have been presented to explain the implementation of the stipulated algorithms.

I. INTRODUCTION

II. STATISTICAL INFERENCE

Statistical inference deduces properties of an underlying distribution by analysis of data. In other words, it infers properties about a population: this includes testing hypotheses and deriving estimates. The population is assumed to be larger than the observed data set. Usually, the observed data is assumed to be sampled from a larger population. Given a hypothesis about a population, for which we wish to draw inferences, statistical inference consists of (firstly) selecting a statistical model of the process that generates the data and (secondly) deducing propositions from the model.

A. Statistical Model

A statistical model is a set of assumptions concerning the generation of the observed data and similar data. In simple words, a statistical model represents the data-generating process. The assumptions embodied by a statistical model describe a set of probability distributions, some of which are assumed to adequately approximate the distribution from which a particular data set is sampled.

B. Paradigms for Inference

In the context of statistical of inference, we have four paradigms namely,

- Classical Statistics or Error Statistics
- Bayesian Statistics
- Likelihood-Based Statistics
- The Akaikean-Information Criterion-Based Statistics

Frequentist Inference draws conclusions from the observed data by emphasizing the frequency or proportion of the data. Frequentist inference does not formally take into consideration the expectation of other probability distributions which may influence the distribution under study.

In the Bayesian Inference, Bayes' theorem is used to update the probability for a hypothesis as more evidence or information becomes available. Bayesian inference is an important technique in statistics, and especially in mathematical statistics. Bayesian updating is particularly important in the dynamic analysis of a sequence of data.

C. Statistical Proposition

The conclusion of a statistical inference is a statistical proposition. Below are some common forms of statistical proposition:

- **A Point Estimate:** A particular value that best approximates some parameter of interest;
- **An Interval Estimate:** An example is a confidence interval (or set estimate), which is an interval constructed using a dataset drawn from a population so that, under repeated sampling of such datasets, such intervals would contain the true parameter value with the probability at the stated confidence level;
- **A Credible Interval:** An example is a set of values containing 95% of posterior belief
- Rejection of a Hypothesis
- Clustering or Classification of Data Points into Groups.

III. MARKOV CHAIN

A. Markov Property

Generally speaking, the term Markov property refers to the memory-less property of a stochastic process. In other words, a stochastic process is said to have the Markov property if the conditional probability distribution of future states of the process (conditional on both past and present states) depends only upon the present state and not on the sequence of events that preceded it.

A stochastic process which possesses this property is called a Markov process. A model where the Markov property is assumed to hold is known as the Markov Assumption. One example of the Markov Assumption is the hidden Markov Model. It shall be discussed in the later sections.

B. Markov Chain

Markov Chain is the most popular Markov process. It refers to the sequence of random variables such a process moves through, with the Markov property defining serial dependence only between adjacent periods (as in a "chain").

Thus, a stochastic process $X = \{X_n ; n \epsilon N\}$ is said to be a Markov Chain when,

$$P\{X_{n+1} = j \mid X_0, ..., X_n\} = P\{X_{n+1} = j \mid X_n\} \quad (1)$$

The "next" state X_{n+1} of the process must be independent of the "past" states $X_0, \ldots, X_{n-1}$ provided that the "present" state X_n is known,

$$P\{X_{n+1} = j \mid X_n = i\} = P(i,j) \qquad i,j \in E \quad (2)$$

Markov Chain can, therefore, be used for describing systems that follow a chain of linked events, where what happens next depends only on the current state of the system.

The changes of state of the system are called transitions. The probabilities associated with various state changes are called transition probabilities. The process is characterized by a state space, a transition matrix describing the probabilities of particular transitions, and an initial state (or initial distribution) across the state space. The transition matrix is given as,

$$T = \begin{bmatrix} P(0,0) & P(0,1) & P(0,2) & \cdots \\ P(1,0) & P(1,1) & P(1,2) & \cdots \\ P(2,0) & P(2,1) & P(2,2) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \quad (3)$$

where, $P(i,j) \geq 0$ for any $i,j \in E$

and $\sum_{j \in E} P(i,j) = 1$

By convention, all possible states and transitions are assumed to have been included in the definition of the process, so there is always a next state, and the process does not terminate.

Random Walk, Gambling and a Simple Weather Model are a few examples of Markov Chain.

