UNIVERSIDAD
DE GRANADA

MODELOS DE MARKOV OCULTOS Y APLICACIONES A LA BIOLOGÍA

XUSHENG ZHENG

Trabajo Fin de Grado

Doble Grado en Ingeniería Informática y Matemáticas

Tutores

Lidia Fernández Rodríguez

FACULTAD DE CIENCIAS

E.T.S. INGENIERÍAS INFORMÁTICA Y DE TELECOMUNICACIÓN

Granada, a 25 de junio de 2023

RESUMEN

En ocasiones, se necesita estudiar sucesos que no son directamente observables pero que generan una serie de consecuencias apreciables. Para inferir sobre los sucesos ocultos podemos utilizar los modelos de Markov ocultos, un modelo estadístico que es capaz de proporcionarnos información de eventos no observables a partir de unas salidas conocidas.

En este trabajo se estudian los elementos que constituyen un modelo de Markov oculto y los algoritmos derivados de dichos modelos. Algunos de los cuales, basándose en la programación dinámica. Con dichos algoritmos, podemos inferir sobre los eventos ocultos a partir de una sucesión de observaciones. Previo al estudio de los modelos de Markov ocultos, introduciremos un tipo de proceso estocástico conocido como cadenas de Markov. El análisis de estos procesos es fundamental, ya que los modelos ocultos pueden considerarse como una generalización de cadenas de Markov. Finalmente, para mostrar las aplicaciones de los modelos de Markov ocultos en la biología, presentaremos e implementaremos modelos que se utilizan en problemas específicos, en concreto, para alineamiento y análisis de secuencias.

Palabras clave: Probabilidad, procesos estocásticos, cadenas de Markov, modelos de Markov ocultos, programación dinámica, bioinformática, secuencias biológicas.

SUMMARY

In many occasions, it is necessary to study events that are not directly observable but generate a series of noticeable consequences. These hidden events typically represent underlying properties that are more important than their consequences or outputs. For instance, when listening to someone speak in an unfamiliar language, we aim to understand what they are saying rather than just hearing the sounds. To infer information about these hidden events, we can employ hidden Markov models (HMMs), which are statistical models capable of providing insights into unobservable events based on observed outputs.

Throughout this project, we will study the concepts, properties and algorithms associated with hidden Markov models. These models can be considered as a generalization of Markov chains, which are stochastic processes where the the probability of a certain event depends only on the immediately preceding event. Markov chains are interesting by their own since they represent the transitions between probable events. By analyzing these transitions, we can make predictions both in the short-term and long-term scenarios.

Therefore, Markov chains have a wide range of applications. They are used in weather prediction and forecasting, studying population dynamics in biology and epidemiology. They are also valuable for modeling the evolution of financial assets and gambling games. Additionally, Markov chains play a significant role in Google's Pagerank algorithm, determining the ranking position of a webpage in search results.

However, things become more complex when we aim to study unobservable events or underlying properties that lead to certain outputs. In such cases, direct application of Markov chains is not possible. Instead, we turn to hidden Markov models (HMMs) for inference. HMMs are robust probabilistic models widely employed in diverse fields, including speech recognition, finance, pattern recognition, bioinformatics, and more. Among the various possibilities, our focus will be on exploring the applications of HMMs in the field of biology, particularly in bioinformatics.

Bioinformatics is defined as the science that studies and analyzes large sets of biological data, particularly genetic data, using mathematical, statistical, and computer science knowledge. The study of genetic sequences aids in understanding biological mechanisms within cells. However, the vast amount of data involved makes it impossible to extract relevant information without an efficient tool. The use of hidden Markov models for this type of problem was progressively introduced starting from the 90s. Previously, they had already been employed in speech recognition and we-

re notable for their effectiveness in modeling correlations between adjacent symbols, properties, or events.

One of the main applications of hidden Markov models is in the analysis of genetic sequences. By studying the similarities between two sequences, we can determine if they are related and potentially have the same function. In order to compare these similarities, the sequences need to be aligned, and variants of HMMs have been developed specifically for sequence alignment and comparison. HMMs also allow us to identify specific regions within a sequence that are important for analysis, such as CpG Islands. These examples represent just a small fraction of the diverse applications of HMMs in biology. The adaptability and flexibility of HMMs make them valuable tools in tackling a wide range of specific problems in the field.

