Modeling and Learning for Sequential data

Leena Chennuru Vankadara

***Abstract*— Statistical learning methods for sequential data are different from traditional machine learning algorithms since the i.i.d assumption is not valid for sequential data. These algorithms need memory and the ability to represent distributions over sequential spaces. In this paper we discuss two different approaches to represent sequential data, Markovian models and Recurrent Neural networks. We discuss the theoretical limitations of each representation and a few extensions/solutions are discussed to mitigate the limitations.**

I. INTRODUCTION

Statistical machine learning in many applications such as time series prediction, structured prediction in Computer Vision or Automated speech recognition involves working with sequential data. Most Machine learning algorithms however invoke the assumption that the data is independent and identically distributed(i.i.d). This assumption is clearly not valid for sequential data where temporally close data points have a strong dependence between each other. For e.g, to predict whether or not it is going to rain today, using i.i.d assumption would imply that the prediction would always remain the same depending on the relative frequency of outcomes in the past. Consider the following example in figure 1. A single data point in the sequence of points which represent the letter "h" cannot be classified into any category without any information about its context.

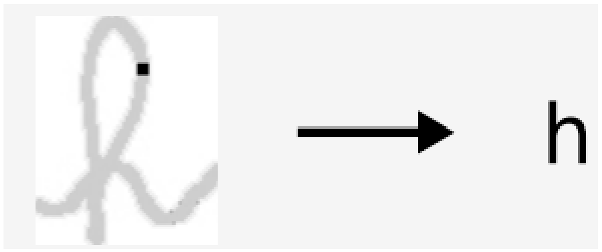

Fig. 1. Importance of contextual information as shown in recognition of the written letter *h*[5].

A single data point of the sequence can only be significant in the context of the neighboring data points. as it can not capture any meaning in itself. However this does not always need to be the case. In computer vision applications, a single data point which is an image can contain meaningful information which is of significance. However by making use of context the information can be augmented to extract additional patterns. To capture the dependencies between temporally close points, different statistical models that do not need to invoke the i.i.d assumption need to be employed. The problem of sequence learning can be formulated in the following way.

Definition: Given a training set consisting of n sequences $s_1, s_2, ..., s_n$ sampled from a population, learn a model such that it minimizes or maximizes some measure of quality of the model on new sequences and have been drawn from the same population as the training set.

In this paper, we discuss the problem of modeling sequential data using Markovian models with both discrete and continuous distributions. In addition, we discuss generative and discriminative approaches to modeling with HMMs. Then we present algorithms for learning and inference for traditional HMM's and some limitations of HMM's. we then discuss some proposed extensions and variant architectures to mitigate these problems. Then we motivate the problem of sequential learning with an example of its application to speech recognition. In addition, we discuss solutions to modeling sequential data using Neural Network approaches.

II. MARKOVIAN MODELS FOR SEQUENTIAL DATA

A k^{th} order Markov chain is a sequence of random variables $x_1, x_2, ..., x_n$ satisfying the conditional independence property as shown in equation 1. A graphical representation of a second order markov chain can be seen in figure 2.

$$P(x_i|x_{i-1}, ..., x_1) = P(x_i|x_{i-1}, ..., x_{i-k}) \tag{1}$$

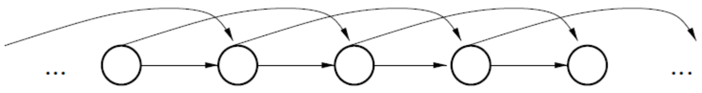

Fig. 2. Graphical Representation of Markov Model of order 2, where the current value is dependent on the previous 2 values[25].

A Markovian model[1] of order K represents the probability distribution over a k^{th} order Markov chain. Recall that the joint probability distribution of any set of variables[1] $x_1, x_2, ..., x_n$ can be factorized as in equation 2:

$$P(x_1, x_2, ..., x_n) = P(x_1)P(x_2|x_1), ..., P(x_n|x_{n-1}.., x_1) \tag{2}$$

Since the i.i.d assumption is relaxed in a k^{th} order Markov chain, the joint probability distribution of a Markov model of order k(according to equation 1) over $x_1, ..., x_n$ can be factorized as in equation 3:

$$P(x_1, x_2, ..., x_n) = P(x_1)\prod_{i=2}^{n} P(x_i|x_{i-1}) \tag{3}$$

[1]Note that the term variables in the paper refers to random variables unless specified otherwise.

The value k, which is the order of the Markov model, intuitively represents the capacity of the memory of the model. With $k = 0$, we have a memory-less model which is equivalent to the i.i.d assumption over the sequence of variables. With $k = 1$, we have a first order Markov chain where each variable on the sequence is only dependent on the previous variable of the sequence. For instance, in order to predict if it would rain today, the only variable that needs to be taken into consideration would be if it had rained the previous day. The preceding variable referred to as the ***"state variable"*** in literature hence encapsulates the summary of all the variables that occurred in the past. Note that from equation 3 a Markov model of order 1 can be completely specified by $P(x_1)$ which is referred to as the ***"Initial State Probabilities"*** and by $\forall i P(x_i|x_{i-1})$ which are referred to as the ***"Transition state probabilities"***[1].