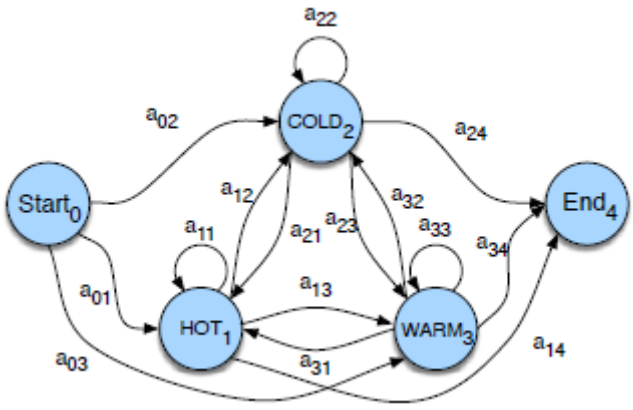

Figure 1. A Markov Chain for Weather Model

IV. HIDDEN MARKOV MODEL

A hidden Markov model (HMM) is a statistical Markov model in which the system being modelled is assumed to be a Markov process with unobserved (hidden) states. An hidden Markov Model can be presented as the simplest dynamic Bayesian network.

A Markov chain is useful when we need to compute a probability for a sequence of events that are observed in the world. In many cases, however, the desired events may not be directly observable in the world. For example, in part-of-speech tagging (assigning tags like Noun and Verb to words) part-of-speech tags in the world are not observed; words need to be analyzed and the correct tags have to be inferred from the word sequence. The part-of-speech tags are, thus, said to be hidden because they are not observed. The same architecture comes up in speech recognition when acoustic events in the world are considered and the presence of "hidden" words, that are the underlying causal source of the acoustics, have to be inferred. A hidden Markov model gives information about both observed events (like words in the input) and hidden events (like part-of-speech tags) that are considered as the causal factors in that probabilistic model.

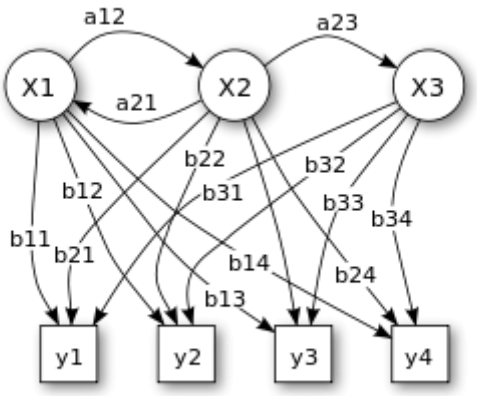

Figure 2. Probabilistic parameters of a hidden Markov model

A typical hidden Markov Model looks as the one shown in Fig 2. Here,

- X represents the states (hidden, to be inferred)
- y represents observations
- a_{ij} represents state transition probabilities (probability of transition from state i to state j)
- $b_i(y_t)$ represents emission probabilities (probability of an observation y_t being generated from a state i)

A hidden Markov Model which has non-zero probability of transition between any two states is known as a fully connected or ergodic HMM.

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

Markov Assumption:

$$P\{X_i \mid X_1, ..., X_{i-1}\} = P\{X_i \mid X_{i-1}\} \quad (4)$$

Secondly, the probability of an output observation y_i depends only on the state that produced the observation and not on any other states or any other observations.

Output Independence:

$$P\{y_t \mid X_1, X_2, .., X_i, .., X_T, y_1, y_2, .., y_t, .., y_T\} = P\{y_t \mid X_i\} \quad (5)$$

Hidden Markov models are mostly used in temporal pattern recognition such as speech, handwriting, gesture recognition, part-of-speech tagging, musical score following, partial discharges and bioinformatics.

V. INFERENCE IN MARKOV CHAINS

In 1989, Rabiner introduced, in his paper named "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", introduced the idea that hidden Markov Models should be characterized by three fundamental problems,

- Likelihood: Given an HMM $\lambda = (A, B)$ and an observation sequence Y, the likelihood $P(Y \mid \lambda)$ is determined.
- Decoding: Given an observation sequence Y and an HMM $\lambda = (A, B)$, the best hidden state sequence X is discovered.
- Learning: Given an observation sequence Y and the set of states in the HMM, the HMM parameters A and B are learned.

