CSI5180. Machine Learning for Bioinformatics Applications

Hidden Markov Models

by

Marcel Turcotte

Version November 6, 2019

Preamble

Preamble

Hidden Markov Models

In this lecture, we focus on learning algorithms suited for **sequence (string) input data**. In particular, we study **Hidden Markov Models**. First, we introduce **Markov processes** and **Markov chains**. Next, using the example of the occasionally dishonest casino, we discern the concept of **hidden variables**. The presentation puts the emphasis on the **graphical nature** of these models. We use the example of a gene finder algorithm as running example.

General objective :

- **Explain** the concepts related to Hidden Markov Models.

Learning objectives

- **Discuss** the properties of a Markovian process
- **Explain** the concept of hidden (latent) variables
- **Describe** Hidden Markov Models
- **Name** the important problems (questions) solved by HMM

Reading:

- Sean R Eddy. What is a hidden Markov model? *Nat Biotechnol* **22**(10):13156, Oct 2004.
- Byung-Jun Yoon. Hidden Markov Models and their applications in biological sequence analysis. *Curr Genomics* **10**(6):40215, Sep 2009.

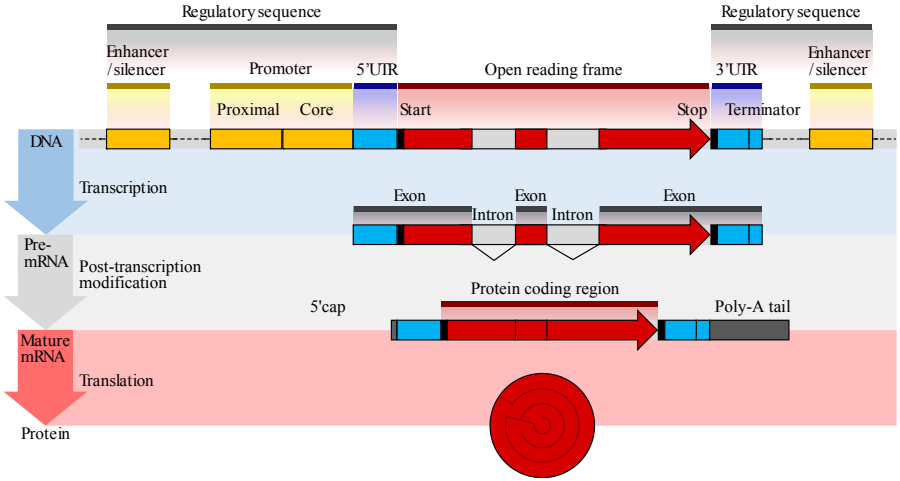
Plan

Problem

Gene structure in eukaryotes - the input

GAATTCTATATAAATAAGTATTAAATTCTGGTTAAAATATAGAAAAAATAGAATTAGATT
CAATGATATCTAATAACATACCAAGGAGTAAAGGACTAATTGAGGATGACTAGTCATTCT
ATAATTGGAAGCACGAATGAGGCTAAAAGAATGATAGTATGTTGTTCGATTCCAAAGGTG
AAAACCAAAGACGGAGAATTCTTATGGAGTCCTGTCTATTTTTATTAACCCTGTGAATTG
AAACATCTTAGTAACAGGAGGAAAAGAAATCAACCGAGATTTTAACGAGTAGTGGCGAGC
GAAAGTAAATGAAAACATTCATGTTTTGATCCGAAATATCTTATCGATGTTTCGATTTTT
TCAAAGACCCCGTACCGGGTCTTGGGGCATGTCTGAAATTGAACATCACACACTTACCCA
TGATAAAGGAGATGGTTTGGATCTTCGATTCTACCATTTTCAGGCAGTGTGTTTATGGAA
TGGGTGGCCAAAGAAGGTGAAAGTCCTGTAAATTTCAGTAGTAGACCACTTATGGAGTAG
AACGAATTTTGTTCAGAAGAAAGGGGTACCATCCTCTAATAAATTAAATATGATAGAATG
AACGATAGTGAAGAGTACCGTGAGGGAAAGTTGAAAAGTACCTCTAAGAGAACGAAGCCT
TCCGAGGCTTCGAATATCAATGCCGGAGGGGTGAAATAGATCCTGAATCAGTTAAGTCTA
AAAGCAGTTTGAGCCAAGATTATGGTGAAGACGTACCTTTTGCATAATGGGTCAGCAAGT
TAATTTTTGGAGCAAGAGAACAAAAGAACGTATCTTTGGTACGTTGGTGATCTAAGTGAA
AACAAAAGAACAAAGTGAGACTTAGTCTTACCCCTATACATAATTTTGCACTCAGTGTGA
CATGGCCAGGTGTAAGACCGA(...52,624,944...)ATCGTAGTAATGCTCTCCGAT
AAGAATCATTGATTCTTCGGACCCACATGGGTACCCATACTCCCCCCAAATGA

Gene structure in eukaryotes



Source: Thomas Shafee (https://commons.wikimedia.org/wiki/File:Gene_structure_eukaryote_2_annotated.svg)

Gene structure

- The **gene structure** comprises several elements, including a **transcription start site**, a **5' untranslated region (5' UTR)**, an **open reading frame (ORF)**, a **3' untranslated region (3' UTR)**.

Gene structure

- The **gene structure** comprises several elements, including a **transcription start site**, a **5' untranslated region (5' UTR)**, an **open reading frame (ORF)**, a **3' untranslated region (3' UTR)**.
- Upstream of the gene, there is a regulatory sequence comprising **enhancers**, **silencers**, and a **promotor**.

Gene structure

- The **gene structure** comprises several elements, including a **transcription start site**, a **5' untranslated region (5' UTR)**, an **open reading frame (ORF)**, a **3' untranslated region (3' UTR)**.
- Upstream of the gene, there is a regulatory sequence comprising **enhancers**, **silencers**, and a **promotor**.
- In **eukaryotes**, genes are made coding segments, called **exons**, and non-coding segments, called **introns**, that need to be removed (spliced) prior to translation.

Gene finding/prediction

- **Problem:** identifying the segments of DNA that are making up protein-coding genes.

Gene finding/prediction

- **Problem:** identifying the segments of DNA that are making up protein-coding genes.
- This can be seen as **segmentation** and **labelling** of the DNA sequence.

Gene finding/prediction

- **Problem:** identifying the segments of DNA that are making up protein-coding genes.
- This can be seen as **segmentation** and **labelling** of the DNA sequence.
- A **Hidden Markov Model** allows representing and integrate these elements into one model. Furthermore, these models have been shown to be effective.

Gene finding/prediction

- **GENSCAN**
  - C Burge and S Karlin. Prediction of complete gene structures in human genomic DNA. *J Mol Biol* **268**(1):7894, Apr 1997.
- **GENIE**
  - Kulp, D., Haussler, D., Reese, M. G. & Eeckman, F. H. A generalized hidden Markov model for the recognition of human genes in DNA. *ISMB International Conference on Intelligent Systems for Molecular Biology* **4**, 134142 (1996).
- **HMMGENE**
  - Krogh, A. Two methods for improving performance of an HMM and their application for gene finding. *ISMB International Conference on Intelligent Systems for Molecular Biology* **5**, 179186 (1997).

Other applications

Other applications include:

1. Modelling **pairwise** and **multiple sequence alignments**
2. **Protein secondary structure** prediction
3. Modelling **transmembrane proteins**

Background information

Remarks

Our presentation will be informal. An entire course could be taught on **Markov chains** and **stochastic processes**.

- **MAT 4374 Modern Computational Statistics**
  Simulation including the rejection method and importance sampling; applications to Monte Carlo Markov chains. Resampling methods such as the bootstrap and jackknife, with applications. Smoothing methods in curve estimation.
- **MAT 5198 Stochastic Models**
  Markov systems, stochastic networks, queuing networks, spatial processes, approximation methods in stochastic processes and queuing theory. Applications to the modelling and analysis of computer-communications systems and other distributed networks.

Markov chains

- Like **finite state automata** (FSA):

Markov chains

- Like **finite state automata** (FSA):
  - Finite Markov chains allow to model processes which can be represented by a **finite number of states**.

Markov chains

- Like **finite state automata** (FSA):
  - Finite Markov chains allow to model processes which can be represented by a **finite number of states**.
  - A **process can be in any of these states at a given time**; for some **discrete units of time** $t = 0, 1, 2, \ldots$.

