HMM for Gene Annotation

The problem and the solution

What we see is DNA and cannot see underlying genes. Not all DNA corresponds to genes. Given a DNA sequence how to detect subsequences of DNA corresponding to genes?

We can use a Hidden Markov Model to determine the underlying proteins (or genes) given a DNA sequence as the input.

The rest of the article explains how this is done.

Background — DNA, RNA and Genes

The following is a DNA sequence.

```
ACCTGCTACTCG
```

We use the process called *Transcription* to convert DNA into RNA. The following is the corresponding RNA strand.

```
UGGACGAUGAGC
```

We did the following mapping in the transcription process

```
A > U, C > G, G > C, T > A
```

Once RNA is *pre-RNA* as it has both *Introns* and *Exons*. *Introns* do not code for proteins and hence should be removed. *Exons* should be spliced together to build *mature mRNA*. *Exons* do code for proteins. For more information about *introns* and *exons* please watch this video by Beverly Biology.

A *Codon* is a DNA strand a length of 3 bases. Next we splice mature mRNA into codons. Let's assume that the mature mRNA is the same as the original pre-RNA. Then we get 4 codons. Here let's assume the sequence has no *introns*.

```
UGG ACG AUG AGC
```

We look up the genetic code table to reveal the corresponding gene.

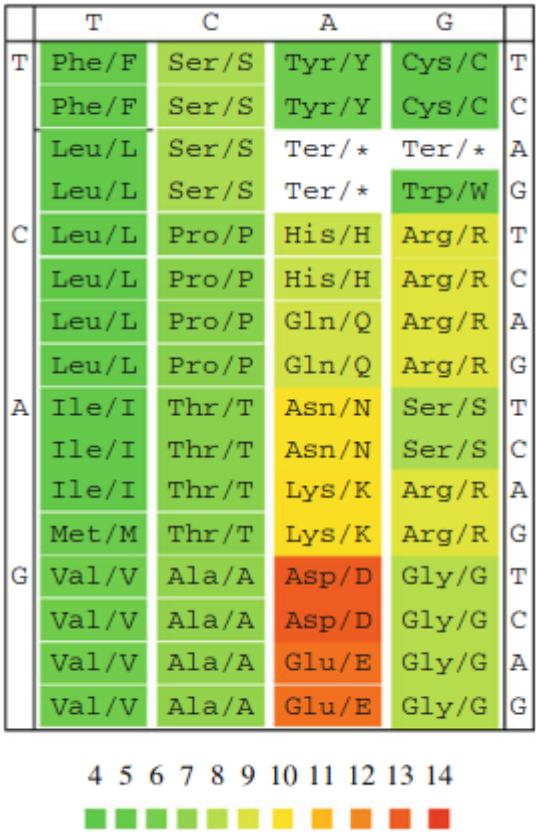


Fig 1 — The genetic code with three-letter and single-letter amino acid designations, and color coding according to amino acid polarity. The three bases in a codon are matched against the chart in the order — leftmost column, header, and the rightmost column. (source- Bioinformatics for Evolutionary Biologists by Bernhard Haubold and Angelika Börsch-Haubold)

Markov Property

Consider predicting the next word of a sentence.

```
The first day at ...
```

Fundamentally we can propose two different models to predict the next word after "*at*". The first model would only consider the current word — *"at"* to predict the next word. The second model would consider a set of previous words to predict the word, e.g., {*the, first, day, at*}.

$$P(X_{n+1} = j \mid X_1 = i_1, X_2 = i_2, \cdots, X_{n-1} = i_{n-1}, X_n = i) = P(X_{n+1} = j \mid X_n = i)$$
$$\forall i, j, i_1, \cdots, i_{n-1} \in \mathcal{S} \textbf{ and } \forall n\,.$$

Fig 2— Probability of **n+1** state being **j** given a history of states, only depends on the current state **n** being **i.** Not to be confused with the total probability of ***n+1*** *state being* ***j*** *which is the sum of all such probabilities.*

The first model is based on Markov Property, i.e., if the conditional probability distribution of future states of a stochastic process depends only upon the present state and not on the sequence of events that preceded it, the model adopts Markov Property.

note — in the above example the first model is not an ideal prediction model as it does not consider the previous states. However, some problems can be formulated with Markov property with greater confidence.

e.g.: determining the state of the next state (whether the base is a part of a gene or not) can be done without "memorizing" the states of the past bases, except for the current base. Since we have a finite number of states and it does not depend on the time when we transit from the current state to next, gene annotation problem comes under **discrete time, discrete states Markov chain**. Also determining the probability of the next state does not depend on how many states were evaluated before, hence we can classify this problem under **time-homogeneous** chains.