There is one algorithm for each problem described above. The Forward Algorithm uses the concept of Likelihood for inference in the Markov Chains. The Forward Algorithm is used to infer about the observation sequence. Similarly, the Decoding problem is implemented using the Viterbi Algorithm. The states responsible for emission could be inferred using the Viterbi Algorithm. The Baum-Welch Algorithm implements the Learning problem. An attempt is made to learn the state transition probabilities such that the required emissions are received from the desired states. All the three algorithms shall be discussed in the next sections.

VI. FORWARD ALGORITHM

The Forward Algorithm computes P(y) under the model. Because many different state paths can give rise to the same sequence y, we must add the probabilities for all possible paths to obtain the full probability of y,

$$P(y) = \sum_{\pi} P(y, \pi) \quad (6)$$

where, π is an event in which a specific path was taken through the HMM.

For an HMM with N hidden states and an observation sequence of T observations, there are NT possible hidden sequences. For real tasks, where N and T are both large, NT is a very large number, the total observation

likelihood cannot be computed by computing a separate observation likelihood for each hidden state sequence and then summing them. In order to overcome the complexity of such an extremely exponential algorithm, an efficient O(N2T) algorithm called the forward algorithm was proposed.

The forward algorithm is a kind of dynamic programming algorithm, that is, an algorithm that uses a table to store intermediate values as it builds up the probability of the observation sequence. The forward algorithm computes the observation probability by summing over the probabilities of all possible hidden state paths that could generate the observation sequence, but it does so efficiently by implicitly folding each of these paths into a single forward trellis.

Each cell of the forward algorithm trellis $\alpha_t(j)$ represents the probability of being in state j after seeing the first t observations, given the automaton λ. The value of each cell $\alpha_t(j)$ is computed by summing over the probabilities of every path that could lead us to this cell. Formally, each cell expresses the following probability,

$$\alpha_t(j) = P(y_1, y_2, ..., y_t, X_t = j | \lambda) \tag{7}$$

Here, $X_t = j$ means "the t^{th} state in the sequence of states is state j". We compute this probability $\alpha_t(j)$ by summing over the extensions of all the paths that lead to the current cell. For a given state X_j at time t, the value $\alpha_t(j)$ is computed as,

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(j)\, a_{ij}\, b_j(y_t) \tag{8}$$

The three factors that are multiplied in (8) in extending the previous paths to compute the forward probability at time t are

$\alpha_{t-1}(i)$ - the previous forward path probability from the previous time step

a_{ij} - the transition probability from previous state y_i to current state y_j

$b_j(y_t)$ - the state observation likelihood of the observation symbol y_t given the current state j

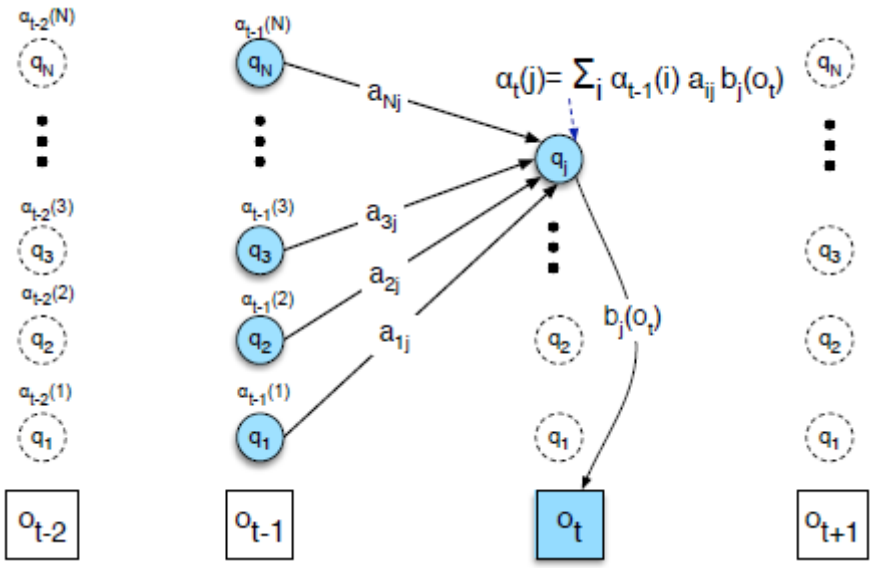

Figure 3. Visualizing the computation of a single element $\alpha_t(i)$ in the trellis by summing all the previous values α_{t-1}, weighted by their transition probabilities a, and multiplying by the observation probability $b_i(y_{t+1})$

Fig 3 shows a visualization of the induction step for computing the value in one new cell of the trellis. For many applications of HMMs, many of the transition probabilities are 0, so not all previous states will contribute to the forward probability of the current state.