Markov chains

- Like **finite state automata** (FSA):
  - Finite Markov chains allow to model processes which can be represented by a **finite number of states**.
  - A **process can be in any of these states at a given time**; for some **discrete units of time** $t = 0, 1, 2, \ldots$.
  - E.g. the type of nucleotide at a given position t in a sequence.

Markov chains

- Unlike FSAs:

Markov chains

- Unlike FSAs:
  - The **transitions from one state to another are stochastic** (not deterministic).

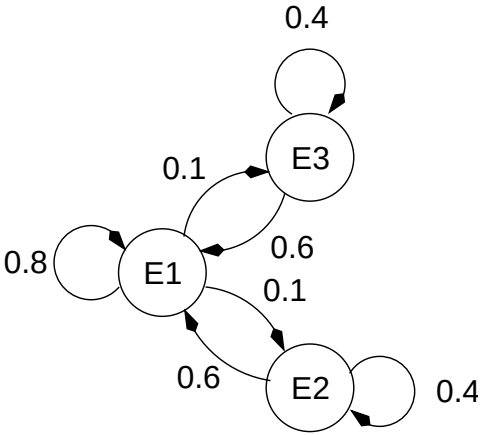
Markov chains

- Unlike FSAs:
  - The **transitions from one state to another are stochastic** (not deterministic).
  - If the current state of the process at time t is E_i then at time $t+1$ either the process stays in E_i or move to E_j, for some j, according to a well-defined probability.

Markov chains

- Unlike FSAs:
  - The **transitions from one state to another are stochastic** (not deterministic).
  - If the current state of the process at time t is E_i then at time $t+1$ either the process stays in E_i or move to E_j, for some j, according to a well-defined probability.
  - E.g. at time $t+1$ the amino acid type for a given sequence position either stays the same of is substituted by one of the remaining 19 amino acid types, according to a well-defined probability, to be estimated.

Markov chains



Properties

A (first order) **Markovian process** must conform to the following 2 properties:

1. **Memoryless**. If a process is in state E_i at time t then the probability that it will be in state E_j at time $t+1$ only depends on E_i (and not on the previous states visited at time $t' < t$, no history). This is called a first-order Markovian process.

Properties

A (first order) **Markovian process** must conform to the following 2 properties:

1. **Memoryless**. If a process is in state E_i at time t then the probability that it will be in state E_j at time $t+1$ only depends on E_i (and not on the previous states visited at time $t' < t$, no history). This is called a first-order Markovian process.
2. **Homogeneity of time**. If a process is in state E_i at time t then the probability that it will be in state E_j at time $t+1$ is independent of t.

Properties

A (first order) **Markovian process** must conform to the following 2 properties:

1. **Memoryless**. If a process is in state E_i at time t then the probability that it will be in state E_j at time $t+1$ only depends on E_i (and not on the previous states visited at time $t' < t$, no history). This is called a first-order Markovian process.
2. **Homogeneity of time**. If a process is in state E_i at time t then the probability that it will be in state E_j at time $t+1$ is independent of t.

Properties

A (first order) **Markovian process** must conform to the following 2 properties:

1. **Memoryless**. If a process is in state E_i at time t then the probability that it will be in state E_j at time $t+1$ only depends on E_i (and not on the previous states visited at time $t' < t$, no history). This is called a first-order Markovian process.
2. **Homogeneity of time**. If a process is in state E_i at time t then the probability that it will be in state E_j at time $t+1$ is independent of t.

GAATTCTATATAAATAAGTATTAAATTCTGGTTAAAATATAGAAAAAATAGAATTAGATT

$$P(X_{t+1} = A | X_t = T)$$

Markov perperty, chain, process

- A **Markov chain** is a **stochastic (probabilistic) model** describing a **sequence** of events.

Markov perperty, chain, process

- A **Markov chain** is a **stochastic (probabilistic) model** describing a **sequence** of events.
- Herein, we focus on **discrete-time homogeneous finite Markov chain models**.

Markov chain

A (first order) **Markov chain** is a sequence of random variables

$$X_0, \ldots, X_{t-1}, X_t$$

that satisfies the following property

$$P(X_t = x_t | X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, \ldots, X_0 = x_0) = P(X_t = x_t | X_{t-1} = x_{t-1})$$

Markov chain

More generally, a *m*-**order** Markov chain is a sequence of random variables:

$$X_0, \ldots, X_{t-1}, X_t$$

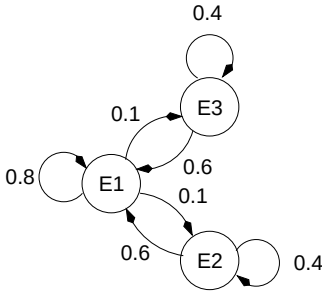
that satisfies the following property

$$P(X_t = x_t | X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, \ldots, X_0 = x_0)$$
$$= P(X_t = x_t | X_{t-1} = x_{t-1}, \ldots, X_{t-m} = x_m)$$

Markov chain models are denoted Mm, where m is the order of the model, e.g. $M0$, $M1$, $M2$, $M3$, etc. A 0-order model is known as a **Bernouilli model**.

Transition probabilities

The **transition probabilities**, p_{ij}, can be represented graphically,



or as a **transition probability matrix**,

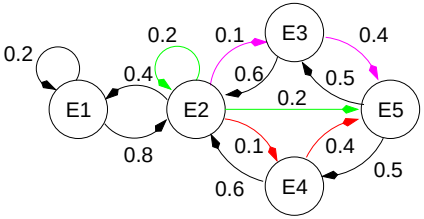
$$P = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.6 & 0.4 & 0.0 \\ 0.6 & 0.0 & 0.4 \end{bmatrix}$$

Transition probabilities

$$P = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.6 & 0.4 & 0.0 \\ 0.6 & 0.0 & 0.4 \end{bmatrix}$$

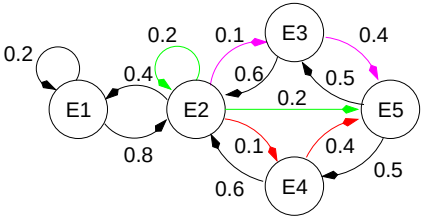
- where p_{ij} is understood as the probability of a transition from state i (row) to state j (column).
- The values in a row represent all the transitions from state i, i.e. all outgoing arcs, and therefore their **sum must be 1**.

Transition probabilities



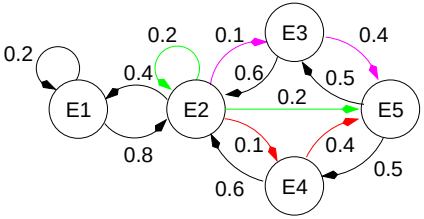
- The framework allows answering elegantly questions such as this one, "**a Markovian random variable is in state E_i at time t, what is the probability that it will be in state E_j at $t + 2$?**"

Transition probabilities



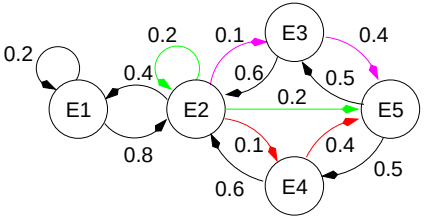
- The framework allows answering elegantly questions such as this one, "**a Markovian random variable is in state E_i at time t, what is the probability that it will be in state E_j at $t+2$?**"
- For the Markovian process graphically depicted above, knowing that a random variable is in state E_2 at time t **what is the probability that it will be in state E_5 at** $t+2$, i.e. after two transitions?

Transition probabilities



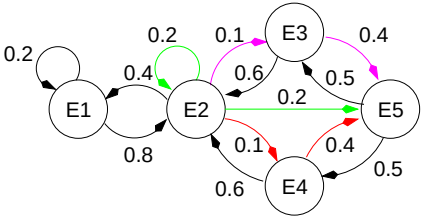
- There are exactly **3 paths of length 2** leading from E_2 to E_5: (E_2, E_2, E_5), (E_2, E_3, E_5) and (E_2, E_4, E_5).

