Chapter 1
Introduction

In the past 10 years within the genomics field, high-throughput technologies and decreasing costs have enabled researcher the ability to access and generate increasingly larger amounts of data (Schadt *et al.*, 2010). Making sense of all the data can quickly become an overwhelming challenge for researchers. In addition, they often need to integrate existing prior knowledge into their analysis as well as many different types of data, such as DNA-seq, RNA-seq, Methyl-seq, and ChIP-seq. Not only do researchers have to design methods for analysis of the data in meaningful ways, they are also under time constraints to get the data analyzed and published quickly. As a result, researchers often rely on existing analytic tools or need to have the ability to quickly design new tools to address their specific analysis problem.

Some sequence analysis problems, such as sequence assembly, alignment, and gene finding, have a well-established base of software. However, when it comes to other problems, such as R-loop prediction or methylation domain classification, there are not any predesigned tools. Consequently, the only option for researchers is to spend precious time developing new tools. Development of new tools, however, can be both time consuming and error prone. In most instances, these new tools are typically a re-implementation and adaptation of existing data analysis methods.

Hidden Markov models (HMMs) are an example of an established analysis method that have been widely applied in bioinformatics. In the past 20 years, HMMs have been applied to a wide-range of sequence analysis problems including sequence alignment, gene finding, motif finding, peak finding, base calling, SNP calling, copy number variation calling, and chromatin status prediction. HMMs form the basis of many bioinformatics programs for three main reasons. First, they are conceptually simple to understand and relate to many different linear labeling problems(Brejová and

Brown, 2008). HMMs label linear events in a linear dependent fashion, much the same that we organize events in time. Second, HMMs are modular. The modularity means that a complex problem can be easily broken into multiple simple models. The simple models can be trained and easily combined to solve the complex problem(Cherry, 2001). Third, HMMs are built upon a simple and powerful statistical framework (Eddy, 2004). This provides the user the to ability to adapt the framework in many different ways and maintain the statistical foundation. It also means that after a model is trained, the user can infer information from the organization and parameters of the model(Cherry, 2001). However in order to take advantage of the strengths of HMMs in labeling linear data, researchers have to first implement multiple algorithms, then train and test a model.

1.1 Brief Historical Background of HMMs

In 1906, Andrey Markov published a paper describing a memory-less stochastic process of "chains" based upon Bernoulli's and Chebyshev's work on sequences of independent random variables(Markov, 1906). His work, which later became known as "Markov chains", was focused on understanding and extending the central limit theorem and weak law of large numbers to sequences of dependent random variables(Basharin *et al.*, 2004). Markov's most famous implementation of Markov chains was his analysis of the first and part of the second chapter of Alexander Pushkin's novel *Eugene Onegin*, where he analyzed and created a model based upon patterns of consonants and vowels(Hayes, 2013; Markov, 2007). The model could then be used to generate words and phrases that were similar to Pushkin's work or to calculate the probability that another work was written by Alexander Pushkin.

Markov models were merely seen as a statistical proof until the communications field adopted them in the late 1940s. In the 1948, Claude Shannon's paper "A Mathematical Theory of Communications" described the need to apply stochastic and deterministic models in the field of communications to account for noise within communications(Shannon, 2001). Shannon used Markov models and their statistical foundation to help formulate information theory(Shannon, 2001). Markov models were seen as generative models up until the late 1960s, when Leonard E. Baum described how to infer an unknown states by using additional observations(Baum and Petrie, 1966). Baum's was the first description of what are formally known as HMMs. A year later, Andrew Viterbi described an algorithm that would find the most likely state path of an HMM model given a sequence of observations (Viterbi, 1967). This became known as the Viterbi algorithm and has gained wide acceptance in many research and technology fields. Additional advancements came in 1970, when Leonard Baum and Lloyd Welch introduced the forward-backward algorithm that could be used to compute the posterior probability of the model and determine model parameters when a training set with defined states was not available(Baum, Petrie, Soules, and Weiss, 1970b). HMMs remained fairly obscure until 1980, when Jack Ferguson and his colleagues at the Institute for Defense Analyses provided a seminar for a select group of researchers detailing the application of HMMs to speech and text(Rabiner). In 1986, Rabiner published a paper titled "An Introduction to Hidden Markov Models", which detailed the statistics and algorithms behind HMMs. That paper and his subsequent paper are now some of the most cited HMM papers and are many researchers first introduction to HMMs(Rabiner and Juang, 1986; Rabiner, 1989). Since the late 1980s, HMMs have become ubiquitous within the fields of communication and speech recognition, and the basis of modern communication devices such as the cell phone, and wireless networks(Ephraim and Merhav, 2002).

With the advancement of sequencing technology in the late 1980s, it did not take long for biologists to see the advantages that HMMs could provide in the analysis of DNA sequences. In 1989, Gary Churchill applied HMMs to multiple mitochondria sequences and human X chromosome sequence fragments. He showed that HMMs could be used to annotate origins of replication and isochore boundaries(Churchill, 1989). Since 1989, HMMs have become the basis for many bioinformatics tools.

1.2 Markov Models Basics

Andrey Markov described Markov models as a set of states (represented as circles) with transitions that linked the states (represented as arrows), each of the transitions had a given transition probabilities (see Figure 1-1)(Markov, 1906).

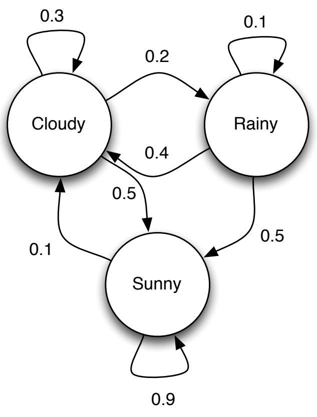

Figure 1-1 Example Markov Chain of 3 states. Cloudy, Rainy, and Sunny are states (drawn as circles). Transitions between states are drawn as arrows with an accompanying probability for the transition.