Markov models can be used to model low range contexts. However as the size of the context increases, the number of parameters that need to be specified in order to represent the joint probability distribution of all the variables becomes exponentially large.

Hidden Markov Models(HMM) employ a hidden variable to concisely summarize the variables occurred in the past and thereby reducing the number of parameters needed to specify the joint distribution by a significant amount. HMMs do not assume that the sequence of observed variables form a Markov chain but assume that the hidden variables form a Markov chain. Traditional HMMs assume that the hidden variables in a HMM form a Markov chain of order 1 since by increasing the number of values the hidden variable can take, a HMM of order 1 can represent a HMM of any higher order. Also note that a HMM is simply a dynamical extension of a Gaussian Mixture model, which in the limit of infinite mixture components is an Universal approximator of densities. Intuitively the hidden state variable of a Gaussian Mixture model can be thought of as the value of the parameter that generates the mixture component generated by that parameter.

A. Hidden Markov Models

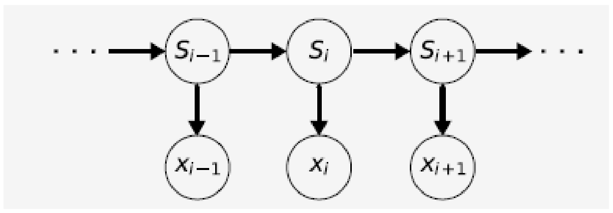

Fig. 3. Graphical representation of a Hidden Markov Model, where the hidden variables form a Markov Chain[25].

A Hidden Markov Model is a directed acyclic graphical model(A Bayesian Network[2]) whose conditional independence properties are depicted in Figure 3. Recall that in a Bayesian network, the joint distribution of the sequence of variables $s_1, ..., s_n$ can be factorized as shown in equation 4, Where $Pa(s_i)$ represents the parents(variables from which there is a directed edge to s_i) of the variable s_i.

$$P(s_1, s_2, ..., s_n) = \prod_{i=1}^{n} P(s_i|Pa(s_i)) \tag{4}$$

From equation 4, we can represent the joint distribution of the variables $x_1, .., x_n, s_1, ..., s_n$ of a HMM as shown in equation 5 where x_i's are the observed variables and s_i's are the hidden variables.

$$P(x_1, .., x_n, s_1, ..., s_n) = P(s_1) \prod_{i=2}^{n} P(s_i|s_{i-1}) \prod_{i=1}^{n} P(x_i|s_i) \tag{5}$$

From equation 5 it can be seen that the joint distribution of the variables represented by a HMM can be completely specified by $P(s_1)$ called the ***"Initial state probabilities"***, $P(s_i|s_{i-1}), \forall i$ called the ***"Transition probabilities"*** and $P(x_i|s_i), \forall i$ which are called the ***"Emission probabilities"***. Homogeneous HMMs have the property that the Transition and the Emission probabilities are independent of *i*. Traditional Hidden Markov Models have a discrete state space. This means that the transition probabilities form a discrete distribution.

The topology of a HMM is defined by its state transition diagram. A state transition diagram is a directed graph where the nodes represent different values that the state variable can take(the state space, S). Each edge in the graph represents the probability of a transition of variable from one state value to the other. An example of the state transition diagram is shown in the figure 4 where k is the hidden state variable which can take values $1, 2, 3$ and A_{ij} represents the transition probability $P(k_t = i|k_{t-1} = j)$. The number of non-zero transitions in a state transition diagram govern the topology of the HMM. It is one of the most important design choices of a HMM since both learning and inference in a HMM depend on the number of non-zero transition probabilities. For instance traditionally in speech recognition literature, a left to right topology is used as depicted in figure 5.

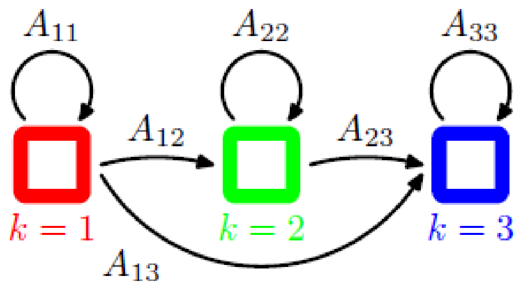

Fig. 4. State Transition Diagram consisting of three states[27].

The initial state probabilities, transition and emission probabilities uniquely define a HMM. These parameters need to be learned. Note that a HMM is a Bayesian Network(directed acyclic graphical model) and any general algorithm that can be used for learning in Bayesian networks can be applied to Hidden Markov Models.

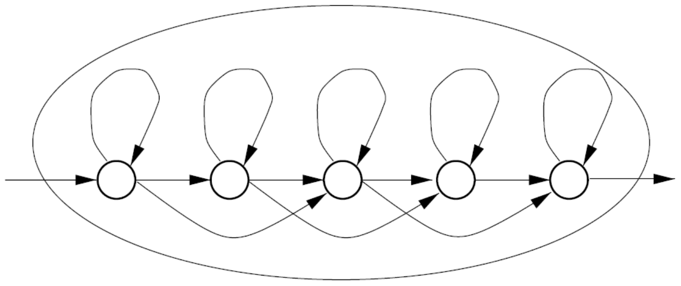
Fig. 5. Left to Right topology of an HMM[27].