Transition probabilities



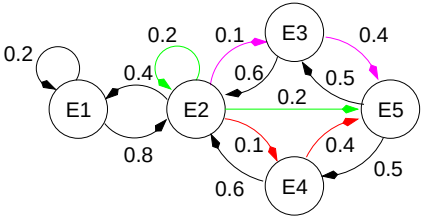
- There are exactly **3 paths of length 2** leading from E_2 to E_5: (E_2, E_2, E_5), (E_2, E_3, E_5) and (E_2, E_4, E_5).
  - The probability that (E_2, E_2, E_5) is followed is $0.2 \times 0.2 = 0.04$

Transition probabilities



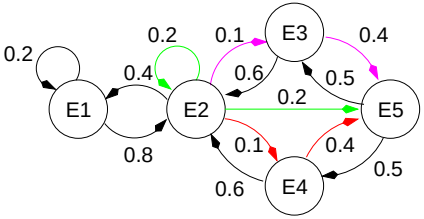
- There are exactly **3 paths of length 2** leading from E_2 to E_5: (E_2, E_2, E_5), (E_2, E_3, E_5) and (E_2, E_4, E_5).
  - The probability that (E_2, E_2, E_5) is followed is $0.2 \times 0.2 = 0.04$
  - The probability that (E_2, E_3, E_5) is followed is $0.1 \times 0.4 = 0.04$

Transition probabilities

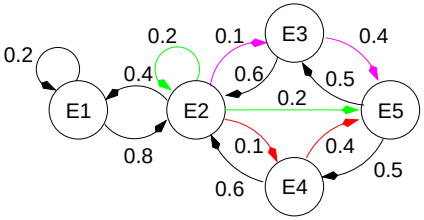


- There are exactly **3 paths of length 2** leading from E_2 to E_5: (E_2, E_2, E_5), (E_2, E_3, E_5) and (E_2, E_4, E_5).
  - The probability that (E_2, E_2, E_5) is followed is $0.2 \times 0.2 = 0.04$
  - The probability that (E_2, E_3, E_5) is followed is $0.1 \times 0.4 = 0.04$
  - The probability that (E_2, E_4, E_5) is followed is $0.1 \times 0.4 = 0.04$

Transition probabilities



- There are exactly **3 paths of length 2** leading from E_2 to E_5: (E_2, E_2, E_5), (E_2, E_3, E_5) and (E_2, E_4, E_5).
  - The probability that (E_2, E_2, E_5) is followed is $0.2 \times 0.2 = 0.04$
  - The probability that (E_2, E_3, E_5) is followed is $0.1 \times 0.4 = 0.04$
  - The probability that (E_2, E_4, E_5) is followed is $0.1 \times 0.4 = 0.04$
  - Therefore, the probability that the random variable is found in E_5 at $t + 2$ knowing that it was in E_2 at t is $0.04 + 0.04 + 0.04 = 0.12$.



- **In general**, the probability that a random variable is found in state E_j at $t + 2$ knowing that it was in E_i at t is,

$$p_{ij}^{(2)} = \sum_k p_{ik} p_{kj}$$

Gene finding

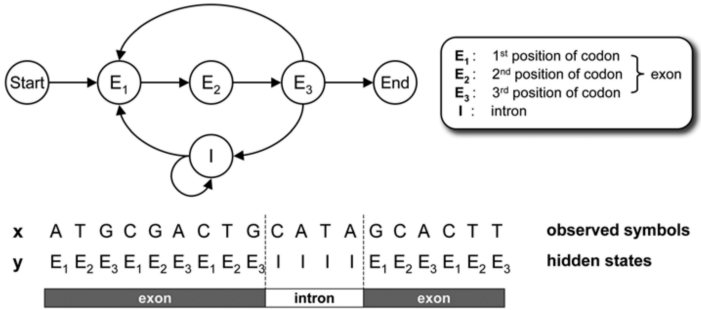
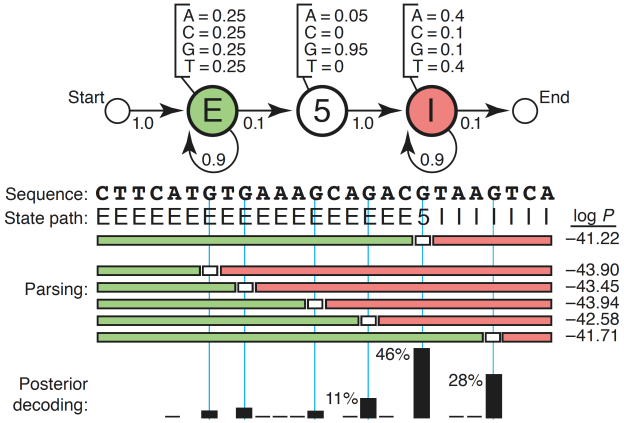


Fig. (1). A simple HMM for modeling eukaryotic genes.

Source: [2] Figure 1

Gene finding



Source: [1] Figure 1

Hidden (latent) variables

What is **hidden**?

Dishonest casino

A **simplified example** will help better understand **hidden variables** and the characteristics of HMMs.

Dishonest casino

A **simplified example** will help better understand **hidden variables** and the characteristics of HMMs.

- **I want to play a game**.

Dishonest casino

A **simplified example** will help better understand **hidden variables** and the characteristics of HMMs.

- **I want to play a game**.
- I will be **tossing a coin** n times.

Dishonest casino

A **simplified example** will help better understand **hidden variables** and the characteristics of HMMs.

- **I want to play a game**.
- I will be **tossing a coin** n times.
- This information can be represented as follows: $\{$ H, T, T, H, T, T, $\ldots\}$ or $\{$ 0, 1, 1, 0, 1, 1, $\ldots\}$.

Dishonest casino

A **simplified example** will help better understand **hidden variables** and the characteristics of HMMs.

- **I want to play a game**.
- I will be **tossing a coin** n times.
- This information can be represented as follows: $\{$ H, T, T, H, T, T, $\ldots\}$ or $\{$ 0, 1, 1, 0, 1, 1, $\ldots\}$.
- In fact, I will be using **two coins**!

Dishonest casino

A **simplified example** will help better understand **hidden variables** and the characteristics of HMMs.

- **I want to play a game**.
- I will be **tossing a coin** n times.
- This information can be represented as follows: $\{$ H, T, T, H, T, T, $\ldots\}$ or $\{$ 0, 1, 1, 0, 1, 1, $\ldots\}$.
- In fact, I will be using **two coins**!
  - One coin is **fair**, i.e. **head** and **tail** are **equiprobable** outcomes,

Dishonest casino

A **simplified example** will help better understand **hidden variables** and the characteristics of HMMs.

- **I want to play a game**.
- I will be **tossing a coin** n times.
- This information can be represented as follows: $\{$ H, T, T, H, T, T, $\ldots\}$ or $\{$ 0, 1, 1, 0, 1, 1, $\ldots\}$.
- In fact, I will be using **two coins**!
  - One coin is **fair**, i.e. **head** and **tail** are **equiprobable** outcomes,
  - but the other one is **loaded** (biased), it returns **head** with probability $\frac{1}{4}$ and tail with probability $\frac{3}{4}$.

Dishonest casino

- If I were using the **same coin** for the duration of the game, then it would be easy for you to guess **which coin I am using**.

Dishonest casino

- If I were using the **same coin** for the duration of the game, then it would be easy for you to guess **which coin I am using**.
- For instance, we could look at the **odds ratio**:

$$P(S|Loaded) = \prod_{i=1}^{L} P(S^{(i)}|Loaded)$$

Dishonest casino

- If I were using the **same coin** for the duration of the game, then it would be easy for you to guess **which coin I am using**.
- For instance, we could look at the **odds ratio**:

$$P(S|Loaded) = \prod_{i=1}^{L} P(S^{(i)}|Loaded)$$

Dishonest casino

- If I were using the **same coin** for the duration of the game, then it would be easy for you to guess **which coin I am using**.
- For instance, we could look at the **odds ratio**:

$$P(S|Loaded) = \prod_{i=1}^{L} P(S^{(i)}|Loaded)$$

$$P(S|Fair) = \prod_{i=1}^{L} P(S^{(i)}|Fair)$$

Dishonest casino

- If I were using the **same coin** for the duration of the game, then it would be easy for you to guess **which coin I am using**.
- For instance, we could look at the **odds ratio**:

$$P(S|Loaded) = \prod_{i=1}^{L} P(S^{(i)}|Loaded)$$

$$P(S|Fair) = \prod_{i=1}^{L} P(S^{(i)}|Fair)$$

$$\frac{P(S|Loaded)}{P(S|Fair)} = \frac{\prod_{i=1}^{L} P(S^{(i)}|Loaded)}{\prod_{i=1}^{L} P(S^{(i)}|Fair)}$$