Because of the probabilistic basis of Markov models all the transition probabilities exiting from a state must sum to 1. If a transition is has a probability of 0, the transition is not allowed. Markov models are described as "memory-less" because the next state is only dependent upon the current state and possibly a limited number of previous states. Using Figure 1-1 as an example, if we start in the "Sunny" state, we can transition to either the "Sunny" or "Cloudy" states. If we then transition to the "Cloudy" state, the next choice is whether to transition to "Cloudy", "Rainy", or "Sunny". The second transition is independent of where we started. A starting state can be defined by a set of transition probabilities from an initial state (q_o) (Rabiner and Juang, 1986; Majoros, 2007). The starting state transitions are essentially the prior probability of starting at a particular state. Likewise, a set of possible ending states can be defined by a set of transition probabilities(Ω) to an ending state(q_e)(Durbin, 1998). The starting and ending states are often left out of the model diagrams and simply defined as a parameter of the model. They provide the ability to define valid starting and ending states for the model.

Through varying the topology (number of states and the transitions) between the states, we can model many different phenomena. The states and their transitions setup rules for what sequence of states the model can generate and the average time spent in a state. This allows us to model observed rules or parameters of the sequence we are studying. These rules are referred to as the grammar (sequences of states that are and are not allowed, and the distribution of states) (Durbin, 1998). Thus, Figure 1-1 could generate a path (Sunny, Sunny, Cloudy, Rainy, Sunny). However, it could not generate a sequence: (Sunny, Rainy), because the transition probability from Sunny to Rainy states is zero.

The average amount of time spent in a state can be determined from the transitions probabilities. A state with no transitions to itself can only have duration of 1. But states with transitions to themselves have a length distributions that are a geometric distribution($E(x) = \frac{1}{1-p}$) (Figure 1-2) (Brejová and Brown, 2008; Durbin, 1998). However, if we want to model something with a non-geometric distribution, we have to link multiple states in such a way as to get a different distribution. If we link multiple geometric distributed states together, we can achieve a negative binomial distribution (Durbin, 1998). Modeling more complex distributions of length(N-1) is accomplished by linking N individual states and setting the transitions to the last state for each of the previous state equal to the cumulative distribution function (Figure 1-3)(Durbin, 1998).

Modeling different distributions through this method drastically increases the number of state and as a consequence the complexity of the model.

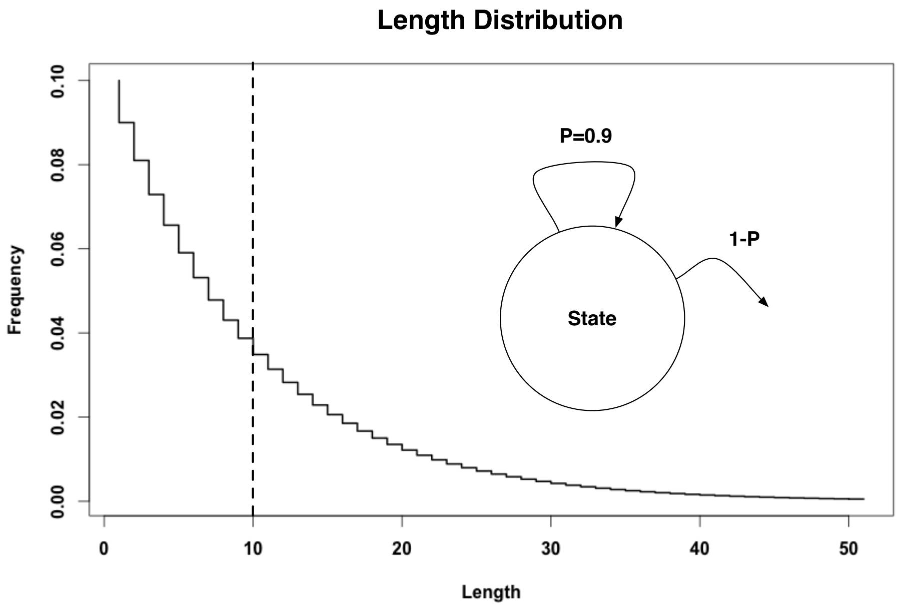

Figure 1-2 Plot of geometric length distribution generated by self-transitioning state. State with self-transition set to 0.9 on average will produce length of 10 (dotted line).

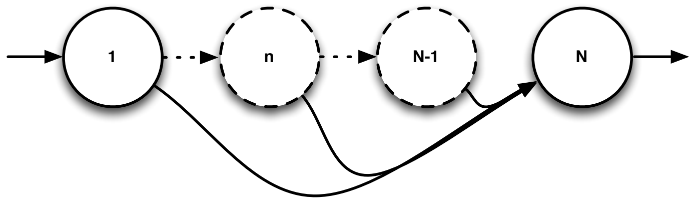

Figure 1-3 Modeling any distributions by linking a series of states together. Dotted lines and states represent multiple states necessary to model a distribution of length (N). The distribution is modeled by setting the transitions to the state(N) equal to the cumulative distribution function.