B. Learning in HMMs

As it is for a Bayesian Network, there are broadly two approaches to learning in HMM's namely the frequentist approach and the Bayesian approach. Bayesian approach to learning requires that we have some prior knowledge about the model. In an Hidden Markov Model, the set of conditional independence properties and the parameters that are defined by the structure of the HMM can be represented as a prior distribution over the parameters and the model. As discussed above, several different algorithms can be used for learning such as Expectation Maximization(EM)[8], Viterbi Learning Algorithm[9] or Markov Chain Monte Carlo methods. We briefly present one of these learning algorithms used for training in the context of HMMs.

EM which is a Maximum Likelihood estimator(or Maximum a posteriori) is however a partly non-bayesian approach to learning since along with generating the joint distribution over the latent and the observed variables it generates a point estimate of the parameters. However by treating the parameters as latent variables the algorithm can be interpreted as fully Bayesian.

EM algorithm computes the Expectation of the log likelihood over all the values that the hidden state variables can take. It attempts to maximize the marginal likelihood of the observed data by iteratively performing two different steps referred to as the E-step and the M-step.

E-Step: This step involves computing an auxiliary function as shown in equation 6. E_S is the expectation over all values of the hidden state variables and the auxiliary function is conditioned on the previous parameter estimates.

$$F(\theta|\theta^k) = E_S[logP(X,S|\theta)|X,S,\theta^k] \tag{6}$$

In order to compute this auxiliary function we first introduce $z_{i,t}$ as indicator variables defined as shown in equation 7.

$$Z_{i,t} = \{1 \ if \ s_t \ = \ i \ and \ 0 \ otherwise\} \tag{7}$$

By reformulating $logP(X,S|\theta)$ in 6 by using the indicator variables as defined in equation 7 we have equation II-B as shown below.

$$F(\theta|\theta^k) = \sum_{p,t,i} E_S[z_{i,t}|x_1^T,\theta^k]logP(x_t|s_t = i,\theta) + \sum_{p,t,i,j} E_S[z_{i,t},z_{j,t-1}|x_1^T,\theta^k]logP(s_t = i|s_{t-1} = j,\theta)$$

From II-B note that the auxiliary function is simply a weighted sum of the likelihood of the emission and the transition distributions. The weights of each term can be rewritten as shown in the equation 8 and 9 and the weights are simply the state posterior probabilities and the transition posterior probabilities:

$$P(s_t = i|x_1^T,\theta^k) = E_S[z_{i,t}|x_1^T,\theta^k] \tag{8}$$

$$P(s_t = i, s_{t-1} = j|x_1^T,\theta^k) = E_S[z_{i,t},z_{j,t-1}|x_1^T,\theta^k] \tag{9}$$

We use the "Forward and the Backward recursions" to compute the posteriors(the weights) as described in[10][11][12].

"Forward Recursion": It constitutes computing $P(x_1,..,x_t,s_t)$ as shown in equations 10 and 12 by exploiting the prior we have over the Model structure.

$$P(x_1,..,x_t,s_t) = P(x_t|x_1,..,x_{t-1},s_t)P(x_1,..,x_{t-1},s_t) \tag{10}$$

Applying the conditional independence properties of a HMM on equation 10 we have equation 11 which is simplified to obtain equation 12.

$$P(x_1,..,x_t,s_t) = P(x_t|s_t)\sum_{s_{t-1}} P(x_1,..,x_{t-1},s_{t-1},s_t) \tag{11}$$

$$= P(x_t|s_t)\textstyle\sum_{s_{t-1}} P(s_t|s_{t-1},x_1,.x_{t-1})P(x_1,.x_{t-1},s_{t-1})$$

$$P(x_1..x_t,s_t) = P(x_t|s_t)\sum_{s_{t-1}} P(s_t|s_{t-1})P(x_1..x_{t-1},s_{t-1}) \tag{12}$$

Backward Recursion: This constitutes computing the term $P(x_{t+1},..,x_T|s_t)$ in a similar fashion in equation 13.

$$P(x_{t+1}^T|s_t) = \sum_{s_{t+1}} P(x_{t+1}|s_{t+1})P(s_{t+1}|s_t)P(x_{t+2}^T|s_{t+1}) \tag{13}$$

The state posterior can be computed from the terms $P(x_1,..,x_t,s_t)$ and $P(x_{t+1},..,x_T|s_t)$. as shown in 14.

$$P(s_t|x_1^T) = \frac{P(x_1^t,s_t)P(x_{t+1}^T|s_t)}{P(x_1^T)} \tag{14}$$

The transition posterior can be computed as shown in equation 15.

$$P(s_t,s_{t-1}|x_1^T) = \frac{P(x_t|s_t)P(x_1^{t-1},s_{t-1})P(x_{t+1}^T|s_t)P(s_t|s_{t-1})}{P(x_1^T)} \tag{15}$$

M-Step: This step identifies the optimal parameters that maximize the auxiliary function which has been computed in the E-step. The algorithm used to update the parameters depend on the type of the distributions used.

