International Research Journal of Computer Science (IRJCS)
Issue 08, Volume 07 (August 2020)
ISSN: 2393-9842
https://www.irjcs.com/archives

AN ANALYSIS OF CONCEPTS AND TECHNIQUES CONCERNING THE USE OF HIDDEN MARKOV MODELS FOR SEQUENTIAL DATA

Advait Pravin Savant
Sardar Patel Institute of Technology, Mumbai, India
adisav17@gmail.com

Publication History
Manuscript Reference No: IRJCS/RS/Vol.07/Issue08/AUCS10083
Received: 09, August 2020
Accepted: 19, August 2020
Published: 31, August 2020
DOI: https://doi.org/10.26562/irjcs.2020.v0708.002
Citation: Advait (2020). An Analysis of Concepts and Techniques concerning the use of Hidden Markov Models for Sequential Data. IRJCS:: International Research Journal of Computer Science, Volume VII, 220-226.
https://doi.org/10.26562/irjcs.2020.v0708.002
Peer-review: Double-blind Peer-reviewed
Editor: Dr.A.Arul Lawrence Selvakumar, Chief Editor, IRJCS, AM Publications, India

Abstract: This A lot of machine learning concerns with creating statistical parameterized models of systems based on the data points that have been extracted from some underlying distribution depending upon the inductive bias. In probabilistic models, we model the joint probability of the input and the output or even the conditional probability of the output given the input (and vice versa using Bayes theorem).To elaborate, generative models determine the probability of the input given a particular output class if it is a classification task and discriminative models find the probability of the output class given the input. For a large class of models such as the standard ANNs, we assume the individual data points to be independent and identically distributed. However, when it comes to data such as rainfall measurements across successive days, there is an inherent sequential structure to the data which the previous assumption fails to take into consideration. Hidden Markov Models are used to model sequential data such as a time series where in for each sequence observed, the successive data points in that sequence are temporally related and are not independent, they are a part of a series of measurements and are evolving with respect to time-here, an independent variable. HMMs(Hidden Markov Models) are able to exploit this structure in the data and are used for a variety of tasks which are inherently sequential such as speech recognition, language modelling, time series etc. In this paper, we will understand and review the mathematics behind building and training an HMM, we will understand Latent Variables, Markov Chains, the forward backward algorithm and the Viterbi algorithm for HMMs. We will understand Lagrange Multipliers, the expectation maximization algorithm and see how it is applied for optimization in an HMM. We will also review some of the applications of an HMM. [3]
Keywords: Hidden Markov Models; Expectation maximization algorithm; Viterbi algorithm; Baum Welch Algorithm; Lagrange Multipliers;

I. INTRODUCTION

In the scientific community, there has been an observation that often probabilistic models are more reasonable than deterministic models.[6] Many phenomena that we observe are not just random phenomena to be described by Random Variables, but they are also evolving with time. A stochastic process is a family of Random Variables which are functions of time. A Markov Chain is a stochastic process with the Markov Assumption which is that the Random Variable at time step t depends only on the Random Variable at the previous time step and the probability distribution of the Random Variable at time step t given all the previous Random Variables reduces to the probability distribution of the Random Variable at time step t given the previous Random Variable at time step (t-1) and is independent of the Random Variables which precede that.[1] Now coming back to our sequence modelling problem, we can model the joint probability distribution of the input observations for a sequence as follows:

$$p(x_1, x_2, \dots x_N) = p(x_1) \prod_{n=2}^{N} p(x_n/(x_1, \dots, x_{n-1}))$$

As we said earlier, we have to relax the assumption about the observations being independent and identically distributed.
Under the Markov Assumption, this equation reduces to- $$p(x_1, x_2, \dots x_N) = p(x_1) \prod_{n=2}^{N} p(x_n / x_{n-1})$$

We assume that the conditional distribution of the Random Variables given the previous one does not change over time steps. This is equivalent to a stationary time series. Such a homogenous Markov Model simplifies the math and allows us to model a large number of sequences. If the Random Variable (for all time steps) takes on K states, then the conditional distribution of the Random Variable at one time step t given the Random Variable at the previous time step will have (K-1) free parameters for each of the K states of the Random Variable at time step (t-1). (K-1) is because we are modelling a probability distribution with the sum of the probabilities of the function (Random Variable) constrained to be 1. Thus, there will be K (K-1) number of free parameters. As we increase the number of terms as given in the conditional probability distribution, the number of free parameters will increase exponentially.