Markov models are trained from existing data by counting all transitions between states. The transitions from a state are normalized to 1 to give a probability of transitioning to different states. These are often represented as probabilities accompanying the transition arrow (Figure 1-1) or as transition matrix. Given a sequence of weather, where S=Sunny, C=Cloudy, R=Rainy:

(S,S,S,S,S,S,S,S,S,S,S,C,C,R,S,S,S,C,R,C,S, so on…)

we can generate a transitions matrix (A).

$$A = \begin{bmatrix} 0.9 & 0.1 & 0 \\ 0.5 & 0.3 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{bmatrix}$$

Once the model has been trained, it can be used as a stochastic machine to generate sequences similar to the training data or it can be used to infer probabilities of a new path through the model. For example, given a sequence of states, we can quickly calculate the probability of the path through the states and compare them to other states paths.

$$P(Sunny, Sunny, Cloudy, Rainy|\ Start\ in\ Sunny) = \ 0.9 \times 0.9 \times 0.1 \times 0.2 = 0.0162$$

$$P(Sunny, Sunny, Sunny, Sunny|\ Start\ in\ Sunny) = \ 0.9^5 = 0.6561$$

The model also allows us to quickly infer the likelihood that the model will be in a particular state at a future time. For example, if we start at the Sunny state, what is the probability of ending in the Sunny state at 5 time points later?

$$P(States\ in\ t + 5|\ Start\ in\ Sunny\ ) = [1\ 0\ 0] \cdot A^5 = [0.83504\ \ 0.13534\ \ 0.02962]$$

We see that the model has a probability of 0.83504 of ending in a Sunny state. The model allowed us to model future events and assign them a probability. While these examples are basic, more complex Markov models can be applied to predict financial markets or hurricane paths.

In sum, the Markov model provides a powerful means to analyze past events or to predict future events. While the Markov model is informative, there are many instances when the states are not observable and all we can observe is some emission or observation that is emitted from a state. An example of this is when we sequence DNA and want to annotate protein coding gene structure.

We observe the DNA base call generated by the sequencer, but we do not know whether the DNA base call corresponds to an intergenic state or a protein coding state. A similar problem was addressed by Baum in 1966, while he was working for the Institute for Defense Analyses on speech and text modeling(Baum and Petrie, 1966). His solution became known as a hidden Markov model.

1.3 Hidden Markov Models

Hidden Markov models are a subclass of Markov models in which the states are hidden and emitted observations are linked to the states as a second stochastic process. The observations, also referred to as the emissions, allow us to infer which state most likely output the observation state(Baum and Petrie, 1966). Each state can emit a single or multiple emissions. These emissions can be either discrete (associated with finite set of values) or continuous distributions (associated with any value between a range)(Rabiner, 1989). Emissions can also have a defined dependence upon a set number of previous emissions and is referred to as an emission's "order of dependence" or "order". The order of a states emission can vary from state to state. Therefore, a first-order emissions probability is $P(x = emission \mid previous\ emission)$. Figure 1-4 shows a model with zeroth-order emissions (not dependent upon any previous emissions) for all the states.

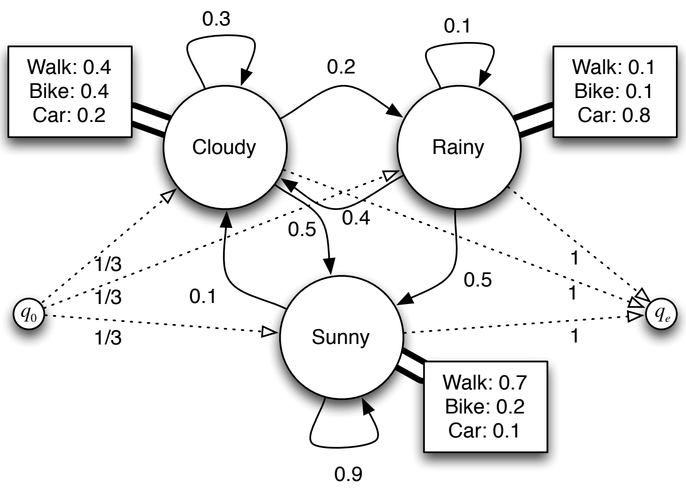

Figure 1-4 Example HMM of 3 states with associated emissions and initial/ending state. Emissions and emission probabilities for states are represented as squares linked by double line to the state. Initial state probabilities are described values of dotted lines from q_0 and Ending state transitions are all defined as 1 to q_e.

1.3.1 Developing & Training Models

The development and training phase of a project is key to its success. It is the initial point where a researchers prior knowledge and understanding of the underlying biological problem help to define the initial parameters of the model. Some initial parameters include the model topology (number and organization of states), emissions, and in some cases emission and transitions probabilities. For the most part the topology is determined by hand and then refined through multiple rounds of testing(Durbin, 1998; Majoros, 2007). Previous research showed that automatic production of a model's topology using a genetic algorithm was not a viable solution to solve complex models(Won *et al.*, 2004). Won et al also found that of smaller models, which were generated automatically, performed similar to hand-designed models but without a clear biological correlation of each state.

Once a topology is determined and a sequence of observations for which the state path is known, a model can be trained. Training is performed by merely counting the occurrence of emissions and transition for the training sequence and its state path(Brejová and Brown, 2008;

Durbin, 1998). If the specific state path is unavailable for the observation sequence, then additional training methods are necessary, such as Viterbi training (Lam and Meyer, 2009a; Majoros, 2007) or Baum-Welch training (Baum, Petrie, Soules, and Weiss, 1970a; Baum, 1972) can be performed.

Two additional techniques are commonly applied after the initial training of the model to improve performance. The first method, called boosting, tests the model on the training set, then duplicates poorly scoring examples in the training set and repeats the training using the new training set(Majoros, 2007). The second method, called bootstrapping, entails using a model trained in a different organism to create a training set in new organism. The model is first trained in a different organism. The model is used to make predictions on the new organism and the prediction results are added to the old training set(Korf, 2004). However, judicious use of both methods is necessary to prevent the model from becoming trained so that it is specific to the training set. Such a model will produce highly accurate predictions when using the training dataset, but poor predictions when applied to a test dataset.

Once a model has been trained, we can then infer the state path through the model that most likely generated a set of observations. To do this we could calculate the probability for each possible path and choose the most likely path. However, this naïve approach would require approximately 1 million calculations for a sequence of 10 observation using Figure 1-4 and would increase exponentially as the number of observations increased. More efficient algorithms (methods of calculation) were discovered, which reduced the number of calculation from 1 million to approximately 160 for a sequence of 10 observations using Figure 1-4(Rabiner, 1989).