Once the parameters are learned, there are two questions that can be asked. The first question is given an observation sequence, what is the likelihood of the sequence given the parameters$P(x_1, .., x_t|\theta)$ (Problem of Evaluation). Since a HMM is a generative model, it represents the data generation process. The second question would be, given an observation sequence, what is the most likely state sequence corresponding to the observed sequence$P(s_1, .., s_t|x_1, .., x_t, \theta)$(Problem of Inference). For instance, in speech recognition, the question would be given a set of phonemes what is the most likely sequence of words corresponding to the sequence of phonemes.

Evaluation: Once the parameters of a HMM are known, the joint distribution $P(x_1, ..x_T, s_1, ..s_T)$ can be computed in linear time O(T). However to compute the likelihood of a given observed sequence the simply need to consider the marginalized distribution over $s_1, ..s_T$ which contain an exponential number of terms. In order to compute the likelihood, the ***"Forward Phase"*** algorithm is used to compute $P(x_1, .., x_T, s_T)$ in a recursive fashion and then marginalize it over s_T. The Forward phase algorithm as discussed above is a recursion as shown in the equation 16.

$$P(x_1..x_t, s_t) = P(x_t|s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}) P(x_1..x_{t-1}, s_{t-1}) \tag{16}$$

Initializing this recursion with the initial state probabilities we can evaluate the likelihood of the observed sequence.

C. Inference in HMMs

Once the parameters of a HMM are learned several questions could be answered and one of the most important ones is to infer the most likely state sequence given an observed sequence. There are several methods of exact and approximate inference in HMM. We discuss the Viterbi Algorithm[9] for inference in this paper. The goal of the algorithm in the context of a HMM is formulated in the equation 17.

$$(s_1, .., s_T)^* = argmax_{s_1,...,s_T} P(s_1, .., s_T, x_1, ..., x_T) \tag{17}$$

The algorithm first defines a function $V(i, t)$ as follows and uses Bellman's dynamic programming to derive a recursive formulation as shown in equations 18 and 19.

$$V(i, t) = max_{s_1,..s_{t-1}} P(x_1, .., x_t, s_1, ..., s_{t-1}, s_t = i) \tag{18}$$

$$V(i, t) = P(x_t|s_t = i) max_j P(s_t = i|s_{t-1} = j) V(j, t-1) \tag{19}$$

This recursion is solved using the initialization as shown in the equation 20.

$$V(i, 1) = max_{s_1} P(x_1|s_1) P(s_1) \tag{20}$$

Note that once this recursion is solved the solution to the original problem as described in the equation 20 is simply as shown in the equation 21. The optimal state sequence can be calculated through a backward recursion after identifying the optimal i^*.

$$argmax_i V(i, T) = (s_1, .., s_T)^* \tag{21}$$

The computational time taken to compute the optimal state sequence is directly proportional to the number of non-zero transitions in the state transition diagram which is the $O(|s|^2)$ where $|s|$ is the number of states in the state transition diagram.

D. Variant Architectures

"Linear Dynamical System":

The transition probability distributions of a HMM are traditionally discrete. However, the emission distributions can be discrete or continuous. Linear Dynamical systems[7] or state space models have continuous transition distributions. The Bayesian network corresponding to the Linear dynamical system is identical that of the HMM. However, the transition distributions of a Linear dynamical system are continuous. For simplicity, the transition distributions follow a Gaussian with the mean as a linear function of the previous state as shown in equation 22.

$$P(s_t|s_{t-1}, x_t) = N(s_t; As_{t-1} + Bx_t, \sum(s_{t-1}, x_t)) \tag{22}$$

"Input Output HMMs": Traditional HMM is a generative model as it maximizes the likelihood of the observed data and presents a joint distribution of the observed and the hidden variables. It can be turned into a discriminative model through inferring the probability of the most likely state sequence given an observed sequence. However, the number and the meaning of states have to be pre-defined in a deterministic fashion for a traditional HMM. However, in order to make the HMM a discriminative model in a natural way, we can employ Input-Output HMMs[6]. Figure 6 shows the Bayesian network corresponding to an Input-Output HMM.

Note that the Input Output HMM is an in-homogeneous model as that the transition and the emission distributions depend on time as they depend on the Input variable x_t which is time varying. Also note that the network represented in the figure above is a synchronous IOHMM and the size of the input and the output sequence is the same.

E. Limitations of a HMM

As mentioned above, a HMM is a dynamical extension of a Gaussian Mixture Model which in the limit of infinite mixture components can model any distribution. However as the number of states increases, the computational time increases by an exponential manifold. As the number of states increase both representation as well as interpretation of the model

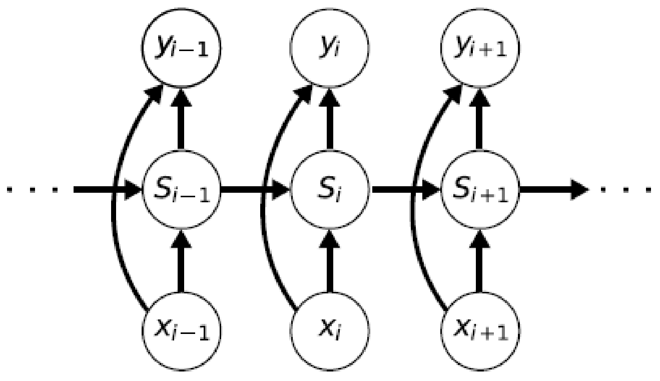

Fig. 6. Input Output HMM[6].

becomes increasingly difficult. A linear dynamical system is limited by the linearity of its dynamics. By extending the HMM these problems could be mitigated. Some of these extensions are described below.