The definitional recursion is given as,

1. Initialization:

$$\alpha_t(j) = a_{0j}\, b_j(y_1) \qquad 1 \le j \le N \tag{9}$$

2. Recursion (since states 0 and X_F are non-emitting):

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij}\, b_j(y_j) \qquad 1 \le j \le N, 1 < t \le T \tag{10}$$

3. Termination:

$$P(Y|\lambda) = \alpha_T(X_F) = \sum_{i=1}^{N} \alpha_T(i) a_{iF} \tag{11}$$

VII. VITERBI ALGORITHM

For any hidden Markov Model, the task of determining which sequence of variables is the underlying source of some sequence of observations is called the decoding task.

For each possible sequence of hidden states, the forward algorithm could be run and the likelihood of the observation sequence given that hidden state sequence could be computed. Then, the hidden state sequence with the maximum observation likelihood could be chosen. However, this is not preferable because there are an exponentially large number of state sequences. Instead, the most common decoding algorithms for HMMs is the Viterbi algorithm. Like the forward algorithm, Viterbi is a kind of dynamic programming that makes uses of a dynamic programming trellis.

The idea is to process the observation sequence left to right, filling out the trellis. Each cell of the trellis, $v_t(j)$, represents the probability that the HMM is in state j after seeing the first t observations and passing through the most probable state sequence $X_0, X_1, X_2, \ldots, X_{t-1}$, given the automaton λ. The value of each cell $v_t(j)$ is computed by recursively taking the most probable path that could lead us to this cell. Formally, each cell expresses the probability

$$v_t(j) = \max_{X_0, X_1, .., X_{t-1}} P(X_0, X_1, \ldots, X_{t-1}, y_0, y_1, \ldots, y_{t-1}, X_t = j|\lambda) \tag{12}$$

The most probable path is represented by taking the maximum over all possible previous state sequences $\max_{X_0, X_1, .., X_{t-1}}$. Like other dynamic programming algorithms, Viterbi fills each cell recursively. Given that probability of being in every state at time t - 1 has been computed already, the Viterbi probability is then calculated by taking the most probable of the extensions of the paths that lead to the current cell. For a given state X_j at time t, the value $v_t(j)$ is computed as

$$v_t(j) = \max_{1 \le i \le N} v_{t-1}(i) a_{ij}\, b_j(y_t) \tag{13}$$

The three factors that are multiplied in (13) for extending the previous paths to compute the Viterbi probability at time *t* are

$v_{t-1}(i)$ - the previous Viterbi path probability from the previous time step

a_{ij} - the transition probability from previous state X_i to current state X_j

$b_j(y_t)$ - the state observation likelihood of the observation symbol y_t given the current state *j*

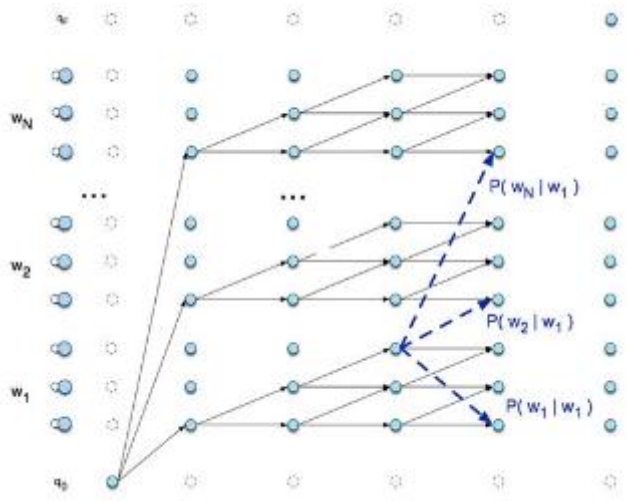

Figure 4. The Viterbi trellis for computing the best path through the hidden state space

Fig 4 shows an example of the Viterbi Trellis for computing the best hidden state sequence for the given observation sequence. The Viterbi algorithm is identical to the forward algorithm except that it takes the max over the previous path probabilities whereas the forward algorithm takes the sum. The Viterbi algorithm has one component that the forward algorithm does not have: *Backpointers*. The reason is that while the forward algorithm needs to produce an observation likelihood, the Viterbi algorithm must produce a probability and also the most likely state sequence. The best state sequence is computed by keeping track of the path of hidden states that led to each state and then at the end back-tracing the best path to the beginning (the Viterbi backtrace).