1.3.2 HMM Algorithms

Because HMM algorithms are methods to efficiently calculate different solutions, they are often defined in mathematical terms. Each parameter of an HMM is assigned a variable; see Table 1-1(Rabiner, 1989).

Table 1-1 Variable description for parts of an hidden Markov model (Rabiner, 1989)

Variables	Descriptions
λ	Model
$Q=(q_1 \ldots q_N)$	States
N	Number of states in Model
$A=(a_{11} \ldots a_{ij})$	Transition probabilities
$\Pi=(\pi_1 \ldots \pi_N)$	Initial state transition
$\Omega=(\omega_1 \ldots \omega_N)$	Ending state transition
$B=P(v_m \mid q_n)$	Emission probabilities
$V=(v_1 \ldots v_M)$	Emission Alphabet/Words
M	Number of Emission
$O=(O_1 \ldots O_T)$	Observation Sequence
T	Length of Sequence
α	Forward score
β	Backward score
δ	Viterbi score
Ψ	Traceback pointer
γ	Posterior probability

The HMM algorithms efficiently solve three different probability questions regarding a model and a set of observations:

1) What is the probability that the model produced the observations sequence?

2) What is the probability that an observation was emitted by a given state?

3) What is the most likely path of states through the model given the observations?

The first two questions are answered by calculating the forward-backward algorithm. The Viterbi algorithm solves the third question.

1.3.2.1 Forward-Backward Algorithm

The forward-backward algorithm is a dynamic programming (DP) algorithm that propagates the probability through all possible paths for a set of observations. The DP algorithms alleviates the need to calculate each path separately and drastically reduce the number of calculations necessary(Rabiner and Juang, 1986). The forward moves from the start of the observations and progresses through all states toward the end of the observations, while the backward operates in reverse.

The forward and backward algorithms can be broken into 3 distinct stages: Initiation (1.1), Recursion (1.2), and Termination (1.3) (Durbin, 1998; Rabiner, 1989). The forward algorithms initialization step is defined equation (1.1), which defines the possible starting states. The recursion step (1.2) propagates the probability along all paths through the observations. Finally, the termination step joins the probabilities from the valid ending states and produces the probability that the model output the observation sequence(1.3).

Initialization: $$\alpha_1(i) = \pi_i \cdot b_i\left(O_1\right), \text{ for } 1 \le i \le N \tag{1.1}$$

Recursion: $$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}\right] \cdot b_j\left(O_{t+1}\right), \text{ for } 1 \le t \le T,\ 1 \le j \le N \tag{1.2}$$

Termination: $$P\left(O \mid \lambda\right) = \sum_{i=1}^{N} \alpha_T(i) \cdot \omega(i), \text{ for } 1 \le i \le N \tag{1.3}$$

The backward algorithm is very similar to the forward at first look. However, comparing the (1.1) to (1.4) and (1.2) to (1.5) show that the backward calculation does not use the emission variable for the current state calculation but uses the $b_j(O_{t+1})$ observation. Both the forward and backward should arrive at the same value $P(O|\lambda)$ (1.6) (Durbin, 1998; Rabiner, 1989).

Initialization: $$\beta_T(i) = \omega_i, \text{ for } 1 \le i \le N \tag{1.4}$$

Recursion: $$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \cdot \beta_{t+1}(j) \cdot b_j(O_{t+1}), \text{ for } t = T\text{ -}1 \cdots 1,\ 1 \le i \le N \tag{1.5}$$

Termination: $$P(O \mid \lambda) = \sum_{i=1}^{N} \pi_i \cdot \beta_1(i) \cdot b_i(O_1), \text{ for } 1 \le i \le N \tag{1.6}$$

Using the values calculated by the forward-backward algorithm and equation (1.7), the posterior probability can be calculated for each state at every position (Rabiner, 1989).

$$\gamma_t(i) = P(q_i \mid O_t) = \frac{\alpha_t(i) \cdot \beta_t(i)}{P(O \mid \lambda)} \tag{1.7}$$

The posterior probability is the probability that a state at a particular position output the observation at that position. The forward score represents the probability from the beginning of the sequence to a given point and the backward score represents the probability from the end of the sequence to the given point.

1.3.2.2 Posterior Decoding Algorithm

The posterior probability can be used to select a path of states through the model for a given set of observations, known as posterior decoding. Unlike the Viterbi decoding, which choses the path with the highest overall probability, posterior decoding selects the most likely state for each observation(Durbin, 1998; Majoros, 2007). Because posterior decoding doesn't depend upon the path, it can return state paths that have invalid grammar. Posterior decoding is recommended when many paths may be equally likely(Durbin, 1998) or when we are not interested in overall most likely path, but rather the likelihood that a state output the observation.

1.3.2.3 Viterbi Decoding Algorithm

The Viterbi algorithm provides a solution to find the most likely path through the model given a sequence. It is closely related to the Forward algorithm and very similar to the Needleman-Wunsch global alignment algorithm. The first difference to the Forward algorithm is seen in the Initialization step where a pointer is initialized to 0. The Viterbi algorithm changes the summation function used in the Forward algorithm (1.2) to the max function (1.9). Because it is seeking only the optimal solution, all other scores are discarded. Like alignment algorithms, it keeps traceback pointers(Ψ) for each state and observation(1.8)(1.9)(1.10). The traceback step (1.11) operates in reverse and follows the traceback pointers stored in (1.9) and (1.10) to determine the most likely overall traceback path(Durbin, 1998; Rabiner, 1989).