What is a HMM?

HMM is an extension of the Markov Model (refer https://en.wikipedia.org/wiki/Markov_model). In HMM we come across events and observations.

The event (or internal state) is not observable and can only be predicted by the observations and events are responsible for emitting observations.

If the number of internal states is N, the transition probabilities are described by a N x N matrix.

the value in i^{th} row and j^{th} column gives the transition probability from i^{th} state to j^{th} state.

If the number of observations is M, the emission probabilities are described by a N x M matrix.

the value in i^{th} row and j^{th} column gives the probability of emitting j^{th} observation when the state is i.

note — if the model was a completely deterministic model (no probability phenomenon) each event will only correspond to a particular observation.

Notation	Description
U	The set of all the N possible internal states
X	The set of all the M possible external states
L	The length of the sequence
k	A time instant, where k ∈ {1,.....,L}
Sk	Internal state at time k, where Sk ∈ U
S = {S1,S2,S3,...}	A sequence of internal states
Ek	Emission at time k, where Ek ∈ X
A	The N x N matrix of elements auv
auv = P(sk = v|sk-1 = u)	The transition probability from state u to state v
B	The N x M matrix of elements bux
bux = P(ek = x|sk = u)	The probabilities of the emission x from the state u
π	The N vector of πk elements
πu = P(s1 = u)	The probability of having the state u as the initial state
λ = (A,B,π)	The definition of the HMM model

Fig 3— Notations used in HMM model src- https://scialert.net/fulltext/?doi=jas.2012.1518.1525

L represents the sequential structure of the input and in our case *k* is not time but represents the current DNA base we are at.

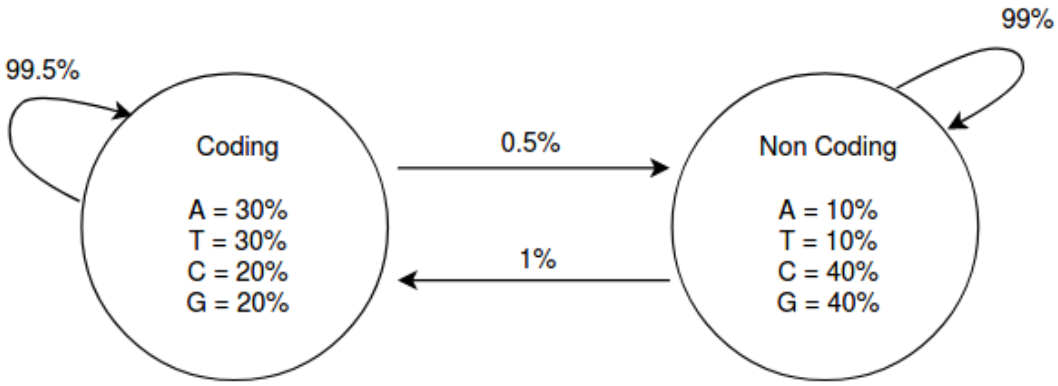
π is a N x 1 vector consisting of initial probabilities. For example if N =3 and π = [0.25 0.15 0.60] then there is a 1/4 chance of starting at state 1.

`From a biological point of view it is most likely to start from a DNA base that does not count to a gene thus, initial probability corresponding to that state will be much higher.`

Example

Consider a model with two internal states Coding (C) and Non-coding (N). A coding state represents *exons* and non-coding state represents the reset of the *codons*. We have four observations — A, C, T, and G. How can we fill in the matrices A and B? This is called the

learning problem of HMM. In this example we only study the evaluation problem and the decoding problem.



Model 1— Coding and Non-coding are the two hidden states and the arrows mark transitions and their probabilities. Inside each state, the observations are described with their emission probabilities at a particular state src-https://scialert.net/fulltext/?doi=jas.2012.1518.1525

```
          C       N
A = C [0.995   0.005]
    N [0.010   0.99]          A   T    C    G
B = C [0.3  0.3  0.2  0.2]
    N [0.1  0.1  0.4  0.4]
```

Evaluation problem

This is used to evaluate the model. Given a DNA sequence (observations sequence), the probability of emitting a particular observation is calculated. Note that a hidden state is responsible for an observation. Because of the probability phenomenon an observation can be decoded to many hidden states. For example in the above matrix B, observing base T can be decoded to both states C and N. Decoding the best hidden state sequence is the decoding problem. It is necessary to consider all the possible hidden state sequences when calculating the probability of an observation. If a real-world observation yields an unsatisfactory probability the model is poor and should be improved by revisiting the learning problem and/or modifying the model.