The definition for the Viterbi Recursion is as follows,

1. Initialization:

$$v_1(j) = a_{0j}\, b_j(y_1) \qquad 1 \le j \le N \tag{14}$$

$$bt_1(j) = 0 \tag{15}$$

2. Recursion (since states 0 and X_F are non-emitting):

$$v_t(j) = \max_{1 \le i \le N} v_{t-1}(i) a_{ij}\, b_j(y_t) \qquad 1 \le j \le N, 1 < t \le T \tag{16}$$

$$bt_1(j) = \operatorname{argmax}_{1 \le i \le N} v_{t-1}(i) a_{ij}\, b_j(y_t) \qquad 1 \le j \le N, 1 < t \le T \tag{17}$$

3. Termination:

$$\text{The Best Score: } P* = v_T(X_F) = \max_{1 \le i \le N} v_T(i) * a_{iF} \tag{18}$$

$$\text{The Start of Back Trace: } X_T* = bt_T(X_F) = \operatorname*{argmax}_{1 \le i \le N} v_T(i) * a_{iF} \tag{19}$$

VIII. BAUM-WELCH ALGORITHM

The standard algorithm for HMM training is the forward-backward, or Baum Welch algorithm, a special case of the Expectation-Maximization or EM algorithm. The algorithm could be used to train both the transition probabilities A and the emission probabilities B of the HMM. Expectation Maximization is essentially an iterative algorithm. It works by computing an initial estimate for the probabilities, then uses those estimates to compute a better estimate, and so on, iteratively improving the probabilities that it learns.

In order to avoid complexity, consider the training of a Markov chain rather than a hidden Markov model. Since the states in a Markov chain are observed, the model could be run on the observation sequence and the path taken through the model and the state that generated each observation symbol could be deduced. A Markov chain of course has no emission probabilities B (alternatively, a Markov chain could be viewed as a degenerate hidden Markov model where all the *b* probabilities are 1.0 for the observed symbol and 0 for all other symbols). Thus, the only probabilities to be trained are the transition probability matrix A. We get the maximum likelihood estimate of the probability a_{ij} of a particular transition between states *i* and *j* could be obtained by counting the number of times the transition was taken, which is given by *C(i→j)*, and then normalizing it by the total count of all times taken for any transition from state *i*,

$$a_{ij} = \frac{C(i \rightarrow j)}{\sum_{q \in Q} C(i \rightarrow q)} \tag{20}$$

This probability could be directly computed in a Markov chain because the current states are known. However, for an HMM, they cannot be computed directly from an observation sequence since not much information is known about the path of states which was taken through the machine for a given input. The Baum-Welch algorithm uses two neat intuitions to solve this problem. The first idea is to iteratively estimate the counts. Initially, an estimate for the transition and observation probabilities is taken and then these estimated probabilities are used to derive better and better probabilities. The second idea is that estimated probabilities are obtained by computing the forward probability for an observation and then dividing that probability mass among all the different paths that contributed to this forward probability.

To understand the algorithm, it is required to know more about a useful probability related to the forward probability, called the backward probability. The backward probability β is the probability of seeing the observations from time *t +1* to the end, given that we are in state *i* at time *t* (and given the automaton λ):

$$\beta_t(i) = P(y_{t+1}, y_{t+2}, \ldots, y_T | X_t = i, \lambda) \tag{21}$$

It is computed inductively in a similar manner to the forward algorithm.