Initialization:
$$\begin{aligned}&\delta_1(i) = \pi_i \cdot b_i(O_1), \text{ for } 1 \le i \le N \\ &\psi_1(i) = 0\end{aligned} \tag{1.8}$$

Recursion:
$$\begin{aligned}&\delta_t(j) = \max_{1 \le i \le N}\left[\delta_{t-1}(i) \cdot a_{ij}\right] \cdot b_i(O_t), \text{ for } 2 \le t \le T,\ 1 \le j \le N \\ &\psi_t(j) = \operatorname*{argmax}_{1 \le i \le N}\left[\delta_{t-1}(i) \cdot a_{ij}\right]\end{aligned} \tag{1.9}$$

Termination:
$$\begin{aligned}&\delta_{end} = \max_{1 \le i \le N}\left[\delta_T(i) \cdot \omega_i\right], \text{ for } 1 \le i \le N \\ &\psi_{end} = q_T = \operatorname*{argmax}_{1 \le i \le N}\left[\delta_T(i) \cdot \omega_i\right]\end{aligned} \tag{1.10}$$

Traceback:
$$q_t = \psi_{t+1}(q_{t+1}), \text{ for t=T-1} \cdots 1 \tag{1.11}$$

The Viterbi algorithm is guaranteed to find the overall most likely path through the model that created the observations. However, it only keeps track of a single pointer through the model. Therefore, if any other paths are equally likely, they will not be seen. This leads to race conditions, where the first evaluated path will be chosen over those that are equally likely (Figure 1-5).

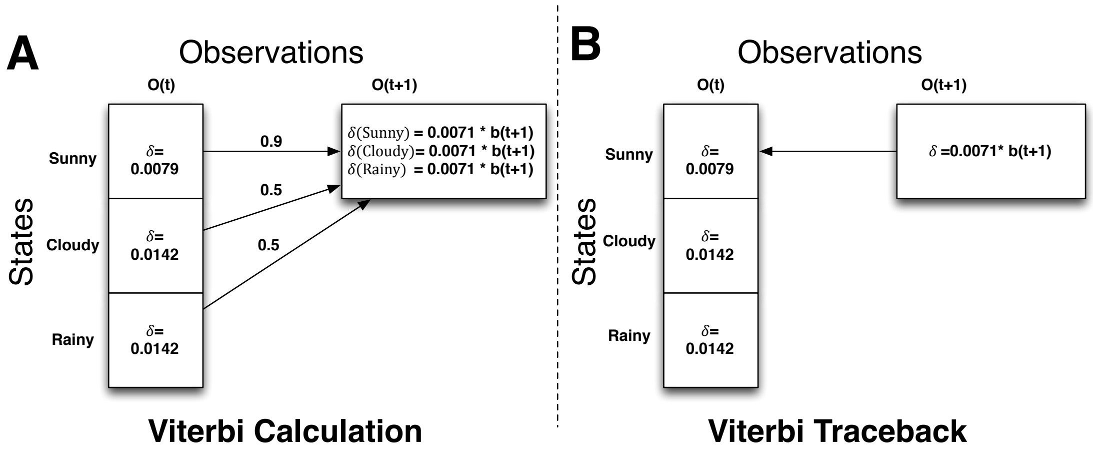

Figure 1-5 Viterbi calculations and assigning traceback pointer of 3 equally likely paths using Weather model (Figure 1-4). (A) Calculation of competing Viterbi scores for the 3 states (Sunny, Cloudy, Rainy). Arrows from O(t) to O(t+1) represent transitions values for transitioning from current state to the next state. (B) Because of MAX and ARGMAX functions in Viterbi, the traceback pointer(Left pointing arrow) for the Viterbi points only to Sunny state, even though all three paths are equally likely.

This illustrates a limitation of the Viterbi algorithm in dealing with situations where other paths are equally likely. When applying the basic HMM framework and algorithms, researchers have encountered many such limitations. To address such limitations encountered with the basic HMM algorithms, researchers have adapted and extended the HMM framework.

1.4 Extending the Hidden Markov Model Framework

The traditional HMM framework provides powerful solutions to solve many problems, however, it also has limitations. First, the optimal HMM solutions isn't always the same as the optimal biological solution. For example, the Viterbi and Posterior decoding algorithms are limited to a single prediction, while alternative splicing is a fundamental process in many eukaryotes. Second, the analysis problem my not fit nicely within the traditional HMM framework. Often researchers have additional data that needs to be integrated into the model in ways that the traditional HMM framework wasn't designed to accommodate.

Notwithstanding these limitations, HMMs simplicity and statistical foundation are the reasons that they are easily adapted to overcome the limitations. The two main focuses for adaptation have been the HMM algorithms and abstraction of the framework to integrate additional data at as emission or transitions with in the model.

1.4.1 Suboptimal paths and stochastic sampling

The two standard HMM decoding algorithms (Posterior and Viterbi) only produce single paths through the states. For many years, this has been one of the limitations that have prevented HMM-based gene finders from predicting alternative splice isoforms. Stochastic sampling allows researchers to generate suboptimal paths and explore the probabilities behind those paths. The stochastic Forward and N-Best Viterbi algorithm addresses some of the single path limitations of the Posterior and Viterbi.

1.4.1.1 Stochastic Forward

In 1995, the stochastic Forward algorithms was proposed as a solution to overcome local maxima when using the Baum-Welch algorithm to train multiple alignment profile HMMs(Eddy, 1995; Durbin, 1998). By sampling additional paths from the posterior probability(Zhu *et al.*, 1998), the Baum-Welch algorithm could be nudged from local maxima and then be allowed to converge on the global maxima (Eddy, 1995; Durbin, 1998). Using mathematical lemmas and an algorithm presented by Jun Zhu(Zhu *et al.*, 1998), the stochastic backward algorithm was applied to alternative isoform gene finding in SLAM(Cawley and Pachter, 2003; Alexandersson *et al.*, 2003) and Augustus(Stanke, Keller, *et al.*, 2006). The stochastic forward was also referenced as an equally viable alternative to the stochastic backward(Cawley and Pachter, 2003).