OBJECTIVES

The objectives to be addressed in this project are as follows:

- To study Markov chains, including the different types of states they can have and their application in inferring long-term behavior.
- To study hidden Markov models (HMMs) and the associated algorithms.
- To explore the applications of hidden Markov models in biology, to analyze specific variants and implements them.

STRUCTURE

We have divided this document into several parts:

- Preliminaries
- Introduction to Markov chains
- Hidden Markov models
- Applications in biology

Preliminaries

To study Markov chains and hidden Markov models, we will start with some basic concepts. We will introduce the definition of probability, conditional probability and the law of total probability. In order to analyze specific states of Markov chains, we will also present theories related to positive matrices. Different types of matrices such as stochastic matrices, irreducible matrices, primitive matrices and their associated propositions and theorems are exhibited in this chapter too. Eventually, the concept of

dynamic programming is introduced. This technique will be applied in the algorithms associated with hidden Markov models.

Introduction to Markov chains

In this chapter, we will formalize the concept of Markov chains. We will introduce the concept of associated transition matrix and explore the various types of states they can exhibit. By analyzing these states, we can obtain information about the probabilities of specific events occurring based on particular states, as well as the time it takes for such events to happen. At the end of the chapter, we will study the stationary distribution of a Markov chain, which characterizes its long-term behavior. To deal with all these theories we will make use of concepts introduced in the previous chapter.

Hidden Markov models

In the third chapter, we study the core focus of this project, which revolves around hidden Markov models. We begin by introducing the key components that constitute a hidden Markov model, as well as the three fundamental problems associated to this model. To address these problems, we present dedicated algorithms developed for each case: the forward-backward algorithm, the Viterbi algorithm, and the Baum-Welch algorithm. For each algorithm, we will present a simple example demonstrating their functionality. Additionally, we will provide improvements of these algorithms to prevent the issue of underflow that may arise during their computation.

Applications in biology

The last chapter shows some of the many applications of HMMs in biology. We first provide a basic understanding of biology, including the elements that compose DNA and the process of transforming DNA into a protein. Additionally, we introduce two Python libraries, NumPy and hmmlearn, which we will utilize to implement different variants of HMMs for specific biological problems.

We then focus on a specific problem in pattern recognition: the identification of CpG islands. By employing a simple HMM, we demonstrate how this problem can be solved. Furthermore, we discuss the problem of pair alignment between two biological sequences and introduce the pair HMMs, a variant specifically developed to address this case. We present the implementation of this model and provide an illustrative example to showcase its application and results.

Finally, we will talk about profile HMMs, which are designed to represent properties of a sequence family. We explain the concept of profile HMMs and proceed to implement and showcase the outcomes of this model.

Keywords: Probability, stochastic process, Markov chains, hidden Markov models, dynamic programming, bioinformatics, biological sequences.

ÍNDICE GENERAL

INTRODUCCIÓN Y OBJETIVOS

A lo largo de este trabajo se estudiará el modelo estadístico conocido como modelo de Markov oculto. Este modelo se puede considerar como una generalización de las cadenas de Markov, un tipo de proceso estocástico en el que la probabilidad de un evento determinado depende únicamente del evento inmediatamente anterior. Además, permiten la realización de predicciones: mediante el análisis de las transiciones entre los estados, podemos predecir el comportamiento futuro del sistema.

Las cadenas de Markov tienen una amplia gama de aplicaciones. Se utilizan en la predicción del clima y las condiciones meteorológicas, el estudio de la evolución de las poblaciones en biología y epidemiología. También son útiles para modelar la evolución de activos financieros y juegos de azar. Además, se emplean en el algoritmo Pagerank de Google para determinar la posición de una página en los resultados de búsqueda.

Sin embargo, en muchas ocasiones los eventos que pretendemos estudiar no son directamente observables, lo que imposibilita el uso directo de cadenas de Markov en esos casos. Para abordar este problema, surgieron los modelos de Markov ocultos. Inicialmente fueron introducidos por Baum y Petrie en 1966 [4] como funciones probabilísticas de cadenas de Markov. A partir de ahí se han desarrollado hasta llegar a la forma en que se conocen hoy en día. Mediante la utilización de estos modelos podemos estudiar sucesiones de procesos no observables conocidos como ocultos, a partir de una serie de observaciones relacionadas con los anteriores. Por esta razón, tienen una amplia aplicación en campos como el reconocimiento del habla, procesamiento de señales, reconocimiento de patrones, biología, epidemiología, etc. Entre todas estas posibles aplicaciones, nos centraremos en el uso de los modelos de Markov ocultos en la biología, en concreto, en la bioinformática.