Dishonest casino

- If I were using the **same coin** for the duration of the game, then it would be easy for you to guess **which coin I am using**.
- For instance, we could look at the **odds ratio**:

$$P(S|Loaded) = \prod_{i=1}^{L} P(S^{(i)}|Loaded)$$

$$P(S|Fair) = \prod_{i=1}^{L} P(S^{(i)}|Fair)$$

$$\frac{P(S|Loaded)}{P(S|Fair)} = \frac{\prod_{i=1}^{L} P(S^{(i)}|Loaded)}{\prod_{i=1}^{L} P(S^{(i)}|Fair)}$$

$$\log(\frac{P(S|Loaded)}{P(S|Fair)}) = \sum_{i=1}^{L} \log(\frac{P(S^{(i)}|Loaded)}{P(S^{(i)}|Fair)})$$

Dishonest casino

- Let's consider a specific sequence: $S = \{ \text{H}, \text{T}, \text{T}, \text{H}, \text{T}, \text{T} \}$

Dishonest casino

- Let's consider a specific sequence: $S = \{$ H, T, T, H, T, T $\}$
- $P(S|Loaded) = \frac{1}{4}^2 \times \frac{3}{4}^4 = 0.01977539062$

Dishonest casino

- Let's consider a specific sequence: $S = \{$ H, T, T, H, T, T $\}$
- $P(S|Loaded) = \frac{1}{4}^2 \times \frac{3}{4}^4 = 0.01977539062$
- $P(S|Fair) = \frac{1}{2}^6 = 0.015625$

Dishonest casino

- Let's consider a specific sequence: $S = \{$ H, T, T, H, T, T $\}$
- $P(S|Loaded) = \frac{1}{4}^2 \times \frac{3}{4}^4 = 0.01977539062$
- $P(S|Fair) = \frac{1}{2}^6 = 0.015625$
- $\frac{P(S|Loaded)}{P(S|Fair)} = 1.265625$

Dishonest casino

- Let's consider a specific sequence: $S = \{$ H, T, T, H, T, T $\}$
- $P(S|Loaded) = \frac{1}{4}^2 \times \frac{3}{4}^4 = 0.01977539062$
- $P(S|Fair) = \frac{1}{2}^6 = 0.015625$
- $\frac{P(S|Loaded)}{P(S|Fair)} = 1.265625$
- $\log(\frac{P(S|Loaded)}{P(S|Fair)}) = 0.1023050449$

Occasionally dishonest casino

- However, I will **not** reveal when I am exchanging the coins.

Occasionally dishonest casino

- However, I will **not** reveal when I am exchanging the coins.
- This information is **hidden** from you.

Occasionally dishonest casino

- However, I will **not** reveal when I am exchanging the coins.
- This information is **hidden** from you.
- **Objective**:

Occasionally dishonest casino

- However, I will **not** reveal when I am exchanging the coins.
- This information is **hidden** from you.
- **Objective**:
  - Looking at a **series of observations**, S, can you predict when the exchanges of coins occurred?

Hidden Markov Model

Hidden Markov Models (HMM)

"A *hidden Markov model* (HMM) is a statistical model that can be used to describe the evolution of observable events [**symbols**] that depend on internal factors [**states**], which are not directly observable."

Hidden Markov Models (HMM)

"A *hidden Markov model* (HMM) is a statistical model that can be used to describe the evolution of observable events [**symbols**] that depend on internal factors [**states**], which are not directly observable."

"An HMM consists of **two stochastic processes** (...)":

- **Invisible** process consisting of states
- **Visible (observable)** process consisting of symbols

Hidden Markov Models (HMM)

"A *hidden Markov model* (HMM) is a statistical model that can be used to describe the evolution of observable events [**symbols**] that depend on internal factors [**states**], which are not directly observable."

"An HMM consists of **two stochastic processes** (...)":

- **Invisible** process consisting of states
- **Visible (observable)** process consisting of symbols

Yoon, B.-J. Hidden Markov Models and their Applications in Biological Sequence Analysis. *Current Genomics* **10**, 402415 (2009).

Definitions

We need to distinguish between the **sequence of states** (π) and the **sequence of symbols** (S).

Definitions

We need to distinguish between the **sequence of states** (π) and the **sequence of symbols** (S).

- The sequence of states, denoted by π and called the **path**, is modelled as a **Markov chain**, these transitions are not directly observable (they are **hidden**),

$$a_{kl} = P(\pi_i = l | \pi_{i-1} = k)$$

where a_{kl} is a **transition probability** from the state k to l.

Definitions

We need to distinguish between the **sequence of states** (π) and the **sequence of symbols** (S).

- The sequence of states, denoted by π and called the **path**, is modelled as a **Markov chain**, these transitions are not directly observable (they are **hidden**),

  $$a_{kl} = P(\pi_i = l | \pi_{i-1} = k)$$

  where a_{kl} is a **transition probability** from the state k to l.

- Each state has **emission** probabilities associated with it:

  $$e_k(b) = P(S(i) = b | \pi_i = k)$$

  the probability of **observing**/emitting the symbol b when in state k.

Definitions

- The **alphabet of emitted symbols**, Σ, the **set of (hidden) states**, Q, a **matrix of transition probabilities**, A, as well as a the **emission probabilities**, E, are the parameters of an HMM: $\mathcal{M} =< \Sigma, Q, A, E >$.

Definitions

- The **alphabet of emitted symbols**, Σ, the **set of (hidden) states**, Q, a **matrix of transition probabilities**, A, as well as a the **emission probabilities**, E, are the parameters of an HMM: $\mathcal{M} =< \Sigma, Q, A, E >$.
- It is often useful, to think of an HMM as a device **generating** sequences.

Definitions

- The **alphabet of emitted symbols**, Σ, the **set of (hidden) states**, Q, a **matrix of transition probabilities**, A, as well as a the **emission probabilities**, E, are the parameters of an HMM: $\mathcal{M} =< \Sigma, Q, A, E >$.
- It is often useful, to think of an HMM as a device **generating** sequences.
  - With some probability the process **stays in the same state** or moves to the **next state**;

Definitions

- The **alphabet of emitted symbols**, Σ, the **set of (hidden) states**, Q, a **matrix of transition probabilities**, A, as well as a the **emission probabilities**, E, are the parameters of an HMM: $\mathcal{M} =< \Sigma, Q, A, E >$.
- It is often useful, to think of an HMM as a device **generating** sequences.
  - With some probability the process **stays in the same state** or moves to the **next state**;
  - At each step, the process **emits a symbol** according to a well defined probability distribution;

Definitions

- The **alphabet of emitted symbols**, Σ, the **set of (hidden) states**, Q, a **matrix of transition probabilities**, A, as well as a the **emission probabilities**, E, are the parameters of an HMM: $\mathcal{M} =< \Sigma, Q, A, E >$.
- It is often useful, to think of an HMM as a device **generating** sequences.
  - With some probability the process **stays in the same state** or moves to the **next state**;
  - At each step, the process **emits a symbol** according to a well defined probability distribution;
  - When looking at a sequence of observable symbols, the observer is **wondering if the sequence could have been produced or not by the model**.

Definitions

- The **alphabet of emitted symbols**, Σ, the **set of (hidden) states**, Q, a **matrix of transition probabilities**, A, as well as a the **emission probabilities**, E, are the parameters of an HMM: $\mathcal{M} =< \Sigma, Q, A, E >$.
- It is often useful, to think of an HMM as a device **generating** sequences.
  - With some probability the process **stays in the same state** or moves to the **next state**;
  - At each step, the process **emits a symbol** according to a well defined probability distribution;
  - When looking at a sequence of observable symbols, the observer is **wondering if the sequence could have been produced or not by the model**.
- Remembering our discussion about **finite state automata**, an **HMM** is equivalent to a **stochastic regular grammar**.

Problems

1. $P(S, \pi)$: the **joint probability** of a **sequence of symbols** S and a **sequence of states** π.

Problems

1. $P(S, \pi)$: the **joint probability** of a **sequence of symbols** S and a **sequence of states** π.
   - The **decoding problem** consists of finding a path π such that $P(S, \pi)$ is maximum;

Problems

1. $P(S, \pi)$: the **joint probability** of a **sequence of symbols** S and a **sequence of states** π.
   - The **decoding problem** consists of finding a path π such that $P(S, \pi)$ is maximum;
2. $P(S|\theta)$: the **probability of a sequence of symbols** S given the **model** θ.