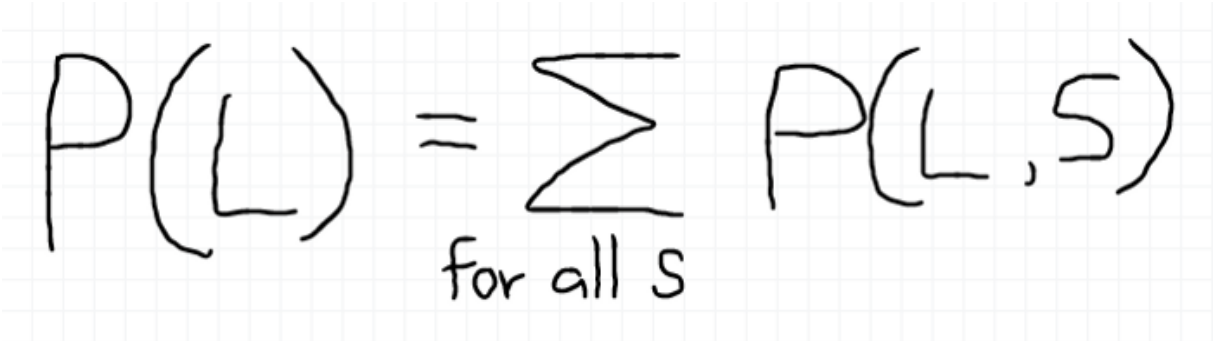


Fig 4— L is the observation sequence. S is an arbitrary internal state sequence.

```
example : If π = [0.8 0.2], evaluating the above HMM for an observation
sequence (TG) yields a probability of 5.6% approximately.
```

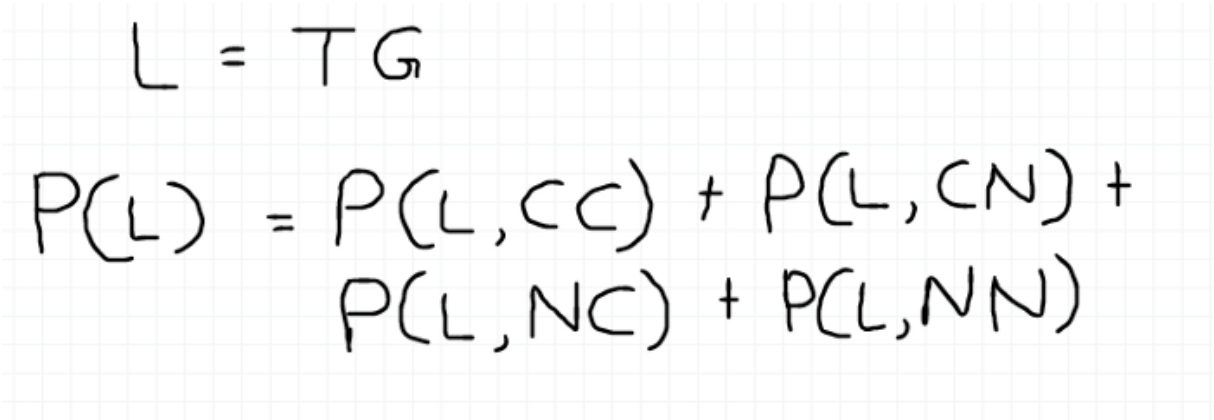


Fig 5—L is the observations sequence. C denotes the internal state "Coding" and N denotes the internal state "Non-coding".

Fig 6— Probability of observing the sequence TG if the internal state sequence is NC.

When the state space and the observation sequence gets larger the more compute-intensive the evaluation becomes, e.g., a model with 2 states has to consider 2^3 hidden state sequences for a given sequence with 3 observations. Because of this forward algorithm and backward algorithms are used to evaluate the model efficiently.

Decoding problem

In this problem we find the most probable hidden state sequence that gives rise to a particular observation sequence. For example for a given DNA sequence the most probable C and NC states sequence is found. There are two variations in decoding problem,

1. Find the sequence of internal states that has, as a whole, the highest probability (Viterby algorithm).
2. Find for each observation the internal state that has the highest probability (Posterior-decoding algorithm).

Though the two problems seem the same, Viterby algorithm gives a "valid" hidden state sequence where Posterior decoding algorithm may give an invalid hidden state sequence. An invalid state sequence would have two consecutive states P and Q, but the transition matrix describes as there is a zero transition probability from P to Q.