F. Extensions of a HMM

"Factorial HMMs": Factorial HMMs[13] use several levels of hidden states and hence uses a distributed hidden state representation to model the sequence of observed variables. The Bayesian Network corresponding to a Factorial HMM is depicted in the figure 7. Note that each state variable is decoupled from the other variables and evolves its dynamics independently.

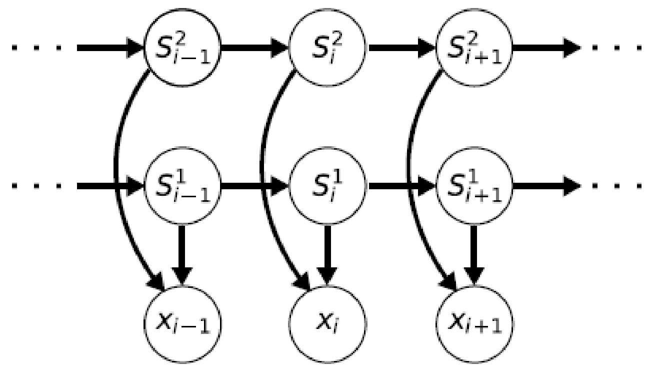

Fig. 7. Factorial HMM[13]

"Switching state space Models": The figure 8 depicts the Bayesian network corresponding to the Switching state space model[16]. Recall that in a Linear dynamical system, the transition distributions are continuous. However, the dynamics of the Linear dynamical system(LDS) is linear as indicated by the name. In order to enable non-linear dynamics in a LDS, a switching variable is introduced.

In a switching state space model, the sequence of observations $y_1, ..., y_T$ are modeled by M sequences of real valued hidden state variables x^i_t and a sequence of discrete valued hidden state variables s_t are used where s_t follows a multinomial distribution with M values say $1, 2, ..., M$. This model can be thought of M different state space models and conditioned on the switching variable assuming a value $i \in \{1, 2, .., M\}$ the switching state space model assumes the form of a state space model specified by the i.

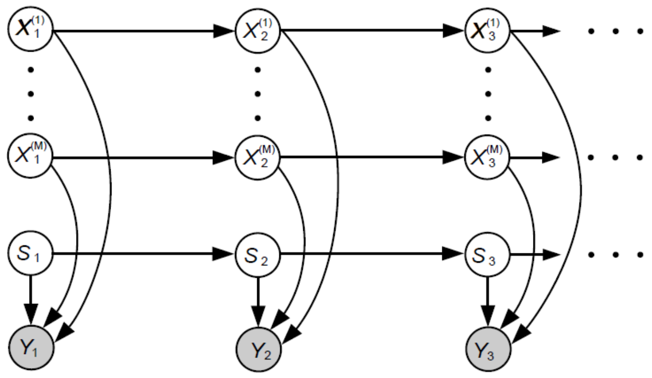

Fig. 8. Switching state space models[16]

III. Speech Recognition

Speech Recognition[4] is a very well known application of modeling sequences using HMMs. There are three important problems that have to be considered here:

- Preprocessing of speech signals: This defines the type of feature used in speech audio
- Acoustic Models: These model define how a word is pronounced and varies with noise, region and language.
- Language Models: This is to impose a basic grammatical structure on the vocabulary. This is specific to a language and the grammatical rules can be restricted to the subset of types of sentences used.

A. Preprocessing

Speech signals cannot be directly used to train HMMs. Preprocessing the time-signal is a fundamental step for speech recognition. There are some standard features that have been used in Automatic Speech Recognition Systems(ASRs). Some of the widely used features are Mel-cepstrum, Linear prediction coding(LPC), energy signals, delta and acceleration coefficients.

B. Acoustic Models

For word recognition or phoneme recognition, building acoustic models is essential. Acoustic Models are basically pronunciation of words and can be trained using phones, phonemes or words. Phonemes are the basic units of speech utterances and phones are the basic set of articulation that are required to product a phonemes. In order to design acoustic models, factors like environmental noise, inter speaker variation and intra-speaker variation should be taken into consideration for task specification.

There are two approaches to solving the problem of modeling pronunciations: phoneme recognition and word recognition. In phoneme recognition, one HMM is trained for each phoneme. This means that the observed states are the features computed on the signal and the hidden state sequence is the sequence of phones that produce that phoneme. Training in phoneme recognition is basically searching for the optimal parameters of the HMM given the observations(features). In recognition or inference, we compute the sequence of states that correspond to the sequence of observed states. An alternate approach for modeling pronunciation is to perform word recognition. In this approach, the features computed on the signal are observed states and the sequences of phonemes that pronounce the word are the hidden states. A graphical representation of the word level HMM, which is trained for the word *the* can be seen in figure 9.