Problems

1. $P(S, \pi)$: the **joint probability** of a **sequence of symbols** S and a **sequence of states** π.
   - The **decoding problem** consists of finding a path π such that $P(S, \pi)$ is maximum;
2. $P(S|\theta)$: the **probability of a sequence of symbols** S given the **model** θ.
   - It represents the likelihood that sequence S has been produced by this HMM, let's call this the **likelihood problem**;

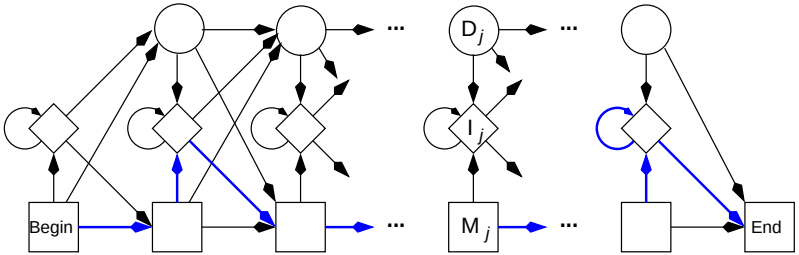
Problems

1. $P(S, \pi)$: the **joint probability** of a **sequence of symbols** S and a **sequence of states** π.
   - The **decoding problem** consists of finding a path π such that $P(S, \pi)$ is maximum;
2. $P(S|\theta)$: the **probability of a sequence of symbols** S given the **model** θ.
   - It represents the likelihood that sequence S has been produced by this HMM, let's call this the **likelihood problem**;
3. Finally, how are the **parameters** of the model (HMM), θ, **learnt**?

Problems

1. $P(S, \pi)$: the **joint probability** of a **sequence of symbols** S and a **sequence of states** π.
   - The **decoding problem** consists of finding a path π such that $P(S, \pi)$ is maximum;
2. $P(S|\theta)$: the **probability of a sequence of symbols** S given the **model** θ.
   - It represents the likelihood that sequence S has been produced by this HMM, let's call this the **likelihood problem**;
3. Finally, how are the **parameters** of the model (HMM), θ, **learnt**?
   - Let's call this the **parameter estimation problem**.

Definitions



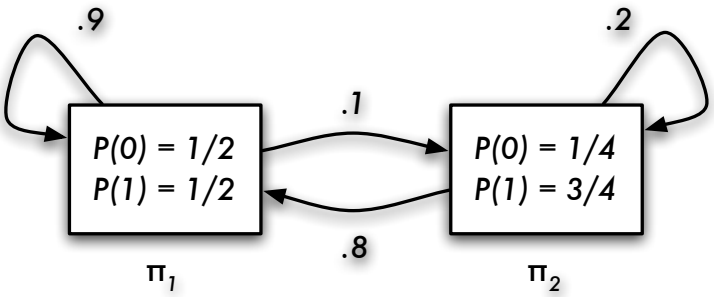
Joint probability of a **sequence of symbols** S and a **sequence of states** π:

$$P(S, \pi) = a_{0\pi_1} \prod_{i=1}^{L} e_{\pi_i}(S(i)) a_{\pi_i \pi_{i+1}}$$

$$P(S = \mathrm{VGPGGAHA}, \pi = \mathrm{BEG}, M_1, M_2, I_3, I_3, I_3, M_3, M_4, M_5, \mathrm{END})$$

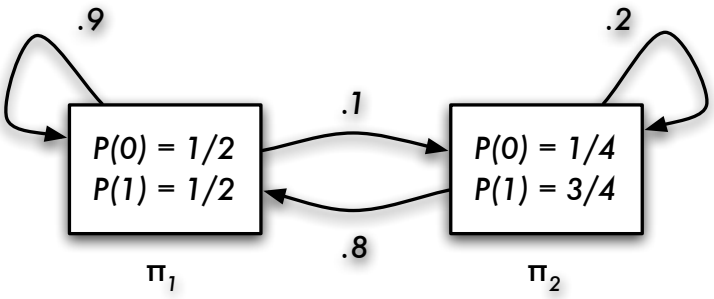
$\Rightarrow$ However in practice, the state sequence π is not known in advance.

Occasionally dishonest casino



- Modelled using an **HMM**, where **each state represents a coin**, with its own **emission probability distribution**, and the **transition probabilities** represent exchanging the coins.

Worked example

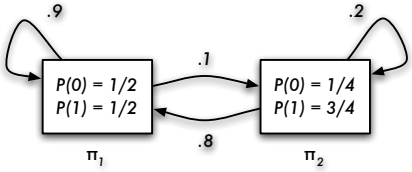


- Given an **input sequence of symbols** (heads and tails), such as 0, 1, 1, 0, 1, 1, 1, which **sequence of states** has the highest probability?

Worked example

- **Which** path leads to the highest joint probability?

S	0	1	1	0	1	1	1
π	π_1	π_1	π_1	π_1	π_1	π_1	π_1
π	π_1	π_1	π_1	π_1	π_1	π_1	π_2
...							
π	π_2	π_2	π_1	π_1	π_2	π_2	π_2
...							
π	π_2	π_2	π_2	π_2	π_2	π_2	π_2



Brute-force

- Since the game consists of printing the series of switches from one coin to the other, selecting the path with the **highest joint probability**, $P(S, \pi)$, seems appropriate.

Brute-force

- Since the game consists of printing the series of switches from one coin to the other, selecting the path with the **highest joint probability**, $P(S, \pi)$, seems appropriate.
- Here, there are $2^7 = 128$ possible paths, **enumerating all** of them is feasible.

Brute-force

- Since the game consists of printing the series of switches from one coin to the other, selecting the path with the **highest joint probability**, $P(S, \pi)$, seems appropriate.
- Here, there are $2^7 = 128$ possible paths, **enumerating all** of them is feasible.
- However, the number of states and consequently the number of possible paths are generally much larger: $\mathcal{O}(M^L)$, where M is the number of states and L is the length of the sequence of symbols.

Decoding problem

- Given an observed sequence of symbols, S, **the decoding problem** consists of finding a sequence of states, π, such that the joint probability of S and π is maximum.

$$\text{argmax}_{\pi} \; P(S, \pi)$$

Decoding problem

- Given an observed sequence of symbols, S, **the decoding problem** consists of finding a sequence of states, π, such that the joint probability of S and π is maximum.

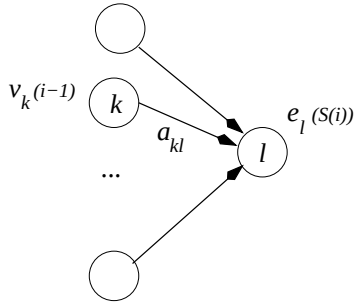
$$\text{argmax}_{\pi} \; P(S, \pi)$$

- For our game, the sequence of states is of interest because it serves to predict the exchanges of coins.

Decoding problem — Viterbi

- The **most probable path** can be found **recursively**. The score for the most probable path ending in state l with observation i, noted $v_l(i)$, is given by,

$$v_l(i) = e_l(S(i)) \max_k [v_k(i-1) a_{kl}]$$

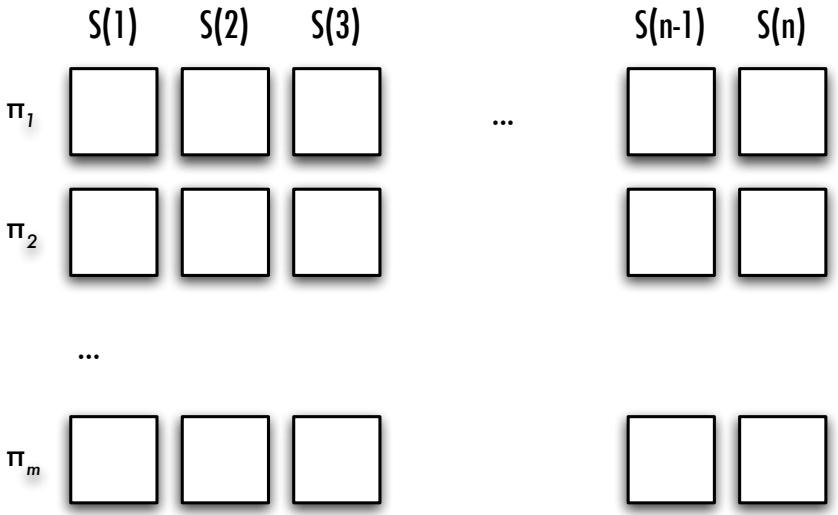


where k is running for states such that a_{kl} is defined.