The notion of a stochastic process

We can present the idea in decoding problem 1 mathematically,

$$P(S = s_1,....s_n,\ L = l_1,....l_n)$$

$$P(S = s'_1,....s'_n,\ L = l_1,....l_n)$$

.
.
.

Fig 7— The joint probabilities — for a given observation sequence L possible hidden state sequence(s) S

We can treat S as a random variable. More precisely as a random vector that has a set of random variables. $s_1,....s_n$ and $s'_1,....s'_n$ are two realizations. In either case, the definition of a stochastic process matches the presentation.

— A random process or stochastic process is a family of random variables indexed by an ordered or index set denoted by T. The index set can be discrete or continuous and usually refers to time.—

Generative idea

Starting from a particular state, if we keep transiting from state to state, **Model 1** would generate varying lengths of coding and non-coding regions but they would have average lengths of 99.5 and 99 bases respectively. That is because there is a 99.5% chance of staying on a coding state and a 99% chance of staying on a non-coding state.

Time dynamics

Though the common usage of HMM in gene annotation is to reveal the underlying states given an observation sequence, we can use the matrix **A** to analyze its time dynamics to find the probability of n^{th} state being j, given the 0th state is i.

$$p_{ij}^{(n)} = P(X_n = j \mid X_0 = i)\,.$$

Fig 8— n-step transition probability

The matrix A^n has entries for n-step transition probabilities.

$$P = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}.$$

$$P^n = \frac{1}{p+q}\begin{pmatrix} q & p \\ q & p \end{pmatrix} + \frac{\lambda_2^n}{p+q}\begin{pmatrix} p & -p \\ -q & q \end{pmatrix}.$$

$$\lim_{n\to\infty} P^{(n)} = \frac{1}{p+q}\begin{pmatrix} q & p \\ q & p \end{pmatrix}$$

Fig 9— If P is a 2 x 2 transition matrix, P^n can be calculated where λ2 = 1-p-q and p and q are transition probabilities

In our example p = 0.005, q = 0.01 and λ2 = 0.94. When n→∞, that is when the state sequence grows infinitely, we will find state *C* and *NC* with 66.67% and 33.33% probabilities respectively.Regardless of the initial state there is a 16.67% chance of landing on state *C* and an 83.33% chance of landing on state *NC*. Therefore, this is a **limit distribution**.

Classification of states

The following are two classifications,

1. Recurrent and transient states (classes)
2. Periodic and aperiodic states (classes)

For more information about the classification of states refer
https://www.probabilitycourse.com/chapter11/11_2_4_classification_of_states.php

In the above state machine, state *C* can be accessed from state *NC* and vice versa. Hence state *C* and *NC* are communicating with each other,i.e., both the states are in the same communicating class and it is the only class. Therefore the state machine is **irreducible** and the class is a **recurrent class** (in a finite Markov chain there is at least one recurrent class). Since the states in a recurrent class are also recurrent, here both the states are recurrent, i.e., the total number of visits to a state over time is infinite.The state machine has **zero absorbing states** (an absorbing state's leaving transition probabilities are zero). The **aperiodicity** of the above state machine can be deduced from three different logic,1. The two states have self-transitions (periodicity is 1). 1 is co-prime to every integer and thus the states are aperiodic. Thus the chain is aperiodic.2. The state machine is irreducible. At least one state has a self-transition. Thus the chain is aperiodic.3. The chain is aperiodic if and only if there exists a positive integer *n* such that all elements of the matrix A^n are strictly positive. A has strictly positive elements. Thus the chain is aperiodic.

Stationary distributions

Initial distribution $\pi = [\pi_1, \pi_2, \pi_3, \ldots, \pi_k, \ldots]$ is a stationary distribution if $\pi A = \pi$. Then π_i describes what proportion of time that is spent in state *i* in the long run. A model may not have a stationary distribution or maybe more than one. For an irreducible Markov chain has a stationary distribution it is unique. If the state space is finite and the chain is irreducible, then a unique stationary distribution exists.

In our case, let $\pi = [\pi_1, \pi_2]$. Then $\pi A = \pi$ gives, $0.995\pi_1 + 0.01\pi_2 = \pi_1$ $0.005\pi_1 + 0.99\pi_2 = \pi_2$Solving these with $\pi_1 + \pi_2 = 1$ gives $\pi_1 = 2/3$ and $\pi_2 = 1/3$.Hence, the chain **has a stationary distribution** and since the chain is irreducible it is unique.