Se define la bioinformática como la ciencia que estudia y analiza grandes conjuntos de datos biológicos, y en particular, genéticos, mediante la aplicación de conocimientos matemáticos, estadísticos y de ciencias de la computación [10]. El estudio de las secuencias genéticas nos ayuda a comprender los mecanismos biológicos en las células. Sin embargo, la gran cantidad de datos que supone hace que sea imposible extraer información relevante sin una herramienta eficaz.

El uso de los modelos de Markov ocultos para este tipo de problema fue introducido por primera vez por Churchill en 1989 [6]. Anteriormente, ya habían sido empleados en el reconocimiento del habla en 1980 [11] y destacaban por la eficacia a la hora de modelar correlaciones entre símbolos adyacentes, propiedades o eventos [19].

El estudio de las secuencias genéticas nos proporciona información sobre sus funciones, con frecuencia, se descubren nuevas secuencias que requieren análisis. Una de las posibilidades es comparar las similitudes significativas entre la nueva secuencia y una secuencia conocida. Dependiendo del resultado, podemos inferir información acerca de la estructura o función de la nueva secuencia. Previo a la evaluación de similitudes, en general es necesario encontrar un alineamiento razonable entre las dos secuencias consideradas. Al hablar de alineamiento entre dos secuencias, tomamos en cuenta la posibilidad de insertar huecos en ambas secuencias.

Para construir estos alineamientos, se diseña un modelo de puntuaciones a partir de consideraciones experimentales y estadísticas. Estas consideraciones también son aplicables en los modelos de Markov ocultos, lo que les permite alinear secuencias desde una perspectiva probabilística. Una ventaja significativa de los modelos de Markov ocultos es que nos permiten evaluar las similitudes entre dos secuencias independientemente de un alineamiento específico.

Esta es una de las posibles aplicaciones de los modelos de Markov ocultos en biología. Existen diversas variantes del modelo general adaptadas a problemas específicos. Esta capacidad de adaptación es una de las ventajas clave de los modelos de Markov ocultos, ya que permiten ajustar su estructura para simular situaciones particulares. Además, como veremos en este trabajo, también permiten el ajuste de sus parámetros basados en observaciones. Esto es de gran utilidad cuando desconocemos los estados que generan dichas observaciones, pues a partir de los parámetros podemos establecer relaciones entre los estados ocultos y las observaciones.

Descripción del trabajo

El presente trabajo desarrolla la teoría de las cadenas de Markov y los modelos de Markov ocultos. Nos centraremos en los algoritmos utilizados en los modelos ocultos y los problemas que se pueden abordar a partir de dichos modelos. Desde el punto de vista matemático, se usarán principalmente conceptos relacionados con la probabilidad y la inferencia estadística. La programación dinámica será la herramienta informática fundamental en el trabajo. Para estudiar las aplicaciones de los modelos de Markov ocultos en la biología, implementaremos clases que simulen los modelos correspondientes en formato de Jupyter Notebook y los utilizaremos para ilustrar el funcionamiento de dichos modelos mediante ejemplos.

Contenido de la memoria

El trabajo se estructura en cuatro capítulos principales. En el primero, se recopilan conceptos básicos de probabilidad y programación dinámica. También presentaremos resultados necesarios sobre matrices positivas para el estudio de las cadenas de

Markov. El segundo capítulo está dedicado a las cadenas de Markov. Se introducen definiciones y propiedades básicas que serán de gran utilidad más adelante para el desarrollo del siguiente capítulo. En concreto se hará una clasificación de los distintos estados de una cadena de Markov y se presentarán algunos resultados sobre el comportamiento y evolución del modelo a largo plazo. En el capítulo 3 se introducen los conceptos básicos relacionados con los modelos de Markov ocultos y se explicarán detalladamente los algoritmos asociados a ellos. Finalmente, analizaremos problemas de biología que pueden ser abordados mediante el uso de los modelos ocultos o sus variantes.