Decoding problem

- The algorithm for solving the decoding problem is known as the **Viterbi algorithm**. It finds the best (most probable) path using the **dynamic programming** technique.
  - **Forward.** This requires filling the table v, for all i and for all l — see the definition of $v_l(i)$ on the previous slide.
  - **Traceback.** Starting with $v_{end}(n)$, the algorithm reverses the computation to find the path with maximum joint probability.
- Sean R Eddy, What is dynamic programming?, *Nat Biotechnol* **22**:7, 90910, 2004.

Decoding problem — table v



Decoding problem — gene finding

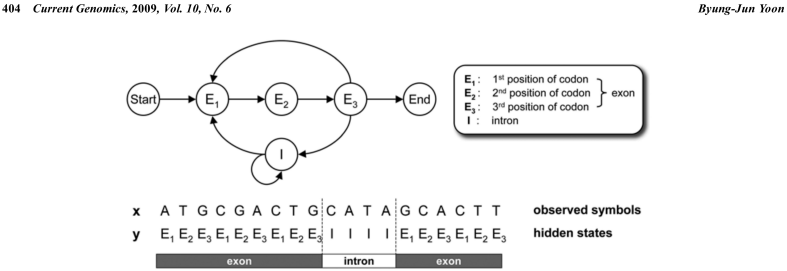
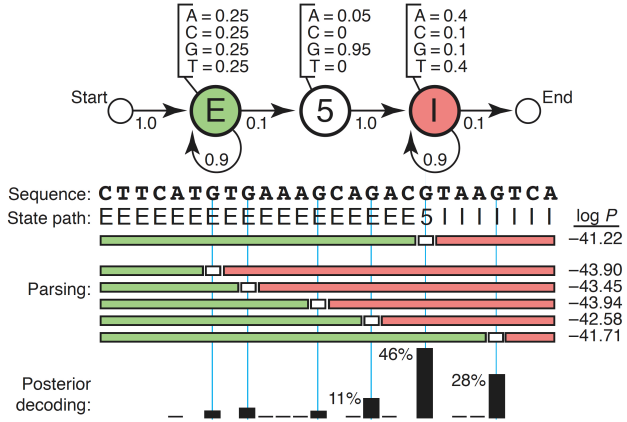


Fig. (1). A simple HMM for modeling eukaryotic genes.

Source: [2] Figure 1

- For a given input sequence, the **decoding problem** reveals the path with maximum join probability. Effectively telling us the nucleotides that are likely to be in **exons** (states E_1, E_2, E_3) and those that likely to be in **introns** (state I).

Decoding problem — Gene finding



Source: [1] Figure 1

The likelihood problem: calculating $P(S|\theta)$

- In the case of a **Markov chain** there is a single path for a given sequence S and therefore $P(S|\theta)$ is given by,

$$P(S|\theta) = P(S(1)) \prod_{i=2}^{n} a_{S(i-1)S(i)}$$

The likelihood problem: calculating $P(S|\theta)$

- In the case of a **Markov chain** there is a single path for a given sequence S and therefore $P(S|\theta)$ is given by,

$$P(S|\theta) = P(S(1)) \prod_{i=2}^{n} a_{S(i-1)S(i)}$$

- In the case of an **HMM**, there are several paths producing the same S (some paths will be more likely than others) and $P(S|\theta)$ should be defined as the sum of all the probabilities of all possible paths producing S,

$$P(S|\theta) = \sum_{\pi} P(S, \pi)$$

The likelihood problem: calculating $P(S|\theta)$

- In the case of a **Markov chain** there is a single path for a given sequence S and therefore $P(S|\theta)$ is given by,

$$P(S|\theta) = P(S(1)) \prod_{i=2}^{n} a_{S(i-1)S(i)}$$

- In the case of an **HMM**, there are several paths producing the same S (some paths will be more likely than others) and $P(S|\theta)$ should be defined as the sum of all the probabilities of all possible paths producing S,

$$P(S|\theta) = \sum_{\pi} P(S, \pi)$$

- The number of paths grows **exponentially** with respect to the **length of the sequence**, therefore all the paths cannot simply be enumerated and summed.

The likelihood problem: forward algorithm

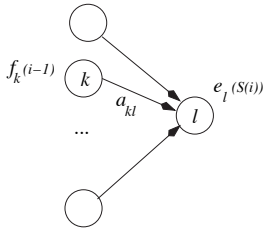
- Modifying the Viterbi algorithm **changing the maximization** by a **sum** calculates the probability of the observed sequence up to position i ending in state l,

$$f_l(i) = e_l(S(i)) \sum_k f_k(i-1) a_{kl}$$

The likelihood problem

- The score represents the probability of the sequence up to (and including) $S(i)$, noted $f_l(i)$, is given by,

$$f_l(i) = e_l(S(i)) \sum_k [f_k(i-1) a_{kl}]$$



where k is running for states such that a_{kl} is defined.

Model specification

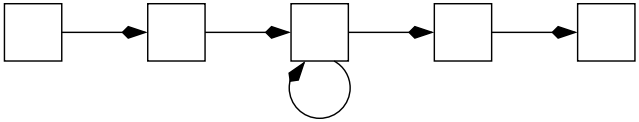
- We now turn to our third and final question. **How to determine the parameters of the model?**
- Let $x_1, \ldots, x_N$ be N independent examples forming the training set (typically, N sequences of observable symbols), the objective is to find a set parameters, θ, such that:

$$\max_{\theta} \Pi_{i=1}^{N} P(x_i|\theta)$$

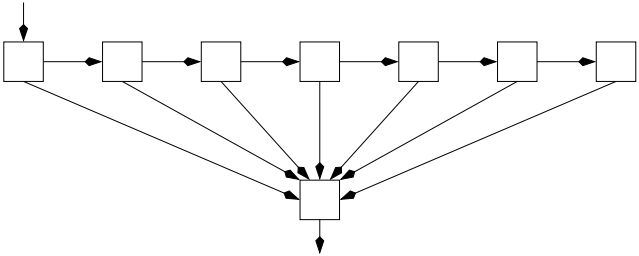
Model Specification

- **Structure (topology)**: **states** + **interconnect**
- Estimating the **transition/emission probabilities**

Modelling the length

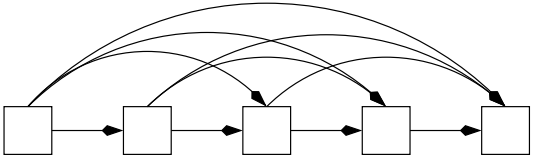


At least 5 symbols long

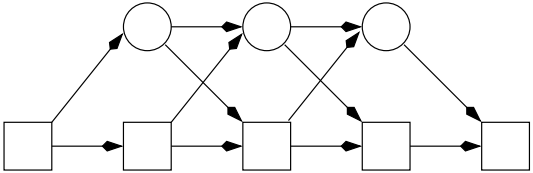


2 to 8 symbols long

Arbitrary deletions



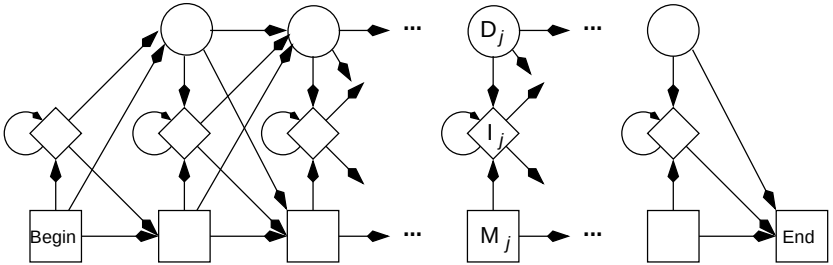
Too expensive, too many parameters to evaluate!



Silent (null) states do not emit symbols.

$\Rightarrow$ Silent states prevent modelling specific distant transitions.

Profile HMMs



$\Rightarrow$ Models insertion/deletions separately.

Gene prediction

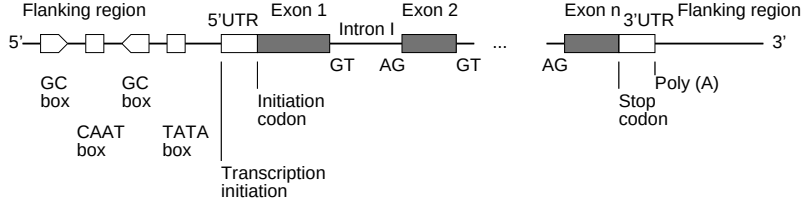


Fig. (1). A simple HMM for modeling eukaryotic genes.