A more refined HMM for gene annotation

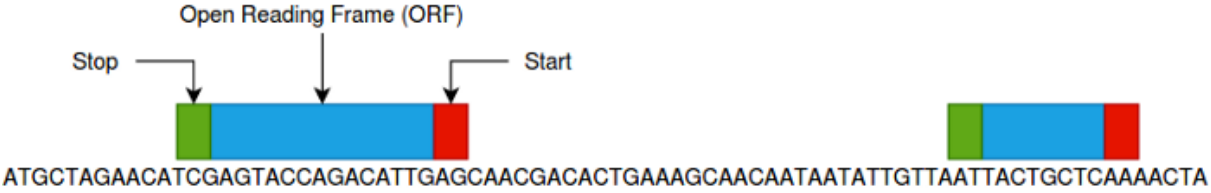
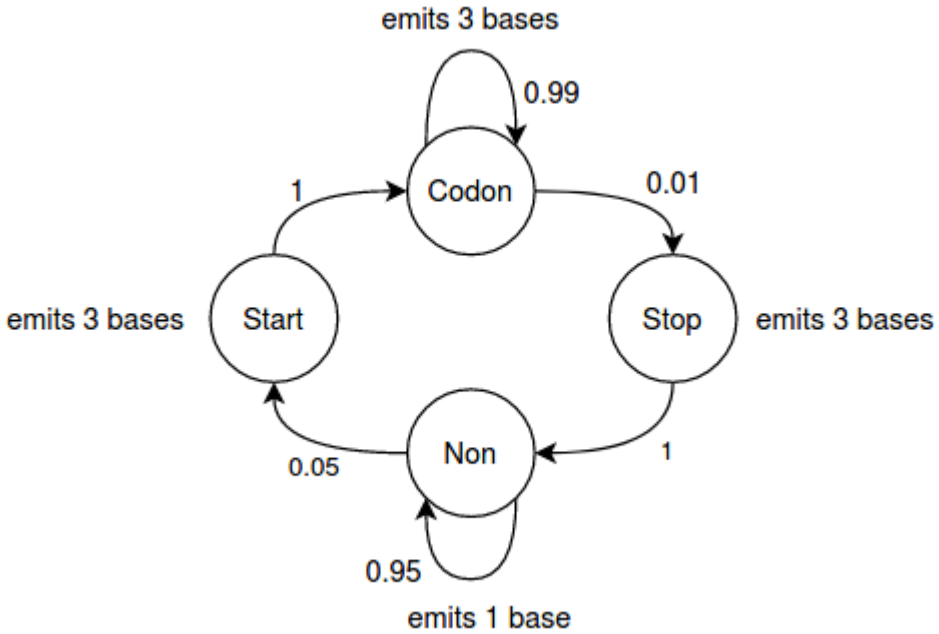


Fig 1 0— An ORF is a continuous stretch of codons that begins with a start codon and ends at a stop codon.

In this scenario we are interested in finding four hidden states —start codon, ORF, stop codon, and the non-coding region. With a similar argument as mentioned earlier we can use Markov property to model this problem.



Model 2 — A codon is a sub-string of 3 bases. Hence Start codon and Stop codon states emit 3 bases in contrast to the first example where each state emitted a single base. Non-State emits 1 base and has a self-transition to make sure the model supports arbitrary lengths between genes (instead of emitting 3 bases). ORF (Codon state) is a multiplication of 3 thus, emits 3 bases. Self-transition in the Codon state ensures varying lengths of ORFs. Based on the current state we are in, the length of the observed sub-sequence vary, e.g., if we are in Non

state we only consider 1 base as the observation at that state. In the same manner, the emission probability space expands to include emission probabilities for groups of 3 bases. The transition probability values were chosen arbitrarily.