Objetivos del trabajo

Los objetivos que se pretenden abordar en este trabajo son:

- Estudiar las cadenas de Markov, los tipos de estados y su aplicación para inferir comportamiento a largo plazo.
- Estudiar los modelos de Markov ocultos, así como los algoritmos relacionados.
- Estudiar las aplicaciones de los modelos de Markov ocultos en la biología, analizar variantes adaptadas a problemas particulares e implementar de dichos variantes.

1

PRELIMINARES

En este capítulo vamos a presentar las herramientas básicas para llevar a cabo el desarrollo del trabajo. Empezaremos recordando conceptos básicos de probabilidad, presentaremos algunos resultados relacionados con las matrices positivas y finalmente daremos un breve repaso a la idea de programación dinámica.

1.1 CONCEPTOS BÁSICOS DE PROBABILIDAD

Comenzamos presentado los elementos necesarios para considerar una probabilidad:

Definición 1.1. Sea Ω un conjunto arbitrario, diremos que una familia no vacía de subconjuntos de Ω, $\mathcal{A} \subseteq \mathcal{P}(\Omega)$, siendo $\mathcal{P}(\Omega)$ conjunto potencia de Ω, es una σ-álgebra si:

- Es cerrada para complementarios: $\forall A \in \mathcal{A}$, $\Omega \setminus A \in \mathcal{A}$.
- Es cerrada para uniones numerables: si $A_n \in \mathcal{A}$, $\forall n \in \mathbb{N} \implies \bigcup_{n\in\mathbb{N}} A_n \in \mathcal{A}$.

A la dupla $(\Omega, \mathcal{A})$ se le conoce como espacio medible.

Definición 1.2. Sea $(\Omega, \mathcal{A})$ un espacio medible, $P : \mathcal{A} \longrightarrow [0,1]$ es una probabilidad si:

- $P(A) \geq 0$, $\forall A \in \mathcal{A}$.
- $P(\Omega) = 1$.
- Dada una secuencia $\{A_n\}_{n\in\mathbb{N}} \subseteq \mathcal{A}$ con $A_i \not\parallel A_j$, $i \neq j$ entonces:

$$P\left(\bigcup_{n\in\mathbb{N}} A_n\right) = \sum_{n\in\mathbb{N}} P(A_n).$$

A la terna $(\Omega, \mathcal{A}, P)$ se le conoce como espacio probabilístico.

Definición 1.3. Sea $(\Omega, \mathcal{A}, P)$ un espacio probabilístico y $A \in \mathcal{A}$ con $P(A) > 0$. Sea $B \in \mathcal{A}$, se define la probabilidad de B condicionada a A como:

$$P(B|A) = \frac{P(B \cap A)}{P(A)}.$$

Relacionado con esta definición presentamos el siguiente teorema que nos será útil:

Teorema 1.4 (Teorema de probabilidad total)**.** Sea $(\Omega, \mathcal{A}, P)$ un espacio probabilístico y $\{A_n\}_{n\in\mathbb{N}} \subseteq \mathcal{A}$ una partición de Ω con $P(A_n) > 0$, $\forall n \in \mathbb{N}$. Sea $B \in \mathcal{A}$, entonces:

$$P(B) = \sum_{n\in\mathbb{N}} P(A_n)P(B|A_n).$$

Demostración. Por ser $\{A_n\}$ una partición de Ω, aplicando la definición de probabilidad condicionada y teniendo en cuenta que $P(A_n) > 0$:

$$P(B) = P\left(\bigcup_{n\in\mathbb{N}} B \cap A_n\right) = \sum_{n\in\mathbb{N}} P(B \cap A_n) = \sum_{n\in\mathbb{N}} P(A_n)P(B|A_n).$$

□

Definición 1.5. Sea $(\Omega, \mathcal{A}, P)$ un espacio probabilístico y $(\Omega', \mathcal{A}')$ un espacio medible. Una función $X : (\Omega, \mathcal{A}, P) \longrightarrow (\Omega', \mathcal{A}')$ es una variable aleatoria si:

$$X^{-1}(B) \in \mathcal{A}, \quad \forall B \in \mathcal{A}'.$$

1.2 MATRICES POSITIVAS

Para el estudio de las cadenas de Markov, necesitaremos algunos resultados previos sobre matrices. Por comodidad, usaremos a lo largo de este trabajo la notación fila para representar los vectores. Los contenidos de esta sección se basan principalmente en [22].