The implementation of the stochastic forward is a mixture of the Viterbi and Forward algorithms. Initialization is the same as Viterbi algorithm. The recursion step calculates the forward the score (1.2) and records each transitions contribution to the full forward score (1.12). Likewise, the termination calculates $P(O|\lambda)$ and the traceback probabilities (1.13). The traceback step of the stochastic forward is a stochastic process and relies on a uniform random number generator that generates random variables between 0 and 1(1.13). From the end state to the first state, at each position a random number is generated and the traceback pointer are chosen according to the traceback probability (1.13).

Initialization: Same as (1.8)

Recursion: Same as (1.2)

$$P\left(\psi_t\left(j\right)=i\right)=\frac{\delta_{t-1}\left(i\right)\cdot a_{ij}}{\alpha_{t+1}\left(j\right)}, \text{for } 2 \le t \le T,\ 1 \le j \le N, 1 \le i \le N \tag{1.12}$$

Termination: Same as (1.3)

$$P(\psi_{end}=i)=\frac{\alpha_T\left(i\right)\cdot\omega_i}{P\left(O\mid\lambda\right)}, \text{ for } 1 \le i \le N \tag{1.13}$$

Traceback: Select back-track path according to X~unif(0,1)

Multiple tracebacks lead to a stochastic sampling of state paths through the model. From these paths the posterior probability can be estimated for sequence features(Cawley and Pachter, 2003; Stanke, Keller, *et al.*, 2006). An additional advantage over using the posterior probability, is that the stochastically sampled path must conform to proper model grammar, where as the posterior has no such guarantee.

1.4.1.2 N-best Viterbi

In 1990, Richard Schwartz presented an algorithm to find the Nth most likely path(Schwartz and Y. Chow, 1990). The N-best Viterbi is a modified Viterbi algorithm that is guaranteed to find the top N most likely paths for a sequence. It increases the number of Viterbi scores and traceback pointers stored from to one to N for each state and position for the Recursion and Termination steps. After the calculation of each competing Viterbi score, the scores are sorted and the best N scores are stored. This leads to an increased memory usage of at least N times. The Traceback step then traces back through the N pointers stored in the Termination step. The first traceback pointer and score correspond to the Viterbi score and path. The second corresponds to the second most likely path and so on.

1.4.2 Integration of additional data

Besides using prior knowledge to inform the model topology and training, an increasing popular area is integration of additional data sources into the HMM to improve predictions. Because of the probabilistic foundation of HMMs, data can be integrated into HMM at multiple points. Additional information could be included in the HMM framework through linking additional evidence to the transition or emission probabilities of states. Successful examples can be found in gene finding, where EST, protein alignments, or related genome alignments, are used to augment model probabilities(Brejová and Brown, 2008).

An example of a way of incorporating additional data is through using multiple emissions states in the model. Multiple emission states can combine the probabilities from two or more observations as either independent or joint distributions. Two examples of tools that employ this method are AUGUSTUS+ and PennCNV. AUGUSTUS+ emits both a sequence and a gene structure hint jointly in order to arrive at better gene predictions(Stanke, Schöffmann, *et al.*, 2006). The hints are based upon EST, protein alignments, or user knowledge. PennCNV combines 4 different signals (intensity, allelic intensity, distance between SNPs, and allele frequencies) to predict the copy number state(Kai Wang, 2007). With increased availability of expression data, genome-wide methylation, histone modification, and replicate datasets, these data sets can be combined to improve sequence analysis(Stanke, Schöffmann, *et al.*, 2006; Allen and Salzberg, 2005; Seifert *et al.*, 2012; Ernst and Kellis, 2012).

Additional data can also be integrated by adjusting the definition of the traditional HMM framework. For example, HMMoc provides researchers the ability to link transitions to the observation sequence(Lunter, 2007). A couple of examples of the most widely modified HMM frameworks are: 1) Generalized HMMs 2) Pair HMMs 3) Profile HMMs

Generalized HMMs were created to address the complexity of HMMs when modeling non-geometric length distributions, which leads to an exponential consumption of limited computational resources. The generalized HMMs model non-geometric length distributions by allowing a state to emit multiple observations with a defined length distribution instead of just a single nucleotide(Kulp *et al.*, 1996; Majoros, 2007). In this way, the complexity of the model is greatly reduced and is able to more accurately model non-geometric length distributions.

Pair HMMs were developed to adapt HMMs to sequence alignment. Unlike multiple emission states, which operates on multiple observations in tandem, the pair HMMs operates upon two observations independently and emits an alignment(Durbin, 1998). Thus, pair HMMs are able to produce results similar to Needleman-Wunsch and Smith-Waterman alignment algorithms(Brejová and Brown, 2008).

Profile HMMs are a further extension of HMM, built of the basic framework of the Pair HMM(Durbin, 1998). Instead of operating on two sequences, the profile HMM operates upon a sequence and a multiple alignment profile(Krogh *et al.*, 1994). One of the most famous examples of a tool that implements profiles HMMs is HMMER, which allows you to score a sequence against a protein family profiles and sequence domains.

1.5 HMM Compilers, Libraries, Toolboxes, and Applications

Currently, there are a few different alternatives available to help researchers who need to develop and apply HMMs. They provide researchers with help to create HMM-based tools without having to invest time into implementing and debugging code for the fundamental HMM algorithms. These HMM implementation tools fall into four categories: 1) Compilers 2) Libraries 3) Toolboxes 4) Applications. Compilers translate XML parameters files to HMM specific code that can then be integrated into applications. However, if the HMM must be changed the code must be re-generated and re-integrated. Libraries provide researchers with a set of functions that allow the user to implement a HMM application quickly. Toolboxes are packages that are provided for existing statistical or mathematics programs, such as R and Matlab. These typically function as teaching tools, because they are not designed to handle large datasets. Applications provide the most accessible option, however, if not implemented correctly they can also be the most restrictive.