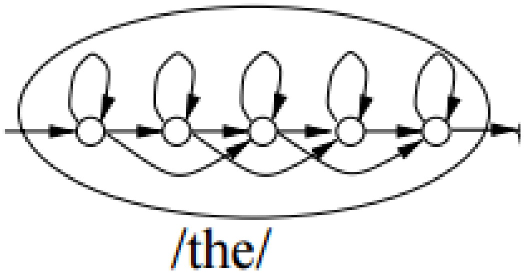

Fig. 9. A word level HMM for the word *'the'*[3].

In order to model speech data using HMM, deciding the number of states for each word or phoneme is a major decision. In phoneme level recognition, one HMM is trained for each phoneme and the number of states for each model is taken to be 3. On the other hand, there are two ways to decide the number of states in word level recognition. The first approach is that number of states for word recognition HMM should correspond to the number of phonemes that is required to produce the word. This translates to roughly 2-10 states per word. An alternate approach is that number of states should be the average number of observations in spoken word.

C. Language Models

Recognizing a phoneme in isolation is neither effective not is it useful. This is because when someone speaks, they speak in continuity. We have already seen in acoustic models, there are two ways to model pronunciation but for doing speech recognition of sentences or continuous utterances, language models are used. The two objectives of using these models are reduce the search space in speech recognition, robustness to noise.

A simple language model that can be used is a finite state machine(FSM). Here transitions between current and next word are defined by the FSM. Since FSM is not a good model for human language and it is deterministic, sequences of utterances can also be modeled using HMMs. The two variants of HMMs used for language models are word and word-class based. Both types of HMMs can be defined for different context lengths. For example, a language model in which the current word or word class is dependent on the previous two words is called a Bi-gram model HMM.

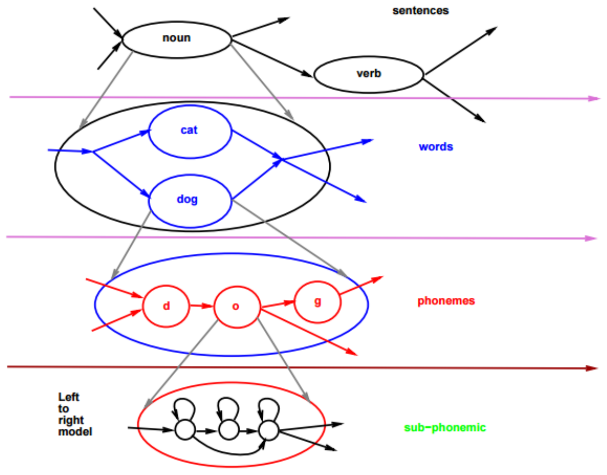

Fig. 10. A hierarchy of HMMs used for speech recognition[3].

Speech recognition using HMMs is a very well studied topic. The ultimate aim of this is to recognize whole lengths of utterances. To achieve this hierarchical HMMs are constructed as seen in figure 10. The most basic level HMMs are to recognize sub-phonemes, then there is a level to recognize different phonemes. Then, the transitions between different phonemes is defined by the word level HMMs. The transitions between different words in a sentence are defined by the language models.

IV. Neural Networks for Sequence Learning

In the previous sections, we discussed the structural features of sequential data and we discussed a class of models called HMMs, which are very successful in modeling sequences. However, there is an alternate class of models called neural networks[17] to understand and infer about sequences. There are two types of neural networks: feedforward neural networks and recurrent neural networks.

A. Feedforward Neural Networks(FNN)

FNNs are a Multilayer perceptrons that do not contain cyclic(Feedback) connections between the neurons. The architecture of the neural networks is defined by an input layer, a set of hidden layers, and an output layer. Since we already discussed that an individual data that arises from a sequential process is meaningful in its context, FNNs model the sequence by a context window on the sequence. This is done by pre-defining the size of the context window and the truncated signal is given as an input into the network with a delay. These networks are referred to as the time delay networks(TDNN) in literature[18]. An example of a TDNN for is shown in figure 11. Learning in these networks is performed through Backpropagation[23].

There are several limitations to this approach. One problem is that we have an additional parameter which is the size of the window and determining this based on the training set will lead to poor generalization. The second most important problem is that these networks don't account for distortions in sequences and are not invariant to time shifts. MLP's are not naturally equipped to handle sequential data and hence Recurrent Neural Networks, an extension of MLPs are used to deal with sequential data.

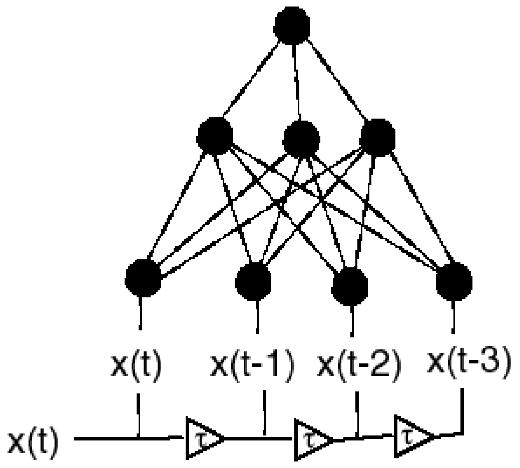

Fig. 11. Time Delay Neural Network[14]

B. Simple Recurrent Neural Network

A fundamental feature of a Recurrent Network is the existence of at least one feedback connection. A simple recurrent network is a three layered perceptron with the activations of the hidden units or output units at the previous time step feeding back into the network along with the input(as shown in figure 12). A context layer(or a delay layer) is introduced to hold the activations in the network for one time step. Jordan[19] and Elman[20] networks are different variations of a Simple Recurrent Network. The difference in their networks is demonstrated in the figure 12. Since these networks can only hold one information from one time step before the network can not capture dependencies for any larger contexts. Learning in Simple Recurrent Networks can be performed through Backpropagation through time[24] which is a variation of Backpropagation performed on the network unfolded in time.