Definición 1.6. Sea una matriz $A = [a_{ij}]$, diremos que A es:

- **no negativa** si $a_{ij} \geq 0$, $\forall i, j$.
- **positiva** si $a_{ij} \geq 0$, $\forall i, j$ y existe $a_{ij} > 0$ para al menos un par de índices i, j.
- **estrictamente positiva** si $a_{ij} > 0$, $\forall i, j$.

Definición 1.7. Sea $A = [a_{ij}]$ una matriz cuadrada de dimensión N, diremos que A es una **matriz estocástica** si:

$$a_{ij} \in [0, 1], \quad \forall i, j \in \{1, ..., N\},$$
$$\sum_{j=1}^{N} a_{ij} = 1, \quad \forall i \in \{1, ..., N\}.$$

Una forma de caracterizar las matrices estocásticas es mediante la siguiente proposición:

Proposición 1.8. Sea A una matriz cuadrada positiva de dimensión $N \times N$:

1. A es estocástica si y solo si 1 es un valor propio de A^T con vector propio $\mathbf{1} = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}$.
2. si A es estocástica, entonces para todo valor propio λ, se cumple que $|\lambda| \leq 1$.

Demostración.

1. Es suficiente con observar que la condición de estocasticidad para una matriz positiva A es equivalente a que $\mathbf{1} \cdot A^T = \mathbf{1}$.
2. Sea $v = (v_1, \dots, v_N)$ un vector propio asociado a λ (a izquierda pues estamos usando la notación fila), por ser A positiva y estocástica, se verifica:

$$\begin{aligned} |\lambda| \sum_{j=1}^{N} |v_j| &= \sum_{j=1}^{N} |\lambda v_j| = \sum_{j=1}^{N} |(vA)_j| = \sum_{j=1}^{N} \left| \sum_{i=1}^{N} a_{ij} v_i \right| \\ &\leq \sum_{j=1}^{N} \sum_{i=1}^{N} a_{ij} |v_i| = \sum_{i=1}^{N} \left(\sum_{j=1}^{N} a_{ij} \right) |v_i| = \sum_{i=1}^{N} |v_i| . \end{aligned}$$

   Puesto que $\sum_{r=1}^{N} |v_r| > 0$, tenemos que $|\lambda| \leq 1$. □

Observación. De la primera afirmación de la proposición 1.8, podemos ver que el producto de matrices estocásticas sigue siendo estocástica. En efecto, si A y B son dos matrices estocásticas entonces $\mathbf{1} \cdot (AB)^T = \mathbf{1} \cdot B^T \cdot A^T = \mathbf{1} \cdot A^T = \mathbf{1}$.

Presentamos a continuación la definición de grafo dirigido y grafo de una matriz no negativa:

Definición 1.9. Un grafo dirigido G es un par (V, L) donde $V = \{v_1, \dots, v_n\}$ es un conjunto finito de elementos llamados **nodos** (o **vértices**) y $L = \{l_1, \dots, l_m\} \subseteq V \times V$ es un conjunto de pares ordenados de dichos nodos llamados **arcos** (o **aristas**).

Definición 1.10. Sea $G = (V, L)$ un grafo dirigido, un **camino dirigido** C desde v_{i_0} a v_{i_p} es una secuencia de nodos $C = \{v_{i_0}, v_{i_1}, \dots, v_{i_p}\}$ tal que $v_{i_k} \in V$ para todo $k = 0, \dots, p$ y $(v_{i_{k-1}}, v_{i_k}) \in L$ para todo $k = 1, \dots, p$.

Definición 1.11. Sea $G = (V, L)$ un grafo dirigido, diremos que:

- un nodo $v_i \in V$ está **conectado** con un nodo $v_j \in V$ si existe un camino dirigido de v_i a v_j.
- un nodo $v_i \in V$ está **fuertemente conectado** con un nodo $v_j \in V$ si ambos están conectados.

Un grafo dirigido está **fuertemente conectado** si sólo tiene un nodo o todos sus nodos están fuertemente conectados entre ellos.

