Contents

Chapter 1

A Biology Primer

1.1 The Central Dogma of Molecular Biology

The central dogma of molecular biology explains the flow of genetic information in the cell between between information-carrying biopolymers (DNA, RNA and protein). It states that the transfer of information from nucleic acid to nucleic acid, or from nucleic acid to protein may be possible, but transfer from protein to protein, or from protein to nucleic acid is impossible.

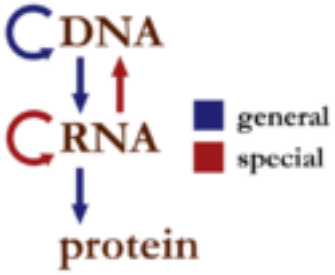


Figure 1.1: Information flows between DNA, RNA and protein. Source: Wikipedia

The genetic code of an organism is stored in DNA, which is converted into portable RNA messages in a process called transcription. These messages travel from the cell nucleus (where the DNA resides) to the ribosomes where they are used as template to make specific proteins in a process called translation. The central dogma states that the pattern of information that occurs most frequently in our cells is:

- From existing DNA to make new DNA (replication)
- From DNA to make new RNA (transcription)
- From RNA to make new proteins (translation).

Besides these, there are some notable possibilities. For instance, retroviruses are able to generate DNA from RNA via reverse-transcription, and some viruses use RNA to make protein. All this is shown in Figure 1.1. The generated proteins carry out most of the cellular functions such as metabolism, DNA regulation, and replication.

1.2 DNA

1.2.1 Function

The DNA molecule stores the genetic information of an organism. DNA contains regions called genes, which encode for the proteins that carry out most of the cellular function. Other regions of the DNA contain

regulatory elements, which partially influence the level of expression of each gene.

1.2.2 Structure

The DNA molecule consists of two strands that wind around to form a shape known as a double helix. Each strand has a backbone made of alternating sugar (deoxyribose) and phosphate groups. Attached to each sugar is one of the four bases: adenine, cytosine, guanine, and thymine, frequently represented using the letters A, C, G, and T respectively. The two strands are held together by bonds between the bases: A and T are connected by two hydrogen bonds, while C and G are connected by three bonds. This specificity in pairing means that one strand can be used as a template to generate the other strand.

The DNA strands also have directionality, which refers to the positions of the pentose ring where the phosphate backbone connects. This directionality convention comes from the fact that DNA and RNA polymerase synthesize in the 5' to 3' direction. The complementary pairing with directionality means that the DNA strands are anti-parallel. In other words the 5' end of one strand is adjacent to the 3' end of the other strand. As a result, DNA can be read both in the 3' to 5' direction and the 5' to 3' direction, and genes and other functional elements can be found in each direction (on either strand). By convention, DNA is written from 5' to 3'.

Base pairing between nucleotides of DNA constitutes its primary and secondary structure. In addition to DNA's secondary structure, there are several extra levels of structure that allow DNA to be tightly compacted and influence gene expression. The tertiary structure describes the twist in the DNA ladder that forms a helical shape. In the quaternary structure, DNA is tightly wound around small proteins called histones. These DNA-histone complexes are further wound into tighter structures seen in chromatin.

1.2.3 Replication

The structure of DNA with its weak hydrogen bonds between the bases in the center allows the strands to easily be separated for the purpose of DNA replication. In the replication of DNA, the two complementary strands are separated, and each of the strands are used as templates for the construction of a new strand. DNA polymerases attach to each of the strands at the origin of replication, reading each existing strand from the 3' to 5' direction and placing complementary bases such that the new strand grows in the 5' to 3' direction. Because the new strand must grow from 5' to 3', one strand (leading strand) can be copied continuously, while the other (lagging strand) grows in fragments that are later pasted together by DNA ligase. The end result is 2 double-stranded pieces of DNA, where each is composed of 1 old strand, and 1 new strand. For this reason, DNA replication is semi-conservative.

1.3 Transcription

1.3.1 mRNA generation

Transcription is the process to produce RNA using a DNA template. The DNA is partially unwound to form a bubble, and RNA polymerase is recruited to the transcription start site (TSS) by regulatory protein complexes. RNA polymerase reads the DNA from the 3' to 5' direction and placing down complementary bases to form messenger RNA (mRNA). RNA uses the same nucleotides as DNA, except Uracil (U) is used instead of Thymine (T).