Source: [2] Figure 1

The parameter estimation problem

- **Problem:** estimate the a_{st} and $e_k(b)$ probabilities.

  Given:

  - a **fixed topology**;
  - n independent positive examples: $S_1, S_2, \ldots, S_n$.

The parameter estimation problem

- **Problem:** estimate the a_{st} and $e_k(b)$ probabilities.

  Given:

  - a **fixed topology**;
  - n independent positive examples: $S_1, S_2, \ldots, S_n$.

Two scenarios:

- The paths are **known** (existing annotated genes)
- The paths are **unknown**

Expectation-Maximization (EM) algorithm

1. Choose an **initial model**. If no prior information is available, make all the transition probabilities equiprobable, similarly for the emission probabilities;

Expectation-Maximization (EM) algorithm

1. Choose an **initial model**. If no prior information is available, make all the transition probabilities equiprobable, similarly for the emission probabilities;
2. Use the **decoding algorithm** for finding the maximum likelihood path for each input sequence (**E-step**);

Expectation-Maximization (EM) algorithm

1. Choose an **initial model**. If no prior information is available, make all the transition probabilities equiprobable, similarly for the emission probabilities;
2. Use the **decoding algorithm** for finding the maximum likelihood path for each input sequence (**E-step**);
3. Using these **paths**, tally statistics for estimating all a_{kl} and $e_k(b)$ values (**M-step**);

Expectation-Maximization (EM) algorithm

1. Choose an **initial model**. If no prior information is available, make all the transition probabilities equiprobable, similarly for the emission probabilities;
2. Use the **decoding algorithm** for finding the maximum likelihood path for each input sequence (**E-step**);
3. Using these **paths**, tally statistics for estimating all a_{kl} and $e_k(b)$ values (**M-step**);
4. **Repeat 3 and 4** until the parameter estimates converge.

Expectation-Maximization (EM) algorithm

1. Choose an **initial model**. If no prior information is available, make all the transition probabilities equiprobable, similarly for the emission probabilities;
2. Use the **decoding algorithm** for finding the maximum likelihood path for each input sequence (**E-step**);
3. Using these **paths**, tally statistics for estimating all a_{kl} and $e_k(b)$ values (**M-step**);
4. **Repeat 3 and 4** until the parameter estimates converge.

Expectation-Maximization (EM) algorithm

1. Choose an **initial model**. If no prior information is available, make all the transition probabilities equiprobable, similarly for the emission probabilities;
2. Use the **decoding algorithm** for finding the maximum likelihood path for each input sequence (**E-step**);
3. Using these **paths**, tally statistics for estimating all a_{kl} and $e_k(b)$ values (**M-step**);
4. **Repeat 3 and 4** until the parameter estimates converge.

Sometimes called the **Baum-Welch** algorithm. **Gradient descent** can also be used.

- Chuong B Do and Serafim Batzoglou, What is the expectation maximization algorithm?, *Nat Biotechnol* **26**:8, 897899, 2008.

Applications

sklearn.hmm

- **sklearn.hmm** has been "deprecated due to it no longer matching the scope and the API of the project." It was removed starting with the release 0.17 [as of writing this, the current version of Scikit-Learn is 0.21.3].
- **Pomegranate** implements probabilistic models, including hidden Markov models.
  - Documentation
- Most of the time, hidden Markov models are implemented in specializedd tools, such as **GENSCAN**, **GENIE**, **HMMGENE**, **UGENE**, **SAM**, **HMMER**, etc.

HMMER

- Eddy, S. R. Profile hidden Markov models. *Bioinformatics* **14**, 755763 (1998).
  - 3371 citations according to Scopus
- J. Mistry, R. D. Finn, S. R. Eddy, A. Bateman, M. Punta. Challenges in Homology Search: HMMER3 and Convergent Evolution of Coiled-Coil Regions. *Nucleic Acids Research* **41**:e121, 2013.
- `http://hmmer.org/publications.html`
- `http://hmmer.org`

Pfam

- "The **Pfam** database is a large collection of protein families, each represented by multiple sequence alignments and **hidden Markov models** (HMMs)."
- E.L.L. Sonnhammer, S.R. Eddy and R. Durbin. Pfam: a comprehensive database of protein families based on seed alignments. *Proteins* **28**:405-420, 1997.
  - 806 citations according to Scopus
- S. El-Gebali, *et al.* The Pfam protein families database in 2019. *Nucleic Acids Research* (2019), doi: 10.1093/nar/gky995
- Pfam **32.0**, September 2018, **17,929 entries**
- `https://pfam.xfam.org`

Prologue

Summary

- A **Markov process** is **memoryless**, which means that the probability that the system be in state E_j at time t depends only on the state, E_i, at time time $t-1$ (first order model).

Summary

- A **Markov process** is **memoryless**, which means that the probability that the system be in state E_j at time t depends only on the state, E_i, at time time $t-1$ (first order model).
- Those probabilities do not depend on the value of t. This property is called **homogeneity of time**. Here, time is finite.

Summary

- A **Markov process** is **memoryless**, which means that the probability that the system be in state E_j at time t depends only on the state, E_i, at time time $t-1$ (first order model).
- Those probabilities do not depend on the value of t. This property is called **homogeneity of time**. Here, time is finite.
- A **hidden Markov model** comprises two elements: a sequence of **observable symbols** and a **sequence of hidden states**.

Next module

- **Support Vector Machine**

Appendix

Decoding problem

- If the observed sequence of symbols was of length one, the sequence of states would also be of length one (in our restricted example).

Decoding problem

- If the observed sequence of symbols was of length one, the sequence of states would also be of length one (in our restricted example).
- Which state would you predict if the observed symbol was a 0? What if it was a 1?

Decoding problem

- If the observed sequence of symbols was of length one, the sequence of states would also be of length one (in our restricted example).
- Which state would you predict if the observed symbol was a 0? What if it was a 1?
- Now consider an observed sequence of length two, let's assume that the last symbol is 1, what is the probability of that symbol being emitted from state π_1?

Decoding problem

- If the observed sequence of symbols was of length one, the sequence of states would also be of length one (in our restricted example).
- Which state would you predict if the observed symbol was a 0? What if it was a 1?
- Now consider an observed sequence of length two, let's assume that the last symbol is 1, what is the probability of that symbol being emitted from state π_1?
- There are two ways of ending up in π_1 while producing $S(2)$:

Decoding problem

- If the observed sequence of symbols was of length one, the sequence of states would also be of length one (in our restricted example).
- Which state would you predict if the observed symbol was a 0? What if it was a 1?
- Now consider an observed sequence of length two, let's assume that the last symbol is 1, what is the probability of that symbol being emitted from state π_1?
- There are two ways of ending up in π_1 while producing $S(2)$:
  1. $S(1)$ could have been produced from π_1, and the state remained π_1,

Decoding problem

- If the observed sequence of symbols was of length one, the sequence of states would also be of length one (in our restricted example).
- Which state would you predict if the observed symbol was a 0? What if it was a 1?
- Now consider an observed sequence of length two, let's assume that the last symbol is 1, what is the probability of that symbol being emitted from state π_1?
- There are two ways of ending up in π_1 while producing $S(2)$:
  1. $S(1)$ could have been produced from π_1, and the state remained π_1,
  2. or 2) $S(1)$ could have been produced from π_2, and there was a transition π_2 to π_1.

Decoding problem

- If the observed sequence of symbols was of length one, the sequence of states would also be of length one (in our restricted example).
- Which state would you predict if the observed symbol was a 0? What if it was a 1?
- Now consider an observed sequence of length two, let's assume that the last symbol is 1, what is the probability of that symbol being emitted from state π_1?
- There are two ways of ending up in π_1 while producing $S(2)$:
  1. $S(1)$ could have been produced from π_1, and the state remained π_1,
  2. or 2) $S(1)$ could have been produced from π_2, and there was a transition π_2 to π_1.
- The two joint probabilities would be $P(S(1)|\pi_1)P(\pi_1 \rightarrow \pi_1)P(S(2)|\pi_1)$ and $P(S(1)|\pi_2)P(\pi_2 \rightarrow \pi_1)P(S(2)|\pi_1)$.