Definición 1.12. Sea A una matriz cuadrada no negativa de dimensión N, el grafo dirigido asociado a A es de la forma $G_A = (V_A, L_A)$ donde:

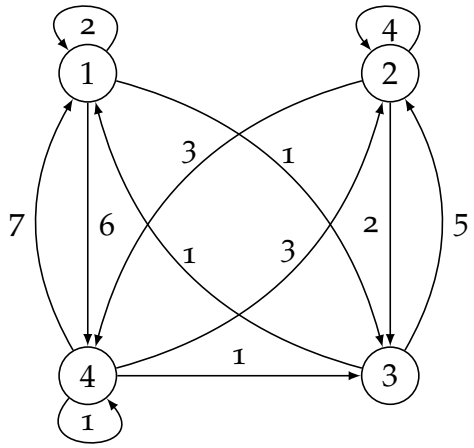
$$V_A = \{1, \ldots, N\}$$

$$L_A = \{(i, j) \in V_A \times V_A \mid a_{ij} > 0\}.$$

Ejemplo 1.1. Dada la matriz:

$$A = \begin{pmatrix} 2 & 0 & 1 & 6 \\ 0 & 4 & 2 & 3 \\ 1 & 5 & 0 & 0 \\ 7 & 3 & 1 & 1 \end{pmatrix}$$

Podemos representar su grafo asociado:



El grafo de esta matriz está fuertemente conectado pues desde cualquier nodo podemos encontrar un camino dirigido a cualquier de los otros nodos, incluido a él mismo.

A continuación presentaremos dos tipos especiales de matrices y algunos resultados relacionados que nos servirán en el siguiente capítulo:

Definición 1.13. Sea A una matriz cuadrada no negativa, se dice que A es **irreducible** si G_A está fuertemente conectado.

Teorema 1.14. Sea A una matriz irreducible, entonces:

1. Existe un valor propio positivo y simple λ_1 tal que para todo valor propio λ, se tiene que $\lambda_1 \geq \lambda$.

2. Existe un único vector propio asociado a λ_1 con todos las componentes estrictamente positivas.
3. Todos los valores propios con módulo igual a λ_1 son simples.

Demostración. Véase [22, Página 263, Teorema 7.13]. □

Definición 1.15. Sea A una matriz cuadrada positiva, se dice que A es **primitiva** si existe $k \in \mathbb{N}$ tal que A^k es estrictamente positiva.

Teorema 1.16 (Teorema de Perron-Frobenius)**.** Sea A una matriz primitiva, entonces:

1. A tiene un valor propio λ_1 real, estrictamente positivo y dominante, esto es:

$$|\lambda_i| < \lambda_1, \ \forall \lambda_i \in \sigma(A) \setminus \{\lambda_1\}$$

   y $\rho(A) = \lambda_1$.
2. Se puede tomar un vector propio v_1 asociado al valor propio λ_1 con todas las componentes estrictamente positivas.

Demostración. Véase [22, Página 202]. □

Como consecuencia de este teorema tenemos el siguiente corolario:

Corolario 1.17. Sea A una matriz primitiva de dimensión $N \times N$, λ_1 su valor propio real, estrictamente positivo y dominante y v_1 un vector propio asociado a λ_1. Entonces, sea X_0 un vector de dimensión N con todas las componentes no negativas y al menos una componente estrictamente positiva:

$$\lim_{n\to\infty} \frac{1}{||X_0 A^n||_1} X_0 A^n = \frac{1}{||v_1||_1} v_1.$$

Demostración. Véase [22, Página 201, Teorema 5.19]. □

1.3 PROGRAMACIÓN DINÁMICA

A continuación recordaremos brevemente la idea subyacente de la programación dinámica, esta técnica se verá aplicada en los distintos algoritmos que iremos presentando a lo largo de este trabajo. Esta sección se ha basado en contenidos de [5].

Cuando encontramos un problema, lo lógico es dividirlo en problemas de menor tamaño siempre que sea posible, resolver dichos subproblemas y combinar las soluciones para resolver el problema original. Puede ocurrir de forma natural que a la hora de dividir el problema original obtengamos subproblemas que necesitan para su resolución realizar los mismos cálculos. Para evitar repetir las mismas operaciones en varias ocasiones, se puede crear una tabla donde se almacenan los resultados de

las operaciones realizadas y que se irá rellenando a medida que se vayan resolviendo los subproblemas. En esta idea simple, es lo que se apoya la programación dinámica.

No obstante, sólo podemos emplear programación dinámica en aquellos problemas en los que sea aplicable el principio de optimalidad. Este principio afirma que en una sucesión óptima de decisiones u opciones, toda subsucesión debe ser también óptima. Los problemas típicos en los que no es aplicable el principio de optimaliadad son problemas sobre la utilización de recursos limitados. En dichos casos, la combinación de soluciones óptimas de subproblemas puede conllevar a superar el límite de recursos.