1.3.2 Post-transcriptional modifications

Messenger RNA (mRNA) in eukaryotes experience post-translational modifications, or processes that edit the mRNA strand further. Most notably, a process called splicing removes introns (intervening regions which don't code for protein), so that only the coding regions (the exons), remain. Different regions of the primary transcript may be spliced out and each can lead to a different protein product. This phenomenon is referred to as alternative splicing. In this way, an large number of protein products can be generated based on different splicing permutations. In addition to splicing, both ends of the mRNA molecule are processed. The

5’ end is capped with a modified guanine nucleotide. At the 3’ end, roughly 250 adenine residues are added to form a poly(A) tail.

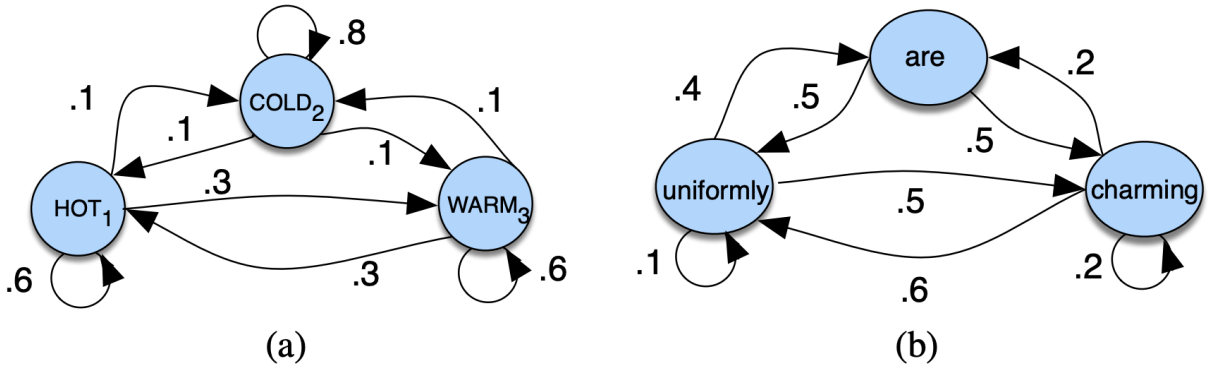
Chapter 2

Hidden Markov Models

2.1 Overview

A **Hidden Markov Model (HMM)** is a type of **Markov model** in which the system being modeled is assumed to be a Markov process. That is, it is a "memoryless" system whose trajectory is solely determined by its current state. The HMM is considered "hidden" because we do not (or can not) know about the states of the variable being observed (say, X). Hence, we attempt to learn about X by observing Y, some sort of observation/event that occurs due to the hidden states. Like all Markov processes, HMM has an additional requirement that the outcome of Y at time $t = t_0$ may be "influenced" **only** by the outcome of X at $t = t_0$ and that the outcomes of X and Y at $t = t_0$ must **not** affect the outcome of Y at $t = t_0$. I.e. the states before the current state have no impact on the future except via the current state. It's as if to predict tomorrow's weather you could examine today's weather but you weren't allowed to look at yesterday's weather!

Here is an example of a 3-state **Markov model**:



As we move from state to state (*node to node* or *circle to circle*), there is a **weight** associated with each edge, indicating the probability that we move from one node to another.

A Markov chain is useful when we need to compute a probability for a sequence of observable events. In many cases, however, the events we are interested in are hidden: we don't observe them directly. For example, we don't normally observe part-of-speech tags in a text. Rather, we see words and must infer the tags from the word sequence. We call the tags hidden because they are not observed.

HMM's have applications in all sorts of areas including **thermodynamics, economics, speech, pattern recognition, bioinformatics,** and more. They provide a foundation for probabilistic models of linear sequence 'labeling' problems.

2.2 Mathematical Definition(s)

Mathematically, if we consider a sequence of state variables $q_1, q_2, \ldots, q_i$ then the **markov assumption** is as follows:

$$P(q_i = a|q_1, q_2, \ldots, q_{i-1}) = P(q_i = a|q_{i-1})$$

The values of weights (or probabilities) associated with each edge coming off of a state (or node) must sum up to 1. A Markov model has a set of states:

$$S = \{s_1, s_2, s_3, \ldots s_n\}$$

The **Markov process** moves from one state to another generating a sequence of states:

$$s_{i1}, s_{i2}, s_{i3}, \ldots s_{ik} \ldots$$

The following need to be defined for a **Markov model**: 1. **Transition probabilities:**