Even though these options exist, many of the recent HMM programs are still generated by hand. The primary reason being that the tools lack the necessary features and flexibility that are required by researcher(Lunter, 2007). Table 1-2 shows that many tools lack support for features that are required to allow researchers to flexibly implement HMMs. Additional reasons why they are not being adopted are: 1) they often lack sufficient documentation and support to allow researchers to adequately learn the tool. 2) Compilers and libraries are inaccessible to any researcher that doesn't program in C/C++, unless they provide bindings to different languages. 3) Code often is not written to work on different systems or is not maintained.

		Features																		
Tools	Supported HMM Types	Defineable Alphabet	Ambiguous Characters	Silent states	Discrete Emission	Higher-order Emissions	Continuous Emission	Multiple Emissions	Joint Emissions	Lexical Transitions	Link Tools to Emission/Transition	Viterbi Decoding	Posterior Decoding	N-Best Decoding	Stochastic Forward	Stochastic Viterbi	Weighted Path	Explicit Path	Add'l Language Bindings	Portability	Dependencies
Compilers																					
HMMoc	traditional, pair, generalized	✓		✓	✓	✓		✓		✓		✓	✓		*		✓			Linux, Windows, OS X	none
HMMConverter	traditional, pair, generalized	✓		✓	✓			✓				✓					✓			Linux	none
Libraries																					
GHMM	traditional,pair, generalized	✓		✓	✓	✓	✓	✓				✓							✓	Linux, OS X	libxml2, python
HMMlib	traditional	✓			✓							✓	✓							Linux, Windows, OS X	Cmake, Boost
StochHMM	traditional	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		Linux, Windows, OS X	none
Toolboxes																					
R-HMM	traditional	✓			✓		✓					✓								Linux, Windows, OS X	n/a
Matlab	traditional	✓			✓							✓	✓							Linux, Windows, OS X	n/a
Implementaion Applications																					
MAMOT	traditional	✓			✓							✓	✓							Linux, Windows, OS X	none
StochHMM	traditional	✓	✓		✓	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓		Linux, Windows, OS X	none

* no dedicated function for sampling provided. Ability to sample from posterior is stated as possible with further implementation

Table 1-2 Comparison of features available in existing HMM compilers, libraries, toolkits, and applications. StochHMM is included in both Libraries and Applications because it is provided as both. (Lunter, 2007; Lam and Meyer, 2009b; Schliep *et al.*, 2003; Sand *et al.*, 2010; Schütz and Delorenzi, 2008)

1.10 References

Alexandersson,M. *et al.* (2003) SLAM: cross-species gene finding and alignment with a generalized pair hidden Markov model. *Genome Res.*

Allen,J.E. and Salzberg,S.L. (2005) JIGSAW: integration of multiple sources of evidence for gene prediction. *BIOINFORMATICS*, **21**, 3596–3603.

Basharin,G.P. *et al.* (2004) The life and work of A.A. Markov. *Linear Algebra and its Applications*, **386**, 3–26.

Baum,L.E. (1972) An equality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities, Vol. 3 (1972), pp. 1-8*, **3**, 1–8.

Baum,L.E. and Petrie,T. (1966) Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *Ann. Math. Statist.*, **37**, 1554–1563.

Baum,L.E., Petrie,T., Soules,G., and Weiss,N. (1970a) A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains. *Ann. Math. Statist.*, **41**, 164–171.

Baum,L.E., Petrie,T., Soules,G., and Weiss,N. (1970b) JSTOR: The Annals of Mathematical Statistics, Vol. 41, No. 1 (Feb., 1970), pp. 164-171. *Ann. Math. Statist.*

Brejová,B. and Brown,D. (2008) Advances in hidden Markov models for sequence annotation.

Cawley,S.L. and Pachter,L. (2003) HMM sampling and applications to gene finding and alternative splicing. *BIOINFORMATICS*, **19 Suppl 2**, ii36–41.

Cherry,C. (2001) CMPUT 606 Computational Biology & Bioinformatics Project Proposal A General Survey of Hidden Markov Models in Bioinformatics.

Chow,L.T. *et al.* (1977) An amazing sequence arrangement at the 5′ ends of adenovirus 2 messenger RNA. *Cell*, **12**, 1–8.

Churchill,G.A. (1989) Stochastic models for heterogeneous DNA sequences. *Bull. Math. Biol.*, **51**, 79–94.

Durbin,R. (1998) Biological Sequence Analysis Cambridge University Press.

Eddy,S.R. (1995) Multiple alignment using hidden Markov models. *Ismb*.

Eddy,S.R. (2004) What is a hidden Markov model? *Nat Biotechnol*, **22**, 1315–1316.

Ephraim,Y. and Merhav,N. (2002) Hidden Markov processes. *IEEE Trans. Inform. Theory*, **48**, 1518–1569.

Ernst,J. and Kellis,M. (2012) ChromHMM: automating chromatin-state discovery and characterization. *Nat Methods*, **9**, 215–216.

Hawkins,R.D. *et al.* (2010) Distinct epigenomic landscapes of pluripotent and lineage-committed human cells. *Cell Stem Cell*, **6**, 479–491.

Hayes,B. (2013) First Links in the Markov Chain. *AMERICAN SCIENTIST*.

Huertas,P. and Aguilera,A. (2003) Cotranscriptionally Formed DNA:RNA Hybrids Mediate Transcription Elongation Impairment and Transcription-Associated Recombination. *Mol. Cell*, **12**, 711–721.

Kai Wang,M.L.D.H.R.L.J.G.S.F.A.G.H.H.M.B. (2007) PennCNV: An integrated hidden Markov model designed for high-resolution copy number variation detection in whole-genome SNP genotyping data. *Genome Res*, **17**, 1665.

Korf,I. (2004) Gene finding in novel genomes. *BMC Bioinformatics*.