We must now understand latent variables. These are variables in our model, which are not directly observed but are inferred based on the values of other observable variables, which can be measured. Latent variables may represent an abstract concept or they may represent a part of physical reality. In latent variable models, the observed variable responses are assumed to be dependent on the latent variables. We can create sequence models which can generalize more than the Markov assumption and are consisting of a relatively limited number of parameters by introducing latent variables.

In the Hidden Markov Model, for each observation x at time step t, we introduce a latent variable z at time step t that is responsible for producing that observation. We make the hidden variables a Markov Chain. The distribution for each observed variable depends only on the latent variable at that time step and the corresponding conditional distribution determines the observed variable values. We introduce a matrix, A which is the transition matrix of the Markov Chain of the latent variables and a matrix B that denotes the emission probabilities of the observations given the latent variable. A(i,j) is the probability of the latent variable going from state i to state j from one time step to the next. B(j,k) is the probability of observing input k in state j where j is the state of the latent variable. In our homogeneous models, the emission distribution is the same across time steps. We have parameterized it using the matrix B, HMMs can have other emission distributions such as a mixture of Gaussians.[1] We let π denote the initial distribution of the states for the first time step.

Now that we have defined our model, we will demonstrate how to find the probability of the sequence, that probability is the likelihood estimate of the data given the parameters. This likelihood is also our objective function during optimization which we try to maximize. We will see how to find the most likely sequence of hidden states given an observation sequence, this is useful after our training is done.

II. FORWARD BACKWARD ALGORITHM

We now calculate the probability of an observation sequence. We marginalize the joint distribution of the observation sequence and latent variable states for all states the latent variables can take.

$$p(x) = \sum_{z_1=1}^{M} \dots \sum_{z_T=1}^{M} p(x_1, x_2, \dots x_T, z_1, z_2, \dots, z_T)$$

This reduces to,

$$p(x) = \sum_{z_1=1}^{M} \dots \sum_{z_T=1}^{M} p(x_1/z_1) \prod_{t=2}^{T} p(z_t/z_{t-1}) p(x_t/z_t)$$

For each time step, we are running a summation for each of the states for an inner function, which is a product, which increases linearly with t. Such a naïve implementation would be $O(M^T T)$. We can do better. We make the observation that there is a lot of shared computation in this expression as we traverse it and there is nested factoring.[4] Consider the expression again, we focus on the time step (t+1) –

$$p(x) = \cdots p(x_{t+1}/z_{t+1}) \sum_{z_t=1}^{M} p(z_{t+1}/z_t)(p(x_t/z_t) \sum_{z_{t-1=1}}^{M} \dots)$$

In the above equation, if we start from left to right, we see that the sub-expression/pattern we have highlighted is a function of x(t+1) and z(t+1).Let this be denoted as α. If we see the part in the brackets, this is the exact same pattern for x(t) and z(t). We have thus realized a recursive computation. What we mean to say is that for particular values of x(t+1) and z(t+1),we will have some corresponding α value. We thus define,

$$\alpha(t+1, z_{t+1}) = p(x_{t+1}/z_{t+1}) \sum_{z_t=1}^{M} p(z_{t+1}/z_t)\alpha(t, z_t)$$

Substituting the values based on the matrices we have defined, we get Initialization,

$$\alpha(1, z_i) = \pi_i B(z_i, x_i)$$

Induction,

$$\alpha(t+1, z_j) = B(j, x_{t+1}) \sum_{i=1}^{M} A(i,j)\,\alpha(t, z_i)$$

And finally,

$$p(x) = \sum_{i=1}^{M} \alpha(T, z_i)$$

This is a $O(TM^2)$ forward backward algorithm which is much better than the naïve approach. We take a look at what α represents. We know,

$$\alpha(1, z_1 = i) = p(x_1/z_1 = i)p(z_1 = i) = p(x_1, z_1 = i)$$

$$\alpha(2, z_2 = j) = p(x_2/z_2 = j) \sum_{i=1}^{M} A(i,j)\,\alpha(1, z_1 = i)$$

$$\alpha(2, z_2 = j) = p(x_2/z_2 = j) \sum_{i=1}^{M} p(z_2 = j/z_1 = i)\,p(x_1, z_1 = i)$$