Decoding problem

- Now consider an observed sequence of length three, let's assume that the last symbol is 1, what is the probability of that symbol being emitted from state π_1?
- There are two ways of ending up in π_1 while producing $S(3)$:
  1. the last state that led to the production of the sequence of symbols $S[1,2]$ was π_1 and the state remained π_1,
  2. the last state that led to the production of the sequence of symbols $S[1,2]$ was π_2 and it is followed by a transition π_2 to π_1, with probability a_{21}.

Let's define $v_k(i)$ as **the probability of the most probable path ending in state** k **while producing the observation** i. Using this notation for formulating the probabilities for the above two scenarios.

$$v_1(3) = \max\ [\ v_1(2) \times a_{11} \times e_1(0),\ \ v_2(2) \times a_{21} \times e_1(0)\ ]$$

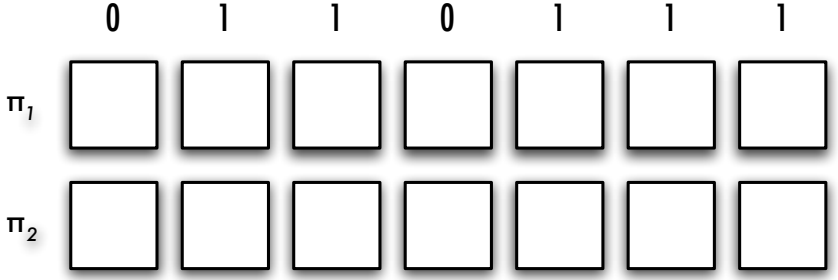
Decoding problem

- For our 2 states HMM, we can write the following equation,

$$v_1(i) = \max\ [\ v_1(i-1) \times a_{11} \times e_1(S(i)),\ v_2(i-1) \times a_{21} \times e_1(S(i))\ ]$$

$$v_2(i) = \max\ [\ v_1(i-1) \times a_{12} \times e_2(S(i)),\ v_2(i-1) \times a_{22} \times e_2(S(i))\ ]$$

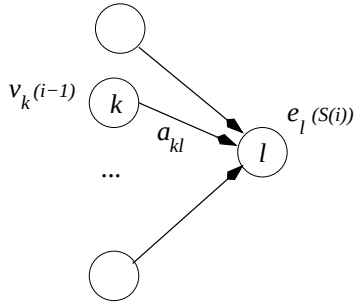
Decoding problem



Decoding problem

- The **most probable path** can be found **recursively**. The score for the most probable path ending in state l with observation i, noted $v_l(i)$, is given by,

$$v_l(i) = e_l(S(i)) \max_k [v_k(i-1) a_{kl}]$$



where k is running for states such that a_{kl} is defined.

Decoding problem

- The algorithm for solving the decoding problem is known as the **Viterbi algorithm**. It finds the best (most probable) path using the **dynamic programming** technique.
  - **Forward.** First, this requires filling the table v, for all i and for all l — see the definition of $v_l(i)$ on the previous slide.
  - **Traceback.** Next, starting with $v_{end}(n)$, the algorithm reverses the computation to find the path with maximum joint probability.
- Sean R Eddy, What is dynamic programming?, *Nat Biotechnol* **22**:7, 90910, 2004.

Decoding problem: Viterbi algorithm

- **Initialization**:

$$v_0 = 1, v_k = 0, \ k > 0$$

- **Recurrence**:

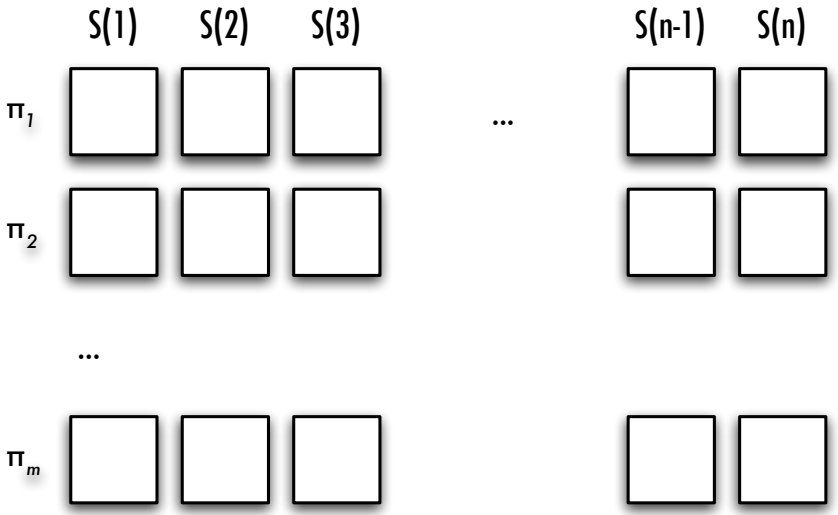
$$v_l(i) = e_l(S(i)) \max_k (v_k(i-1)a_{kl})$$

where, $v_k(i)$ represents the probability of the most probable path ending in state k and position i in S.

- A pointer (backward) is kept from l to the value of k that maximizes $v_k(i-1)a_{kl}$.

$\Rightarrow$ **Implementation issues:** because of the products (small) probabilities leads to underflow the algorithm is implemented using the logarithm of the values and therefore the products becomes sums.

Decoding problem — table v



Decoding problem

```
# transition probabilities (t)
$t[0][0] = 0.9; $t[0][1] = 0.1;
$t[1][0] = 0.2; $t[1][1] = 0.8;

# emission probabilities (e)
$e[0][0] = 0.50; $e[0][1] = 0.50;
$e[1][0] = 0.05; $e[1][1] = 0.95;

# observed sequence (S)
@S = (0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1);

# initialization (d is the dynamic programming table)
$d[ 0 ][ 0 ] = $e[ 0 ][ $S[ 0 ] ];
$d[ 1 ][ 0 ] = $e[ 1 ][ $S[ 0 ] ];
```

Decoding problem

```
for ( $j=1; $j < @S; $j++ ) {
    for ( $i=0; $i <= 1; $i++ ) {
        $m = 0;
        for ( $k=0; $k <= 1; $k++ ) {
            $v = $d[$k][$j-1]*$t[$k][$i]*$e[$i][$S[$j]];
            if ( $v > $m ) {
                $from = $k; $to = $i; $m = $v;
            }
        }
        $d[ $i ][ $j ] = $m;
        $tr[ $i ][ $j ] = "($from->$to)";
    }
}
```

Decoding problem

```
for ( $i=0; $i <= 1; $i++ ) {
    for ( $j=0; $j < @S; $j++ ) {
        printf "\t%5.5f", $d[ $i ][ $j ];
    }
    print "\n";
    for ( $j=0; $j < @S; $j++ ) {
        printf "\t %s", $tr[ $i ][ $j ];
    }
    print "\n";
}
```

Decoding problem

t[0][0] = 0.9; t[0][1] = 0.1; t[1][0] = 0.2; t[1][1] = 0.8;
e[0][0] = 0.50; e[0][1] = 0.50; e[1][0] = 0.05; e[1][1] = 0.95;

```
      0       1       0       1       0       1       1       1       1       1       1       1
0.50000 0.22500 0.10125 0.04556 0.02050 0.00923 0.00415 0.00187 0.00084 0.00038 0.00017 0.00008
         (0->0)  (0->0)  (0->0)  (0->0)  (0->0)  (0->0)  (0->0)  (0->0)  (0->0)  (0->0)  (0->0)
0.05000 0.04750 0.00190 0.00962 0.00038 0.00195 0.00148 0.00113 0.00086 0.00065 0.00049 0.00038
         (0->1)  (1->1)  (0->1)  (1->1)  (0->1)  (1->1)  (1->1)  (1->1)  (1->1)  (1->1)  (1->1)
```

References


Sean R Eddy.
What is a hidden Markov model?
Nat Biotechnol, 22(10):1315–6, Oct 2004.


Byung-Jun Yoon.
Hidden Markov Models and their applications in biological sequence analysis.
Curr Genomics, 10(6):402–15, Sep 2009.


C Burge and S Karlin.
Prediction of complete gene structures in human genomic DNA.
J Mol Biol, 268(1):78–94, Apr 1997.


Sean R Eddy.
What is dynamic programming?
Nat Biotechnol, 22(7):909–10, Jul 2004.


Chuong B Do and Serafim Batzoglou.
What is the expectation maximization algorithm?
Nat Biotechnol, 26(8):897–899, Aug 2008.

Marcel **Turcotte**

Marcel.Turcotte@uOttawa.ca

School of Electrical Engineering and **Computer Science** (EECS)
University of Ottawa