Krogh,A. *et al.* (1994) Hidden Markov models in computational biology: Applications to protein modeling. … *of molecular biology*.

Kulp,D. *et al.* (1996) A generalized hidden Markov model for the recognition of human genes in DNA. *Proceedings / International Conference on Intelligent Systems for Molecular Biology ; ISMB International Conference on Intelligent Systems for Molecular Biology*, **4**, 134–142.

Lam,T.Y. and Meyer,I.M. (2009a) Efficient algorithms for training the parameters of hidden Markov models using stochastic expectation maximization EM training and Viterbi training. *CORD Conference Proceedings*, –.

Lam,T.Y. and Meyer,I.M. (2009b) HMMCONVERTER 1.0: a toolbox for hidden Markov models. *Nucleic Acids Res*, **37**, e139–e139.

Lee,D.Y. and Clayton,D.A. (1996) Properties of a primer RNA-DNA hybrid at the mouse mitochondrial DNA leading-strand origin of replication. *Journal of Biological Chemistry*, **271**, 24262–24269.

Lister,R. *et al.* (2011) Hotspots of aberrant epigenomic reprogramming in human induced pluripotent stem cells. *Nature*, **471**, 68–73.

Lister,R. *et al.* (2009) Human DNA methylomes at base resolution show widespread epigenomic differences. *Nature*, **462**, 315–322.

Lobry,J.R. (1996) Asymmetric substitution patterns in the two DNA strands of bacteria. *Mol Biol Evol*.

Lunter,G. (2007) HMMoC--a compiler for hidden Markov models. *BIOINFORMATICS*, **23**, 2485–2487.

Majoros,W.H. (2007) Methods for computational gene prediction Cambridge Univ Pr.

Markov,A.A. (2007) An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains. *SIC*, **19**, 591.

Markov,A.A. (1906) Extension of the law of large numbers to dependent quantities. *Izv. Fiz.-Matem. Obsch. Kazan Univ.(2nd Ser)*, **15**, 135–156.

Rabiner,L. First-Hand:The Hidden Markov Model. *ieeeghn.org*.

Rabiner,L. and Juang,B. (1986) An Introduction to Hidden Markov Models. *IEEE ASSP Mag.*, **3**, 4–16.

Rabiner,L.R. (1989) A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, **77**, 257–286.

Roberts,R. and Crothers,D. (1992) Stability and properties of double and triple helices: dramatic effects of RNA or DNA backbone composition. *Science*, **258**, 1463–1466.

Roy,D. and Lieber,M.R. (2009) G clustering is important for the initiation of transcription-induced R-loops in vitro, whereas high G density without clustering is sufficient thereafter. *Mol Cell Biol*, **29**, 3124–3133.

Sand,A. *et al.* (2010) HMMlib: A C++ Library for General Hidden Markov Models Exploiting Modern CPUs. *CORD Conference Proceedings*, 126–134.

SantaLucia,J.,Jr *et al.* (1996) Improved Nearest-Neighbor Parameters for Predicting DNA Duplex Stability†. *Biochemistry*.

Schadt,E.E. *et al.* (2010) Computational solutions to large-scale data management and analysis. *Nat Rev Genet*, **11**, 647–657.

Schliep,A. *et al.* (2003) Using hidden Markov models to analyze gene expression time course data. *BIOINFORMATICS*, **19 Suppl 1**, i255–63.

Schütz,F. and Delorenzi,M. (2008) MAMOT: hidden Markov modeling tool. *BIOINFORMATICS*, **24**, 1399–1400.

Schwartz,R. and Chow,Y. (1990) … N-best algorithms: an efficient and exact procedurefor finding the N most likely sentence hypotheses. *Acoustics*.

Seifert,M. *et al.* (2012) MeDIP-HMM: genome-wide identification of distinct DNA methylation states from high-density tiling arrays. *BIOINFORMATICS*, **28**, 2930–2939.

Shannon,C.E. (2001) A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, **5**, 3–55.

Stanke,M., Keller,O., *et al.* (2006) AUGUSTUS: ab initio prediction of alternative transcripts. *Nucleic Acids Res*, **34**, W435–W439.

Stanke,M., Schöffmann,O., *et al.* (2006) Gene prediction in eukaryotes with a generalized hidden Markov model that uses hints from external sources. *BMC Bioinformatics*, **7**, 62.

Sugimoto,N. *et al.* (1995) Thermodynamic parameters to predict stability of RNA/DNA hybrid duplexes. *Biochemistry*, **34**, 11211–11216.

Sun,Q. *et al.* (2013) R-loop stabilization represses antisense transcription at the Arabidopsis FLC locus. *Science*, **340**, 619–621.

T Asai,T.K. (1994) D-loops and R-loops: alternative mechanisms for the initiation of chromosome replication in Escherichia coli. *Journal of Bacteriology*, **176**, 1807.

Thomas,M. *et al.* (1976) Hybridization of RNA to double-stranded DNA: formation of R-loops.

Viterbi,A. (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, **13**, 260–269.

Won,K.J. *et al.* (2004) Training HMM structure with genetic algorithm for biological sequence analysis. *BIOINFORMATICS*, **20**, 3613–3619.

Wu,P. *et al.* (2002) Temperature dependence of thermodynamic properties for DNA/DNA and RNA/DNA duplex formation. *European Journal of Biochemistry*, **269**, 2821–2830.

Yu,K. *et al.* (2003) R-loops at immunoglobulin class switch regions in the chromosomes of stimulated B cells. *nature immunology*.

Zarrin,A.A. *et al.* (2004) An evolutionarily conserved target motif for immunoglobulin class-switch recombination. *nature immunology*, **5**, 1275–1281.

Zhu,J. *et al.* (1998) Bayesian adaptive sequence alignment algorithms. *BIOINFORMATICS*, **14**, 25–39.