1. Initialization:

$$\beta_T(j) = a_{iF} \qquad 1 \leq i \leq N \tag{22}$$

2. Recursion (since states 0 and X_F are non-emitting):

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(y_{t+1}) \beta_{t+1}(j) \qquad 1 \leq i \leq N, 1 < t \leq T \tag{23}$$

3. Termination:

$$P(Y|\lambda) = \alpha_T(X_F) = \beta_1(X_0) = \sum_{i=1}^{N} a_{0j}\, b_j(y_1) \beta_1(j) \tag{24}$$

The estimate for state transition probabilities is given as,

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)} \tag{25}$$

where, $\xi_t(i,j) = \frac{\alpha_t(i) a_{ij} b_j(y_{t+1}) \beta_{t+1}(j)}{\alpha_T(X_F)}$ (26)

Likewise, the estimate for emission probabilities is given as,

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 \; s.t. \; Y_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{27}$$

where, $\gamma_t(j) = \frac{\alpha_t(j) \beta_t(j)}{P(Y|\lambda)}$ (28)

These re-estimations form the core of the iterative forward-backward algorithm. The forward-backward algorithm starts with some initial estimate of the HMM parameters λ = (A;B). Next, two steps are iteratively run. Like other cases of the EM (expectation-maximization) algorithm, the forward-backward algorithm has two steps: the expectation step, or E-step, and the maximization step, or M-step. In the E-step, the expected state occupancy count g and the expected state transition count x from the earlier A and B probabilities are computed. In the M-step, g and x are used to re-compute new A and B probabilities. Although in principle the forward-backward algorithm can do completely unsupervised learning of the A and B parameters, in practice the initial conditions are very important. For this reason the algorithm is often given extra information.

IX. EXAMPLE

Consider a village where all villagers are either healthy or have a fever and only the village doctor can determine whether each has a fever. The doctor diagnoses fever by asking patients how they feel. The villagers may only answer that they feel normal, dizzy, or cold.

The doctor believes that the health condition of his patients operate as a discrete Markov chain. There are two states, "Healthy" and "Fever", but the doctor cannot observe them directly; they are hidden from him. On each day, there is a certain chance that the patient will tell the doctor he/she is "normal", "cold", or "dizzy", depending on her health condition.

The observations (normal, cold, dizzy) along with a hidden state (healthy, fever) form a hidden Markov model (HMM), and can be represented as shown in Fig. 5.

Here, the Start Probability represents the doctor's belief about which state the HMM is in when the patient first visits (all he knows is that the patient tends to be healthy). The Transition Probability represents the change of the health condition in the underlying Markov chain. In this example, there is only a 30% chance that tomorrow the patient will have a fever if he is healthy today. The Emission Probability represents how likely the patient is to feel on each day. If he is healthy, there is a 50% chance that he feels normal; if he has a fever, there is a 60% chance that he feels dizzy.

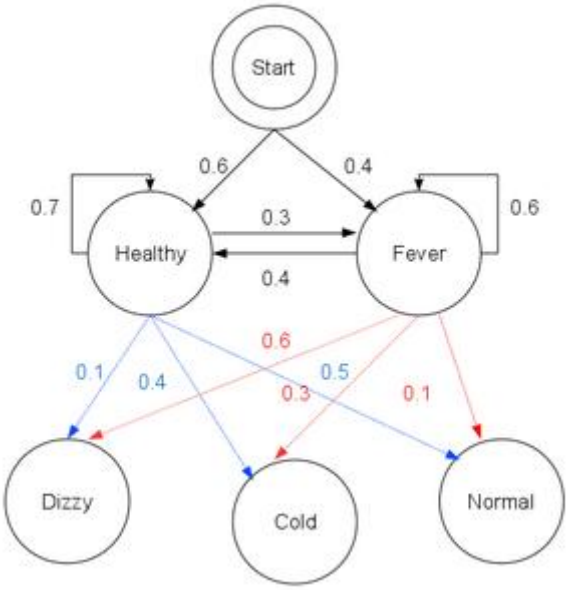

Figure 5. Hidden Markov Model for the given situation (example)

The patient visits three days in a row and the doctor discovers that on the first day she feels normal, on the second day she feels cold, on the third day she feels dizzy. The question is: what is the most likely sequence of health conditions of the patient that would explain these observations?

This shall be answered using the Viterbi Algorithm. The Trellis Structure in its initial stage is as shown in Fig 6.

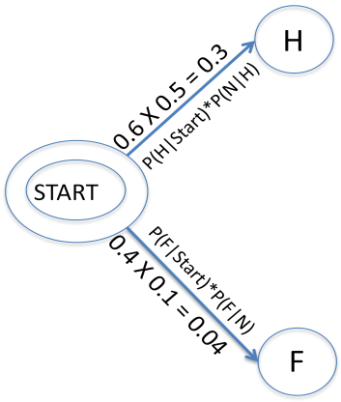

Figure 6. Trellis Structure for Day 1 Observation: Person is feeling "Normal"

Initially, on Day 1, when the person feels Normal, the probabilities of the person being Healthy and the person having a Fever are computed using the Start and Emission probabilities. Thus,

Probability of the person being Healthy on Day 1 = 1 X 0.3 = 0.3

Probability of the person having Fever on Day 1= 1 X 0.04 = 0.04

As there is just one path through which each state could be reached, there is no search for the most probable path.