$$A = (a_{ij}), a_{ij} = P(s_i, s_j)$$

2. **Initial Probabilities (π):**

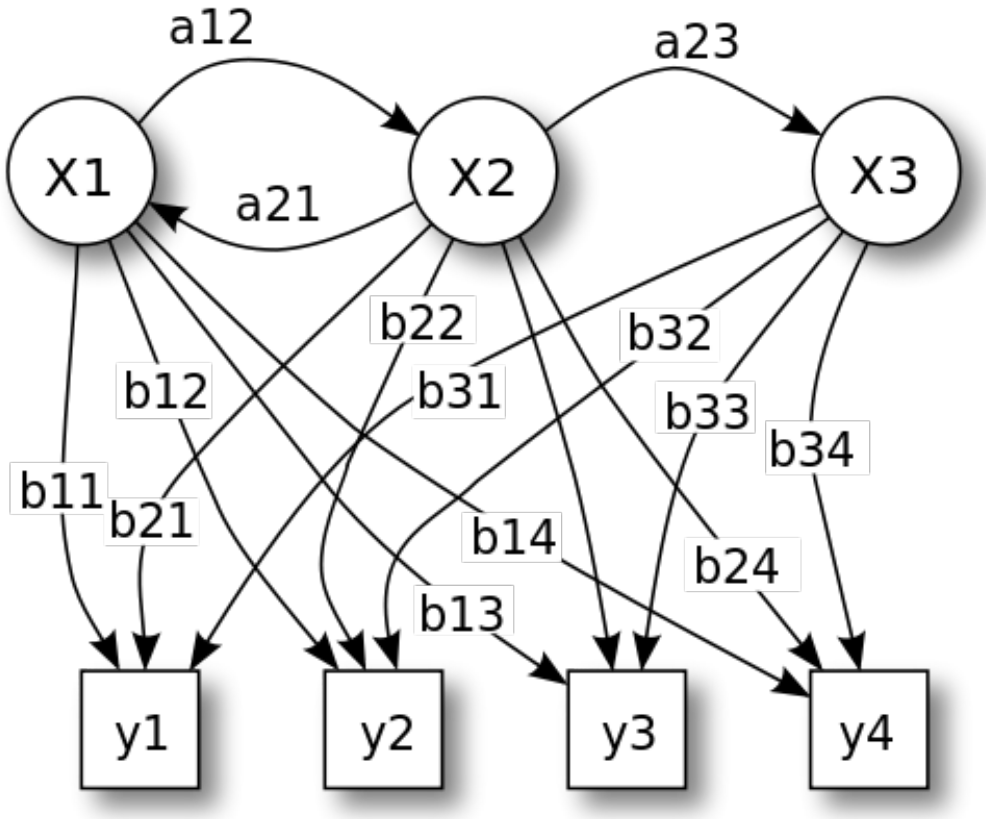
$$\pi = \{P(s_1), P(s_2), \ldots, P(s_i)\}$$

A **hidden** Markov model requires one more mathematical definition. We need to know the probability of observing an event **given** a state:

$$B = (b_i(v_m)), b_i(v_m) = P(v_m|s_i)$$

These are known as **emission probabilities**. The probability that given a state, we "emit" to a certain observation.

Given the above, we can alter the graph model above to represent a **hidden** Markov model:



2.3 An Example

The following example problem is pulled from wikipedia:

Consider two friends, Alice and Bob, who live far apart from each other and who talk together daily over the telephone about what they did that day. Bob is only interested in three activities: walking in the park, shopping, and cleaning his apartment. The choice of what to do is determined exclusively by the weather on a given day. Alice has no definite information about the weather, but she knows general trends. Based on what Bob tells her he did each day, Alice tries to guess what the weather must have been like.

Alice believes that the weather operates as a discrete Markov chain. There are two states, "Rainy" and "Sunny", but she cannot observe them directly, that is, they are *hidden* from her. On each day, there is a certain chance that Bob will perform one of the following activities, depending on the weather: "walk", "shop", or "clean". Since Bob tells Alice about his activities, those are the *observations*. The entire system is that of a hidden Markov model (HMM).