$$\alpha(2, z_2 = j) = p(x_2/z_2 = j) \sum_{i=1}^{M} p(x_1, z_1 = i, z_2 = j)$$

Here we see that we are just marginalizing over z_1 values, we thus get,

$$\alpha(2, z_2 = j) = p(x_2/z_2 = j)p(x_1, z_2 = j)$$

Thus we see that,

$$\alpha(2, z_2 = j) = p(x_1, x_2, z_2 = j)$$

Similarly we can show,

$$\alpha(3, z_2 = j) = p(x_1, x_2, x_3, z_3 = j)$$

$$\alpha(t, z_t = j) = p(x_1, \dots, x_t, z_t = j)$$

At any time step the α value is the joint probability of the observations up until that time step and the value of the latent variable state at that time step.

There is a term β we need to define in a similar fashion for further uses which we will go through later on. The idea is similar, to break the computations into step wise abstractions and solve them recursively. We define β,

$$\beta(t+1, z_{t+1} = j) = p(x_{t+2}, \dots, x_T/z_{t+1} = j)$$

$$\beta(t+1, z_{t+1} = j) =$$

$$\sum_{z_{t+2}=1}^{M} p(z_{t+2}/z_{t+1} = j)\,p\left(\frac{x_{t+2}}{z_{t+2}}\right) \sum_{z_{t+3}=1}^{M} p(z_{t+3}/z_{t+2})p(x_{t+3}/z_{t+3}) \dots \sum_{z_T=1}^{M} p(z_T/z_{T-1})p(x_T/z_T)$$

Here, for each latent variable state at time steps ahead of (t+1), we can define a recursive computation C for each state of the latent variable in the following manner,

$$C(z_{t+2}) = p(x_{t+2}/z_{t+2}) \sum_{z_{t+3}=1}^{M} p(z_{t+3}/z_{t+2})C(z_{t+4})$$

$$C(z_T) = p(x_T/z_T)$$

$$\beta(t+1, z_{t+1} = j) = \sum_{z_{t+2}=1}^{M} p(z_{t+2}/z_{t+1} = j)C(z_{t+2})$$

III. VITERBI ALGORITHM

The next step is to find the most likely sequence of hidden states given an observation. This means we find a sequence that best explains the observations.

$$z' = \arg\max(p(z/x))$$

$$z' = \arg\max(p(z,x)/p(x))$$

Since we are finding out the z values, this is equivalent to,

$$z' = \arg\max(p(z,x))$$

We know,

$$p(z,x) = \pi_{z_1} B(z_1, x_1) \prod_{t=2}^{M} A(z_{t-1}, z_t) B(z_t, x_t)$$

Instead of looping through all possible sequences to find the one with the maximum probability, we use a dynamic programming based approach.[4] Let σ(t,j) be the maximum probability that we have visited an optimal sequence of states for (t-1) time steps and we have arrived at state j in time step t. σ(t,j) will give us the maximum probability of the sequence of latent variable states up until time step t wherein we are in state j at time step t. We hence have the recursive relation based on the principle of optimality as is with dynamic programming

$$\sigma(t,j) = \max_{i=\{1,\ldots M\}} \{\sigma(t-1,i)A(i,j)\} * B(j,x(t))$$

$$p^* = \max_{j=\{1,\ldots M\}} \sigma(T,j)$$

T is the final time step. Initially,

$$\sigma(1,j) = \pi_j B(j,x(1))$$

We use backtracking to find the best hidden state sequence. Let ϕ(t,j) tell us the optimal previous state at time step (t-1) given that we are in state j at time step t in an optimal sequence of states.

$$\phi(\mathrm{t},\mathrm{j}) = arg \max_{i=\{1,\ldots M\}} \{\sigma(t-1,i)A(i,j)\}$$

Initially,

$$\phi(1,\mathrm{j}) = 0$$

Let $z^*(t)$ tell us the optimal state at time step t. Therefore,

$$z^*(T) = \arg \max_{i=\{1,\ldots M\}} \sigma(T,i)$$

We can then iterate backwards and thus find the optimal sequence of states,

$$z^*(T-1) = \phi(\mathrm{T}, z^*(T))$$

We can continue this for all previous time steps.