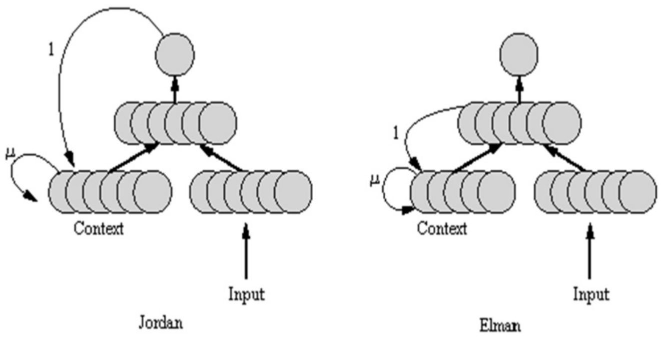

Fig. 12. In Jordan networks, the output units are fed back into the network and in the Elman Network, the hidden units are fed back into the network[15].

C. Recurrent Neural Networks

A Recurrent Neural Networks(RNNs) feature cyclic(or Feedback) connections as described before. RNNs are a natural extension of Multilayer perceptrons(MLPs) to deal with sequential data. This is demonstrated by figure 13. They map a space of sequential data to the space of outputs which could also be sequential in the case of sequence to sequence mapping. RNNs with sufficient number of hidden units can universally approximate any sequence mapping. Training in RNNs is also done using Backpropagation through time(BPTT)[24].

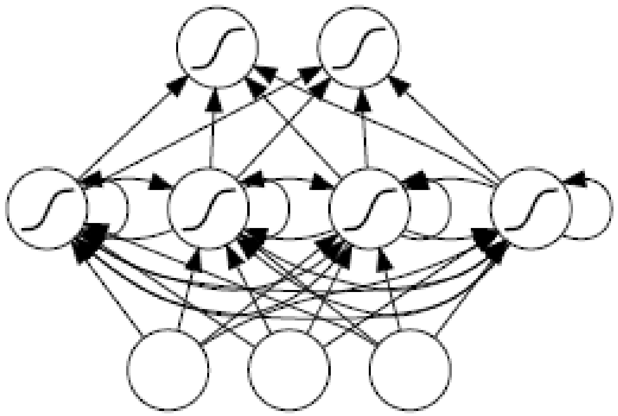

Fig. 13. Recurrent Neural Network[5].

D. Limitations of RNNs

Although RNNs are universal approaximators for any sequence to sequence mapping training RNNs is extremely difficult[21]. Note that BPTT treats a RNN as an infinitely deep(in number of layers) feed forward network. BPTT, which is an adaptation of gradient descent to MLPs works by propagating the gradient across the layers. As the gradient is propagated through the layers the magnitude of the gradient, depending on the spectral radius of the network, is either exploded or diminished leading to ineffective training. Another important problem with training RNNs using BPTT is that the solution can converge to a bad local minima which leads to poor generalization.

Some solutions to mitigate the problem are described below:

- Control spectral radius
- Normalize the gradient in each layer
- Use Hessian Free optimization instead of BPTT
- Echo State Networks
- Unsupervised pre-training.

In this paper we refrain from going into the details of the other solutions and only discuss echo state networks[22] as a solution to the mitigate the limitations referred to above.

E. Echo State Networks(ESN)

ESN mitigates the problem of exploding and vanishing gradients by controlling the spectral radius and eliminating

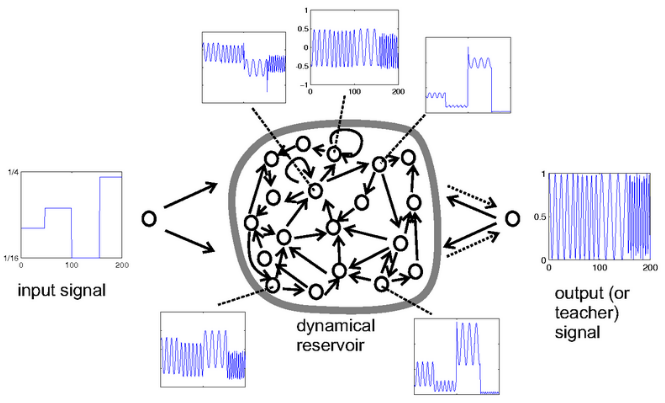

Fig. 14. Echo State Network[26]

the learning for all the layers except the last layer. This is done by creating a reservoir as shown in the figure 14. This reservoir consists of a large number of neurons with connections, both cyclic and directed, selected at random. The reservoir computes several non linear functions from the input and the final layer is trained on the output of the reservoir usually using simple Linear Regression.

Since gradient propagation is completely avoided in training the network the problem of exploding and vanishing gradients is completely eliminated in ESNs.