Alice knows the general weather trends in the area, and what Bob likes to do on average. In other words, the parameters of the HMM are known. They can be represented as follows in Python:

```
states = ('Rainy', 'Sunny')

observations = ('walk', 'shop', 'clean')
```

```
start_probability = {'Rainy': 0.6, 'Sunny': 0.4}

transition_probability = {
   'Rainy' : {'Rainy': 0.7, 'Sunny': 0.3},
   'Sunny' : {'Rainy': 0.4, 'Sunny': 0.6},
   }

emission_probability = {
   'Rainy' : {'walk': 0.1, 'shop': 0.4, 'clean': 0.5},
   'Sunny' : {'walk': 0.6, 'shop': 0.3, 'clean': 0.1},
   }
```

In this piece of code, `start_probability` represents Alice's belief about which state the HMM is in when Bob first calls her (all she knows is that it tends to be rainy on average). The particular probability distribution used here is not the equilibrium one, which is (given the transition probabilities) approximately `{'Rainy': 0.57, 'Sunny': 0.43}`. The `transition_probability` represents the change of the weather in the underlying Markov chain. In this example, there is only a 30% chance that tomorrow will be sunny if today is rainy. The `emission_probability` represents how likely Bob is to perform a certain activity on each day. If it is rainy, there is a 50% chance that he is cleaning his apartment; if it is sunny, there is a 60% chance that he is outside for a walk.