IV. LAGRANGE MULTIPLIERS

Without loss of generality, we consider the constrained minimization problem. We have to minimize f(x) given that h(x)=0 where $x \in R^n$, f maps from R^n to R and h maps from R^n to R^m ,that is, there are m constraints. Here m is less than n. We consider the Jacobian matrix of the functions corresponding to the constraints, a point in n dimensional space x* is said to be a regular point if it satisfies all the constraints and the corresponding gradients are linearly independent.[5]

$$Dh(x^*) = \begin{bmatrix} Dh_1(x^*) \\ \cdot \\ \cdot \\ Dh_m(x^*) \end{bmatrix} = \begin{bmatrix} \nabla h_1(x^*)^T \\ \cdot \\ \cdot \\ \nabla h_m(x^*)^T \end{bmatrix}$$

Here each of the m gradients is n dimensional. The set of these constraints define a surface S which is a subset of R^n, assuming that the rank of the Jacobian matrix is m, we get the dimensionality of the dimensionality of the surface to be n-m. Consider a curve x(t) parameterized by t which represents some curve in the Surface S. We mean that for each value of t in some domain, we will get a corresponding x vector, which will be a function of t. For some constraint h(x)=0 satisfied on the Surface, we get h(x(t))=0. Since the h value is constant along the curve, we get the derivative of h with respect to t to be equal to 0. Using the chain rule we get,

$$h'(t) = D(h(x(t)) * D(x(t)) = \nabla \mathrm{h}(x)^T \mathrm{D}(\mathrm{x}(\mathrm{t})) = 0$$

We know from calculus that for such a parameterized curve, the derivative of the curve at a point is a tangent vector to the curve.[5] We thus get that the gradient of the equality constraint function is perpendicular to the tangent vector for a curve defined in the Surface of regular points. We consequently define the tangent space to be the set of all vectors which are perpendicular to the gradient vectors of the equality constraint function for a regular point x*.This is equivalent to the null space of the Jacobian matrix. We define the normal space corresponding to a regular point x* to be the space spanned by the gradient vectors of the equality constraint function. Indeed we see that the tangent space vectors are perpendicular to the normal space vectors. Now, let the point x* be a minimizer of f(x). We can write f(x) as f(x(t)) and set the derivative with respect to t to 0(as is the case for a minima) using chain rule.
We get,

$$\nabla \mathrm{f}(x^*)^T \mathrm{D}(x^*(\mathrm{t})) = 0$$

We thus see that the tangent vector of a parameterized curve in the Surface of the regular points which is corresponding to the derivative of x(t) with respect to t is perpendicular to both the gradient of f and the gradient of h. This is true for all equality constraint functions h, that is, both f and h gradient vectors are perpendicular to the same set of vectors(the vectors in the tangent space at x*). Therefore the gradient of f(x) lies in the same space as that spanned by the gradient vectors of the h(x) functions. As that gradient of f(x) lies in the normal space of x*(the point of the minima) defined by the gradient vectors of all h functions at that point, we can thus express the gradient of f(x) as a linear combination of the gradients of h.

We define the Lagrangian function as follows,

$$l(x,\lambda) = f(x) + \lambda^T h(x)$$

If we differentiate this with respect to x (taking partial derivatives for each of the dimensions in x) and equate to 0 we get the condition that we explained earlier about the gradient vector of being a linear combination of the gradients of h functions. Similarly, if we differentiate with respect to λ and equate to 0 we get the equality constraint conditions (h=0). Thus, we have m+n equations for m+n unknowns (m λ coefficients for the constraints and n dimensions in x). We can use Lagrange Multipliers in this manner to solve constrained optimization problems.

V. MAXIMUM LIKELIHOOD ESTIMATE

In applications such as Parts of Speech tagging in NLP, we essentially have the values of the Parts of Speech tags which act as hidden states. In such a scenario, we can use the maximum likelihood estimate wherein we find the parameters which maximize the conditional probability of observing the data given the parameters.

$$Parameters = arg\max_{\pi,A,B} p(X;\pi,A,B)$$

We have represented this probability earlier. Here we will do it similarly using the indicator function. Let we have N observation sequences, M hidden variable states, T time-steps and K observation values(x(t)). The likelihood becomes,

$$L = \prod_{n=1}^{N}\{\prod_{i=1}^{M}\prod_{i=1}^{K}\pi_i^{1(z_1=i)}B(i,k)^{1(z_1=i,x_1=k)}\prod_{t=2}^{T}\prod_{i=1}^{M}\prod_{j=1}^{M}\prod_{k=1}^{K}A(i,j)^{1(z_{t-1}=i,z_t=j)}B(j,k)^{1(z_t=j,x_t=k)}\}$$

It is easier for us to work with the logarithm of the likelihood and since the log is a monotonically increasing function, maximizing the likelihood is equivalent to maximizing the log likelihood. The products turn into summations as is with logarithms and we have the equality constraints for π values summing upto 1 and each row of A and B summing upto 1. This problem is solved with Lagrange multipliers and it turns out that we get the A(j,k) values to be the count of the state transitions from j to k in our data divided by the count of the state being j. We get the same frequency counts for B and π.