V. CONCLUSIONS

Sequential learning involves learning a mapping from an input space of sequences to an output space of sequences and hence algorithms which can remember and extract patterns from contextual information are essential. Hidden Markov models and their extensions use hidden state representations(both distributive and non-distributive) in a Dynamic Bayesian Network to represent sequential(or temporal) information. Though in theory they have the capability to represent any distribution, inference and learning becomes extremely costly(computationally) with increasing number of states in both distributive and non-distributive approaches. Several approximate inference and sampling based methods have been proposed in literature to mitigate this problem. Interpretation of the hidden states in HMM's is comparatively easier. In contrast, RNN's however employ highly complex representations which are difficult to be interpreted. Inference, however is straight forward in RNN's. Learning in RNN's is subject to several limitations since it leads to poor generalization. Alternate training methods and architectures like Hessian Free Optimization and Echo state networks help to solve these problems in an effective way. In order to choose a particular algorithm for a particular sequence learning task, all the limitations and the solutions discussed in this paper like computational efficiency, comprehensibility of representation, generalization capabilities need to be considered and given the right parameters and design choices, both HMM's and RNN's being Universal function approximators can represent any probability distribution or function respectively.

REFERENCES

[1] Markov, A., An example of statistical investigation in the text of 'eugene onyegin' illustrating coupling of 'tests' in chains, Proceedings of the Academy of Science, St. Petersburg, vol. 7, pp. 153-162, 1913.
[2] Pearl, J., Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference, Morgan Kaufmann, 1988.
[3] Bengio, Y., Markovian Models for Sequential Data, Neural Computing Surveys, pp. 122-162, 1999.
[4] R. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proceedings of the IEEE, vol. 77, no. 2, pp. 257-286, 1989.
[5] Graves, A., Supervised Sequence Labelling with Recurrent Neural Networks, Studies in Computational Intelligence, Springer, 2012.
[6] Bengio Y., and Frasconi, P., An input/output HMM architecture, in Advances in Neural Information Processing Systems (G. Tesauro, D. Touretzky, and T. Leen, eds.), pp. 427-434, MIT Press, Cambridge, MA, 1995.
[7] Kalman R., Bucy, R., New results in linear filtering and prediction," Journal of Basic Engineering(ASME), vol. 83D, pp. 95-108, 1961.
[8] P. Dempster, N. M. Laird, and D. B. Rubin, Maximum-likelihood from incomplete data via the EM algorithm, Journal of Royal Statistical Society B, vol. 39, pp. 1-38, 1977.
[9] Viterbi, A., Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Transactions on Information Theory, pp. 260-269, 1967.
[10] E. Baum and J. Eagon, An inequality with applications to statistical prediction for functions of Markov processes and to a model of ecology, Bull. Amer. Math. Soc., vol. 73, pp. 360-363, 1967.
[11] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, A maximization technique occuring in the statistical analysis of probabilistic functions of Markov chains, Ann. Math. Statistic., vol. 41, pp. 164-171, 1970.
[12] E. Baum, An inequality and associated maximization technique in statistical estimation for probabilistic functions of a Markov process, Inequalities, vol. 3, pp. 1-8, 1972.
[13] Ghahramani, Z., and Jordan, M. I., Factorial hidden markov models, in Advances in Neural Information Processing Systems 8 (M. Mozer, D. Touretzky, and M. Perrone, eds.), MIT Press, Cambridge, MA, 1996.
[14] willamette.edu.
[15] intechopen.com.
[16] Ghahramani Z., and G. Hinton, Switching state-space models, Tech. Rep. Technical Report CRGTR- 91-3, University of Toronto, 1996.
[17] Rosenblatt, F., The Perceptron: A Probabilistic Model For Information Storage And Organization In The Brain". Psychological Review 65 (6): 386408, 1958.
[18] Waibel, A., Phoneme Recognition Using Time-Delay Neural Networks, IEEE Transactions on Acoustics, Speech and Signal Processing, Volume 37, No. 3, pp. 328-339 March 1989.
[19] Jordan, M.I., Attractor dynamics and parallelism in a connectionist sequential machine, In Proc. of the Eighth Annual Conference of the Cognitive Science Society, pages 531-546, 1986.
[20] Elman, J.L., Finding structure in time, Cognitive Science, 14:179-211, 1990.
[21] Bengio, Y., Simard, P., Frasconi, P., Learning long-term dependencies with gradient descent is difficult. Neural Networks, IEEE Transactions on, 5(2), 157-166, 1994.
[22] Tong, M. H., Bickett, A. D., Christiansen, E. M., Cottrell, G. W., Learning grammatical structure with echo state networks. Neural Networks, 20(3), 424-432, 2007.
[23] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., Learning representations by back-propagating errors, Nature, 323, 533-536, 1986.
[24] Mozer, M. C., Y. Chauvin and D. Rumelhart, A Focused Backpropagation Algorithm for Temporal Pattern Recognition. Hillsdale, NJ: Lawrence Erlbaum Associates. pp. 137169, 1995.
[25] ccs.neu.edu.
[26] Jaeger H., Echo State Network, Scholarpedia, 2(9):2330, (2007).
[27] Bishop, C. M., Pattern Recognition, Machine Learning, 2006.