Moving on to Day 2, when the person feels "Cold", the Trellis Structure is updated to as shown in Figure 7.

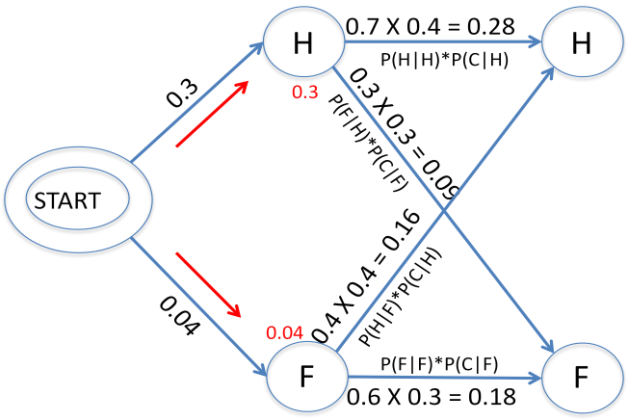

Figure 7. Trellis Structure for Day 2 Observation: Person is feeling “Cold”

Now, the probabilities of the person being Healthy and the person having Fever are computed using the Transition Probabilities and Emission Probabilities. The Person could be Healthy in two ways - firstly, when the person is healthy on Day 1 and she continues to be Healthy on Day 2 as well; secondly, if the person has fever on Day 1 and then, she becomes Healthy on Day 2. Therefore,

Probability of the person being Healthy on Day 2 = max(0.3*0.28, 0.04*0.16) = max(0.084, 0.0064) = 0.084

Probability of the person having Fever on Day 2 = max(0.3*0.09, 0.04*0.18) = max(0.027, 0.0072) = 0.027

As there are multiple paths to reach any given state, the least probable paths are eliminated and only the most probable paths are retained before proceeding to the next stage of Trellis Structure (Day 3). The resulting Trellis Structure is shown in Figure 8.

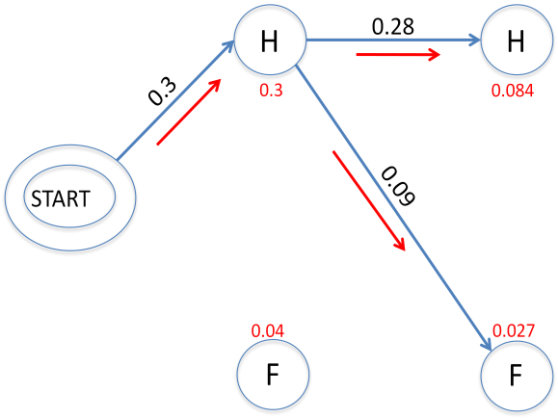

Figure 8 Updated Trellis Structure for Day 2 Observation

Finally, on Day 3, when the person reports that she is feeling "Dizzy", the Trellis Structure gets updated to what is presented in Figure 9.

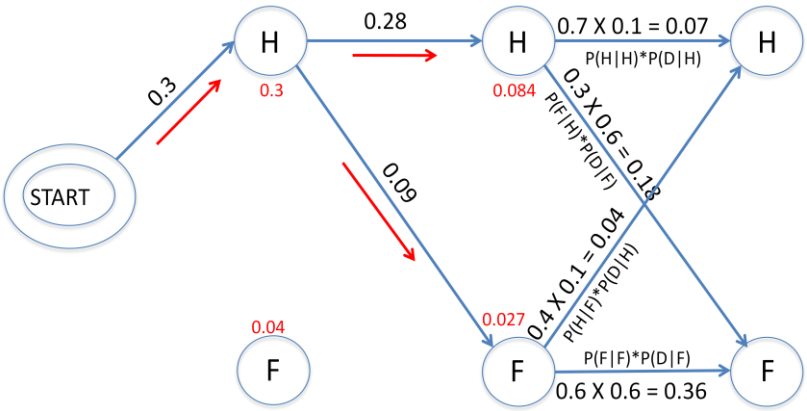

Figure 9. Trellis Structure for Day 3 Observation: Person is feeling "Dizzy"

Similar to how the probabilities for the person to be healthy and the person having a fever on Day 2 were computed, the probabilities for Day 3 are calculated as well.