VI. EXPECTATION MAXIMIZATION ALGORITHM

Here we describe the general EM algorithm, which is used in the case of latent variables in optimization problems. Consider the general log likelihood estimate from earlier. For a scenario where we have n observations,

$$l(\theta) = \sum_{i=1}^{N} log\ (p(x_i;\theta))$$

This is the same as marginalizing the joint distribution over z,

$$l(\theta) = \sum_{i=1}^{N} \log(\sum_{z_i=1}^{M} p(x_i,z_i;\theta))$$

The summation over z_i values denotes all the M different values that z_i can take. The numbers 1 to M denote the indices of the values which z_i takes. We see that we have a summation inside the logarithm as we have to sum over the latent variables and we do not have a computationally feasible closed form solution for the log likelihood. What we do in the EM algorithm is that we find the tight lower bound to the log likelihood in the parameter space and then maximize that lower bound with respect to the parameters to obtain a new set of parameters. We then again find the tight lower bound using these new parameters and then optimize that lower bound to obtain another new set of parameters. We repeat this procedure until convergence. [1] Consider Q to be an arbitrary probability distribution over z. Multiplying and dividing by $Q(z_i)$ inside the log likelihood we get,

$$l(\theta) = \sum_{i=1}^{N} \log \sum_{z_i=1}^{M} Q_i(z_i) * \left(\frac{p(x_i, z_i; \theta)}{Q_i(z_i)}\right)$$

We take a close look at the term inside the logarithm. Observe that it is an expectation with respect to the distribution $Q(z_i)$ for the random variable z_i. The term being multiplied to $Q(z_i)$-the probability value is a function of the random variable z for a given x and θ and can itself be viewed as a Random Variable. We recollect the law of the Unconscious Statistician and take the expectation. Now, we recollect Jensen's inequality for a concave function and notice that the log is a concave function.

$$\log\big(E(X)\big) \geq E(\log(X))$$

Using this inequality in our log likelihood, we get,

$$l(\theta) \geq \sum_{i=1}^{N} \sum_{z_i=1}^{M} Q_i(z_i) * log\left(\frac{p(x_i, z_i; \theta)}{Q_i(z_i)}\right)$$

In the Jensen's in equality, we achieve the equality when the random variable under consideration is a constant. Let that constant be C.

$$\frac{p(x_i, z_i; \theta)}{Q_i(z_i)} = C$$

$$Q_i(z_i) = \frac{1}{C} * p(x_i, z_i; \theta)$$

Marginalizing over z,

$$\sum_{z_i=1}^{M} Q_i(z_i) = \frac{1}{C} * \sum_{,z_i=1}^{M} p(x_i, z_i; \theta)$$

$$C = p(x_i; \theta)$$

Therefore,

$$Q_i(z_i) = \frac{p(x_i, z_i; \theta)}{p(x_i; \theta)} = p(z_i / x_i; \theta)$$

In the E step, we estimate the distribution Q as the posterior probability of z given x for a set of parameters in order to obtain the tight lower bound of the likelihood function. In M step, for estimated distribution Q based on a set of parameters, we maximize the lower bound that we have obtained with respect to the parameters to obtain a new set of parameters for the new time step. This optimization is easier to perform as compared to the original log likelihood function.

$$\theta = arg \max_{\theta} \sum_{i=1}^{N} \sum_{z_i=1}^{M} Q_i(z_i) * log\left(\frac{p(x_i, z_i; \theta)}{Q_i(z_i)}\right)$$

We again find the tight lower bound based on these new parameters and repeat the same procedure as be said before, until convergence. It can see seen that the likelihood values after each iteration are indeed increasing.