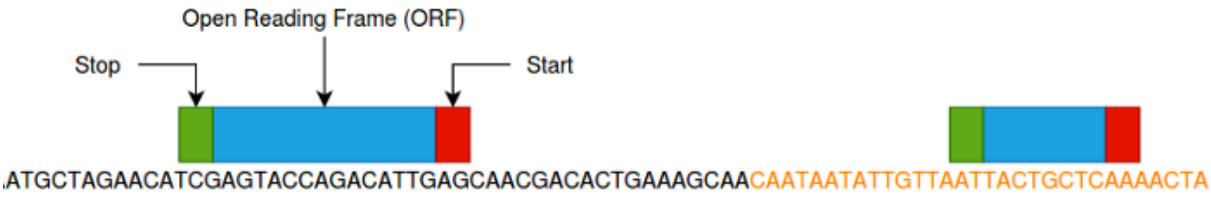
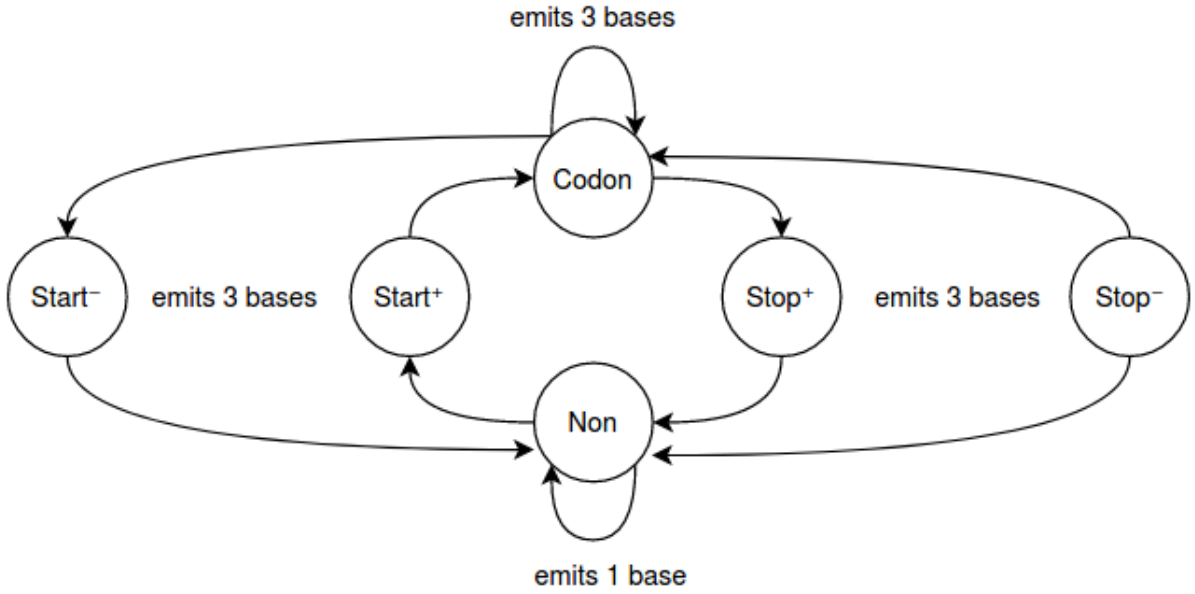


Fig 1 1— The portion of the DNA bases in orange represents the reverse strand. When genes on the reverse strand are considered, technically the stop codons are met before ORFs.



Model 3— Now that we consider the reverse strand two new states are introduced. The reason we cannot have a transition from the original Stop state to Codon state is biologically the $Stop^+$ state and $Stop^-$ states are different. They emit codons with different emission probabilities. The same argument is valid for having $Start^+$ and $Start^-$ states

In both above Markov processes all the states communicate with each other. Thus the Markov chain is **irreducible** and there exists one equivalence class.The state machine has **zero absorbing states.**Two states have self-transitions. Hence the states are recurrent - the class is **recurrent.**Using the same argument we can deduce that the chain is **aperiodic.**

Let's assume the **Model 2** has the following transition matrix A,

```
            Codon     Stop     Start     Non
A = Codon [0.99      0.01      0          0]
    Stop  [0         0         0          1]
    Start [1         0         0          0]
    Non   [0         0         0.05   0.95]
```

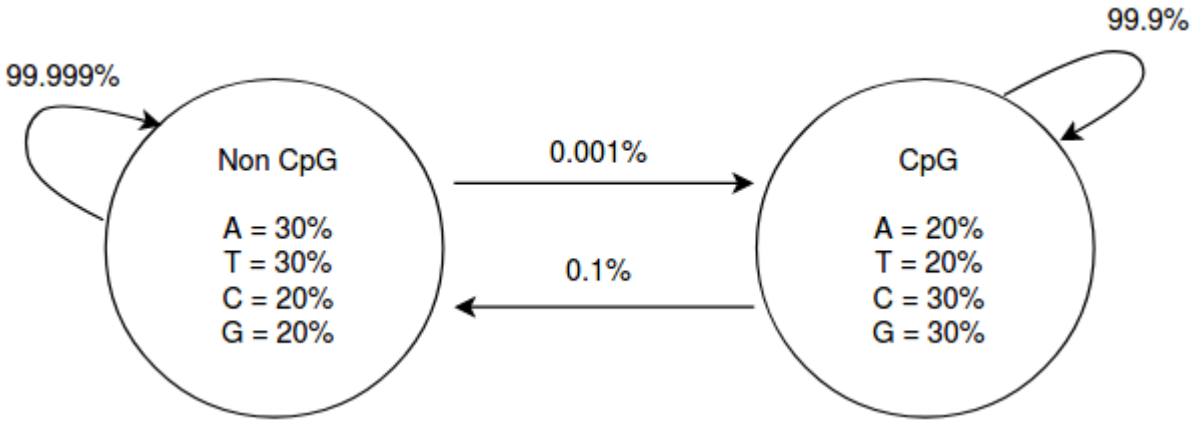
Then we can try to find a stationary distribution s.t $\pi A = \pi$ for $\pi = [\pi_1,\pi_2,\pi_3,\pi_4]$