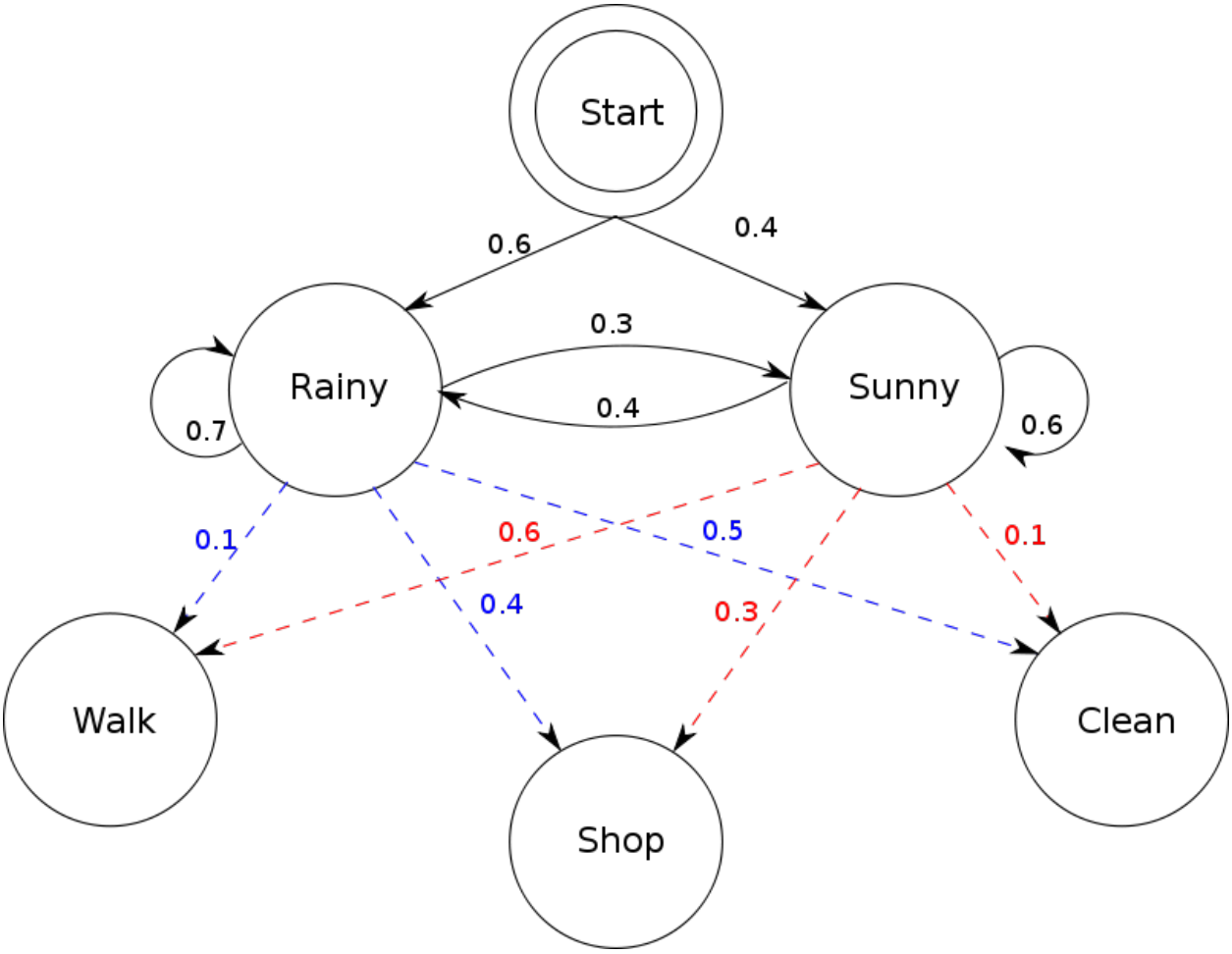


Figure 2.1: Hidden Markov Model to predict weather

2.4 Computational problems with HMMs

There are many computational problems with HMMs. Below are just a few. In general, they involve the use of dynamic programming and gradient descent while solving for the maximum likelihood of a certain sequence of states given observations. Oftentimes, the probabilities in these algorithms are represented in **log space** to make it easier to work with the math while preventing **underflow** errors at the CPU level (numbers way too small for a computer to handle).

2.4.1 Decoding problem :

Given the HMM M=(A,B,π), and an observation sequence O calculate the most likely sequence of states that produced O. This is commonly solved using the Viterbi Algorithmand involves the application of dynamic programming to recurse through a state matrix and for obtaining the maximum *a posteriori* probability estimate of the most likely sequence of hidden states—called the Viterbi path—that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models (HMM)

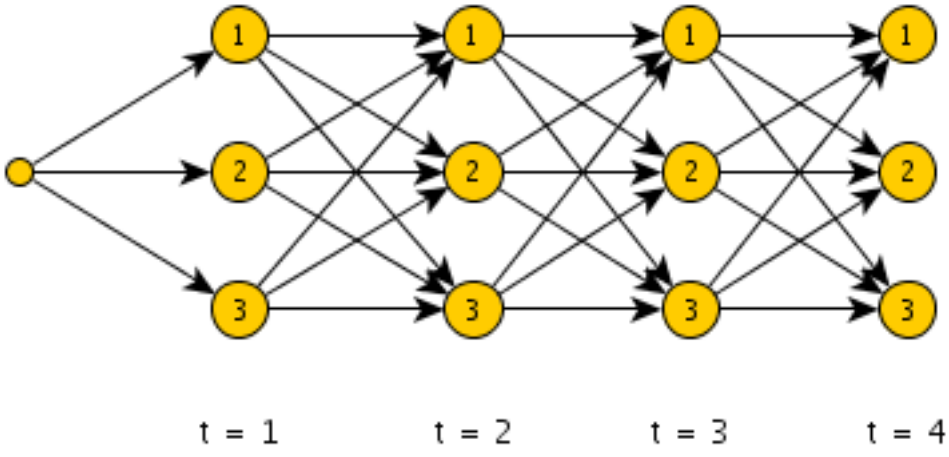


Figure 2.2: Viterbi algorithm

2.4.2 Likelihood Problem:

Similar to the above decoding problem, given the HMM M=(A,B,π), and an observation sequence O,$o_i \in \nu_1, \nu_2, \ldots, \nu_M$ we need to calculate the likelihood P(O|M) using the probabilties of observations given a set of states:

$$P(O|S) = \prod_{i=1}^{T} P(O_i|S_i)$$

However, the state sequence is unknown.

2.4.3 Learning problem:

Given an observation sequence, O, and general structure of HMM, determine HMM parameters that best fit the training data. Here we are solving a sort of reverse problem. That is, we **do not know** the specific probabilities of the transition or emission states. All we know is the overall structure of the model, and using a set of training data, we can fit our model to produce "optimal" values for A, B, and π, such that the model can be applied elsewhere.

The most well-known algorithm for this is the Basum-Welch algorithm, it utilizes a stochastic gradient descent algorithm and thus is not garautneed to be a truly optimal solution (local optima). It can also very computationally complex.

2.5 Conclusions

HMM's offer great prediction and modeling potential in the form of a highly-interpretable and statistically sound model/algorithm. They can be applied to many real-world problems and are often computationally efficient (when making inferences). They still, however, have both pros and cons:

2.5.1 Pros:

- HMM models are highly studied, statistically sound, and highly interpretable models.
- Easy to implement and analyze.
- Incorporates prior knowledge into the model architecture.
- Can be initialized close to something believed to be correct
- Widely applicable

2.5.2 Cons:

- Bounded by the Markov assumption: The next state is only determined byt the current state and not previous ones
- For EM learning problems, the number of parameters to be evaluated is huge. So it needs a large data set for training.
- Training an HMM can often be computationally challenging.