VII. BAUM WELCH ALGORITHM

The Baum Welch Algorithm is the EM algorithm applied to HMMs. We will see the mechanism of this algorithm and go through some intuition for it. We first need to define a few important terms. We have defined the following earlier along with the matrices A and B,

$$\alpha(t, z_t = j) = p(x_1, \dots, x_t, z_t = j)$$
$$\beta(t+1, z_{t+1} = j) = p(x_{t+2}, \dots, x_T / z_{t+1} = j)$$

We know define, we know that posterior probabilities for the latent variables are used in the EM algorithm.

$$\zeta_t(i, j) = p(z_t = i, z_{t+1} = j / x)$$
$$\zeta_t(i, j) = p(z_t = i, z_{t+1} = j, x) / p(x)$$

The denominator can be obtained by marginalizing over all z_t, z_{t+1}values. We take a close look at the numerator and see that it can be defined as follows using the rules of conditional probability,

$$\zeta_t(i, j) = p(x_1, \dots, x_t, z_t = i) * p(z_{t+1} = j / z_t = i) * p(x_{t+1} / z_{t+1} = j) * p(x_{t+2}, \dots, x_T / z_{t+1} = j)$$
$$Numerator = \alpha(t, z_t = i) * A(i, j) * B(j, x_{t+1}) * \beta(t+1, z_{t+1} = j)$$

$$\zeta_t(i,j) = Numerator / \sum_{i=1}^{M}\sum_{j=1}^{M} Numerator$$

We have described in the forward backward algorithm how to calculate α and β values. We also define,

$$\gamma_t(i) = p(z_t = i/x) = \sum_{j=1}^{M} \zeta_t(i,j)$$

In the E step here, we find ζ and γ for all time steps based on the values of π, A and B at iteration. We initialize arbitrary values of π, A and B for the first iteration based on prior knowledge if any. In the M step, we find the values of π, A and B based on the posterior probabilities, which we have found out, as is the case in EM algorithm applications.

$$A(i,j) = \frac{\sum_{t=1}^{T-1} \zeta_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$B(j,k) = \left(\sum_{t=1}^{T-1} \gamma_t(j) * 1(x_t = k)\right) / \sum_{t=1}^{T-1} \gamma_t(i)$$

$$\pi_i = \gamma_1(i)$$

We get this result as expand the lower bound of the log likelihood based on Q(z) as we did in the EM algorithm, then substituting the probability values based on our parameters. Solving the constrained optimization problem for π, A and B using Lagrange multipliers and substituting the value of Q(z) as p(z/x) in our equations, as we did in the EM algorithm.

VIII. APPLICATIONS AND CONCLUSION

The Baum Welch algorithm wherein we calculated the posterior probabilities for the latent variables given an observation sequence can be generalized to n sequences.[1] HMMs used in this manner are powerful models for sequential data and are still relevant as a framework for unsupervised machine learning. HMMs are used in automatic speech recognition where we extract spectral features from the audio data after sampling and quantization[2]. The audio data consists of the sound pressure levels measured across time. The key is that the Fourier transform[2] gives us the measures of various frequency components in the audio wave and each phone ,which is a basic unit of speech, has a distinct frequency pattern based on how the sound is produced through the vibration of the vocal cords and the configuration of the vocal tract.[2] HMMs are also used in language models as in parts of speech tagging as we described earlier and in text classification. In text classification, we train an HMM for each class of the text based on our training data after using NLP concepts like stemming and lemmatization to prepare the data. For a test sentence, we find the log likelihood of that sentence for each HMM and we return the class whose HMM gave us the highest log likelihood. We can also include the prior probabilities for each class. Along with all this, HMMs are used in tasks such as online character recognition. With the advancement of deep learning and recurrent neural networks, HMMs are used along with LSTMs for creating hybrid sequence models. With this, we conclude our study of Hidden Markov Models. In this paper, we understood what an HMM is, how an HMM works, we understood the underlying concepts behind an HMM and how it is used for various sequence modelling applications. Thank you.

REFERENCES

1. Christopher Bishop, Pattern Recognition and Machine Learning, Springer, 2006.
2. Daniel Jurafsky and James H.Martin ,Speech and Language Processing, Pearson Education, 2009.
3. Pietrzykowski, Marcin & Sałabun, Wojciech. (2014). Applications of Hidden Markov Model: state-of-the-art.
4. L. R.Rabiner, B. H. Juang .IEEE ASSP magazine.(1986).An Introduction to Hidden Markov Models.
5. Edwin Chang and Stanislaw Zak, An introduction to optimization, Wiley, 2013.
6. J Medhi, Stochastic Processes, New Age International Publishers, 2017.