$0.99\pi_1 + \pi_3 = \pi_1$
$0.01\pi_1 = \pi_2$
$0.05\pi_4 = \pi_3$
$\pi_2 + 0.95\pi_4 = \pi_3$ gives $\pi_2/\pi_1 = 0.01$, $\pi_3/\pi_4 = 0.05$ Solving these with $\pi_1+\pi+\pi_3+\pi_4 = 1$ gives,

$\pi_1=1/1.22$, $\pi_2=1/122$, $\pi_3=1/122$ and $\pi_4=10/61$.Hence, the chain **has a stationary distribution** and since the chain is irreducible it is unique.

HMMs in bioinformatics

The same HMM presented in **Model 1** can be used to detect CpG islands in a DNA sequence with little modifications. Simply a CpG island is a region that has a relatively high density of C and G bases. We can argue that the current state is enough to predict the next state in this problem as well.



Model 4 — CpG and Non-CpG are the two hidden states and the arrows mark transitions and their probabilities. Inside each state the observations are described with their emission probabilities at a particular state src-https://scialert.net/fulltext/?doi=jas.2012.1518.1525 Note the emission probabilities of the bases C and G are higher in the CpG state and lower in the Non-CpG state. Thus the model holds the underlying biological aspect.

If we consider the generative idea for **Model 4** the generator would produce arbitrary lengths of CpG regions but on average they will be 999 bases (≈1kb) long. Non-CpG regions will be 99999 bases (≈100kb) long on average.

Viterby algorithm

Recall that we can use Viterby algorithm to determine the best possible still valid hidden state sequence for a given observation sequence.

$$P(S = s_1,....s_n \mid L = l_1,....l_n) = \frac{P(S = s_1,....s_n,\ L = l_1,....l_n)}{P(L = l_1,....l_n)}$$

$$= \frac{P(S = s_1,....s_n)P(L = l_1,....l_n,\ S = s_1,....s_n)}{P(L = l_1,....l_n)}$$

Fig 12—Finding the probability of a hidden sequence S given the observation sequence L can be expressed as the joint probability and then using Baye's rule transformed into three calculatable components. The denominator is the evaluation problem and remains constant during the process of finding the optimal S.

Let's define R_i^s as the optimal hidden state sequence up to i^{th} observation that has the hidden state *s* as the i^{th} state.

example : Find the optimal hidden state sequence S' for the observation sequence L = [ACT]. Denote Non-CpG state by *N* and CpG state by *I*. The initial probabilities are [0.99 0.01] for *N* and *I* respectively. The initial probability makes sense as it is highly likely to start at a Non-CpG region.

Here, we have to calculate R_0^N, R_0^I, R_1^N, R_1^I, R_2^N, R_2^I, R_3^N, and R_3^I. Each R_i^s has to choose the best from the previous R_{i-1}. For example R_1^N has to choose between R_0^N x Prob. of transiting from state *N* to *N* x Prob. of emitting base A at state *N* Vs R_0^I x P(*I* | *N*) x P(A | *N*). In a more compact form,

$$R_1^N = \max(R_0^N \times P(N \mid N) \times P(\text{'A'} \mid N),\ R_0^I \times P(N \mid I) \times P(\text{'A'} \mid N))$$