Probability of the person being Healthy on Day 3 = max(0.084*0.07, 0.027*0.04) = max(0.00588, 0.00108)

= 0.00588

Probability of the person having Fever on Day 3 = max(0.084*0.18, 0.027*0.36) = max(0.01512, 0.00972)

= 0.01512

As earlier, since, multiple paths exist to reach a given state for Day 3 observation as well, the paths with least probability are removed and the Trellis Structure is updated to reflect the changes (refer Fig. 10).

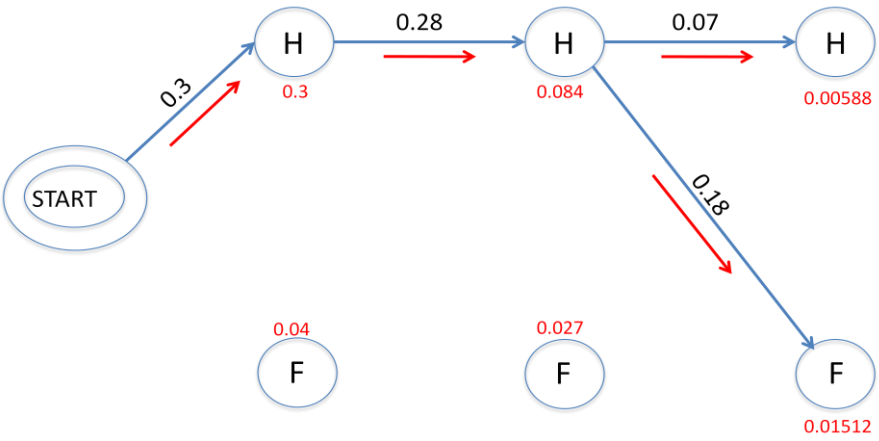

Figure 10. Updated Trellis Structure for Day 3 Observation

Now, when all the observations are exhausted, the most probable state is observed. In other words, the state which is more probable at the end of the Trellis Structure is deduced and the path to that state is retained. Hence, the Trellis Structure looks as shown in Fig 11. The path to the other state is eliminated. Usually, just like Start

Probability, End Probability is also provided. In such case, only a single path is received at the end and the most probable states are quite clear. No further analysis is needed in such a case.

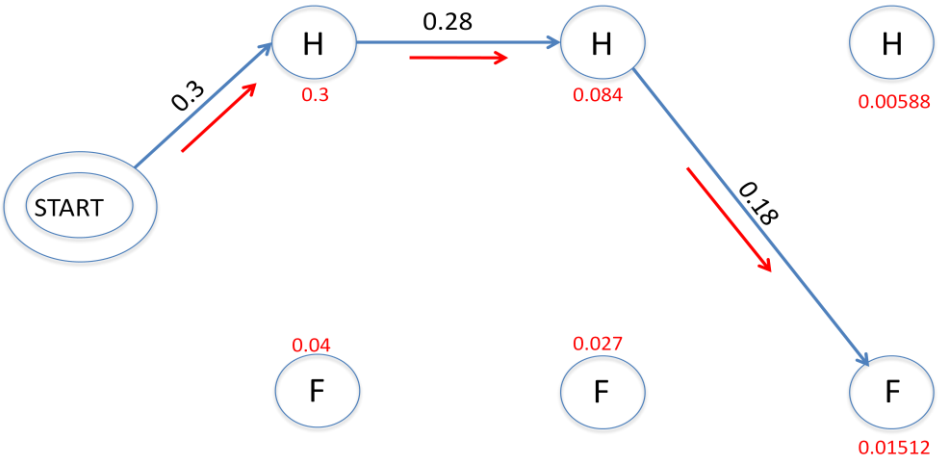

Figure 11. Final Trellis Structure

Now, the whole Viterbi path is back-traced to know all the probable states for all the three days. As could be seen from Fig 11, it is clear that the observation set {'Normal', 'Cold', 'Dizzy'} are most likely to be generated by the states {'Healthy', 'Healthy', 'Fever'}.

X. CONCLUSION

.