$$R_0^N = 0.99 \text{ and } R_0^I = 0.01 \text{ (initial probabilities)}$$

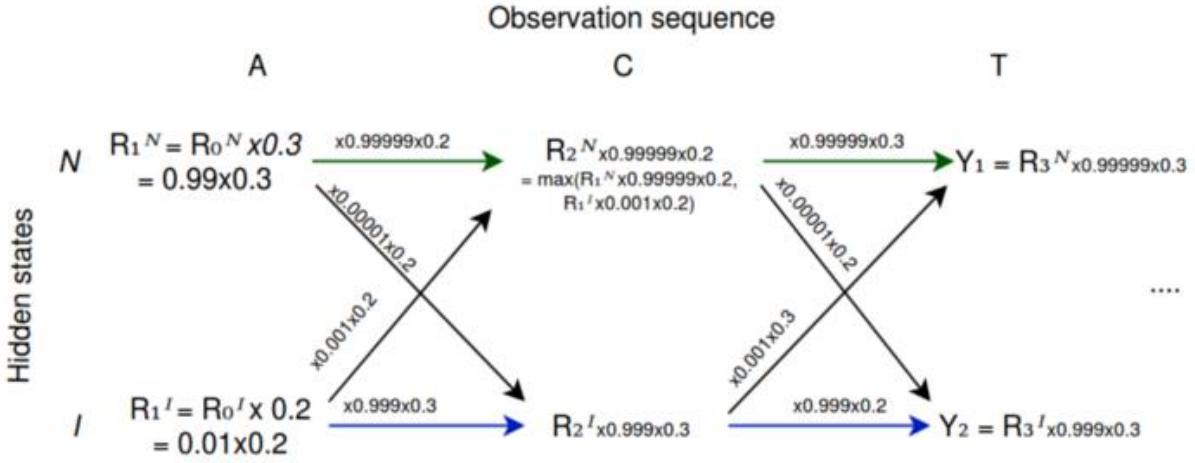


Fig 1 3 — Two hidden states N *and* I; ACT observation sequence. We compare Y_1 and Y_2 *to find max and backtrack to find the optimal hidden state sequence. Generally, the backtracking path could be the green arrows, blue arrows, or a different path. In this case* Y_1 is larger and green arrows build the backtracking path. Hence, the optimal hidden state sequence is NNN.

Why Viterbi algorithm works?

As shown in Fig 13 the algorithm yields *NNN*… or *III*…as the hidden state sequence most of the time. It depends on the initial probability. If the initial probability of being in the state *I* is higher then we will get *III*...

However, we are interested in a mix of *N* and *I* in the hidden state sequence. To yield such a sequence we understand that there should be a moment like R_{i-1}^N x P(*N* | *N*) x P(E_i | *N*) < R_{i-1}^I x P(*N* | *I*) x P(E_i | *N*) where *I*→*N* is favored over *N*→*N*, or a moment like R_{i-1}^I x P(*I* | *I*) x P(E_i | *I*) < R_{i-1}^N x P(*I* | *N*) x P(E_i | *I*) where *N*→*I* is favored over *I*→*I*.

There is a huge penalty involved with *I*→*N* compared to *N*→*N* (≈x10^3 times costly to jump from *I*→*N* than from *N*→*N*). Similarly, it is ≈10^5 times costly to jump from *N*→*I* than from *I*→*I*. But what is fascinating is Viertby works and this kind of jumps occur at the correct moment.

Though it seems impossible the emission probabilities play the role of compensating.

For example consider P('G'|I) = 0.3 and P('G'|N) = 0.2. P('G'|I)/P('G'|N) = 1.5Powers of 1.5

n =	20	40	60	80
$(1.5)^n$ =	$3x10^3$	$1x10^7$	$3x10^{10}$	$1x10^{14}$

Let's say the initial probability of starting at *N* is higher. When the observation sequence has a part like ...A_{60}GCCCG... there is a moment where $R_{i-1}{}^I$ x P(*I*|*I*) x $P(E_i|I)$ < $R_{i-1}{}^N$ x P(*I*|*N*) x $P(E_i|I)$. And at that moment *N*→*I* occurs. It is in intuitively correct as we expect more Gs in a CpG state.**note 1** - 99.999/99.9≈1 hence, we can ignore P(*N*|*N*) and P(*I*|*I*).**note 2** - a cross jump does not only have to pay the penalty of the cross jump itself but also the penalty of the ratio difference between the initial probabilities, e.g.,*I→N transition has to compensate for the initial probability ratio difference* 0.99/0.01 too.

For more information on this watch https://youtu.be/d5NMrA2HkG4?t=4079

The run time for a k-state HMM on a sequence of length L when all states are connected to each other is $O(k^2L)$.

References

1. Gene Finding Using Hidden Markov Model — Mini Review https://scialert.net/fulltext/?doi=jas.2012.1518.1525
2. Classification of States https://www.probabilitycourse.com/chapter11/11_2_4_classification_of_states.php#example11_6
3. https://en.wikipedia.org/wiki/Markov_property
4. Introns vs Exons https://www.youtube.com/watch?v=_asGjfCTLNE
5. Bioinformatics for Evolutionary Biologists by Bernhard Haubold and Angelika Börsch-Haubold
6. Markov and Hidden Markov Models of Genomic and Protein Features https://www.youtube.com/watch?v=d5NMrA2HkG4
7. Events and Random Variables http://www.randomservices.org/random/prob/Events.html