Cuando es aplicable el principio de optimalidad a un problema, entonces la solución óptima es una combinación de soluciones óptimas de aquellos subproblemas que sean relevantes para el problema original considerado. Por lo tanto, la programación dinámica resuelve todos los subproblemas, determina los que son realmente relevantes y finalmente combina las soluciones en una solución óptima del problema original.

2

INTRODUCCIÓN A LAS CADENAS DE MARKOV

Las cadenas de Markov fueron introducidas por el matemático ruso Andréi Márkov en 1906. Desde su definición, se han utilizado para describir procesos importantes en la teoría de probabilidad. Hoy en día, tienen aplicaciones en campos como la biología, la economía, la química o en algoritmos para Internet (PageRank).

En este capítulo se va a introducir la teoría de cadenas de Markov, avanzado progresivamente hacia los modelos de Markov ocultos. Las fuentes principales de este capítulo son [25, Capítulo 4], [3, Capítulos 2 y 3] y [22, Capítulo 6].

2.1 PROPIEDAD DE MARKOV

Definición 2.1. Sea un conjunto finito $\mathbb{S} = \{s_1, ..., s_N\}$, un **proceso estocástico** sobre $\mathbb{S}$ es una sucesión $\{\mathcal{X}_0, \mathcal{X}_1, \mathcal{X}_2, ...\}$, o $\{\mathcal{X}_t\}_{t=0}^{\infty}$ para abreviar, donde cada $\mathcal{X}_t$ es una variable aleatoria que toma valores en $\mathbb{S}$.

A pesar de que el índice t puede representar cualquier magnitud, lo más común es que represente el tiempo. En este caso, las nociones de "pasado" y "futuro" aparecen de forma natural, esto es, si $t < t'$, entonces $\mathcal{X}_t$ es una variable "pasada" para $\mathcal{X}_{t'}$, mientras que $\mathcal{X}_{t'}$ es una variable "futura" para $\mathcal{X}_t$. Sin embargo, esto no sucede siempre así: por ejemplo, si el proceso estocástico corresponde a la secuencia del genoma de un organismo, el conjunto $\mathbb{S}$ estará formado por los cuatro símbolos para las subunidades de nucleótidos $\{A, C, G, T\}$ y las secuenciaciones tienen un significado más espacial que temporal.

Definición 2.2. Un proceso estocástico $\{\mathcal{X}_t\}_{t=0}^{\infty}$ se dice que posee **la propiedad de Markov**, o que es una **cadena de Markov**, si para todo $t \geq 1$ y $(u_0, ..., u_{t-1}, u_t) \in \mathbb{S}^{t+1}$ se tiene que:

$$P[\mathcal{X}_t = u_t | \mathcal{X}_0 = u_0, ..., \mathcal{X}_{t-1} = u_{t-1}] = P[\mathcal{X}_t = u_t | \mathcal{X}_{t-1} = u_{t-1}]. \tag{2.1}$$

Es decir, un proceso con la propiedad de Markov es aquel en el que la probabilidad de que ocurra un determinado suceso en el instante t sólo depende de lo que ocurrió en el instante $t-1$ y no de los estados previos.

Introducimos la notación $\mathcal{X}_j^k = (\mathcal{X}_j, \mathcal{X}_{j+1}, \ldots, \mathcal{X}_k)$ para denotar los estados $\mathcal{X}_i$ con $j \leq i \leq k$. Con esta notación, podemos reescribir la definición 1.1 como sigue: un proceso estocástico $\{\mathcal{X}_t\}$ es una **cadena de Markov** si, para todo $(u_0, ..., u_{t-1}, u_t) \in \mathbb{S}^{t+1}$ es cierto que:

$$P[\mathcal{X}_t = u_t | \mathcal{X}_0^{t-1} = (u_0, \ldots, u_{t-1})] = P[\mathcal{X}_t = u_t | \mathcal{X}_{t-1} = u_{t-1}]. \tag{2.2}$$

Tenemos que, por definición de probabilidad condicionada, para cualquier proceso estocástico $\{\mathcal{X}_t\}$ y cualquier secuencia $(u_0, ..., u_{t-1}, u_t) \in \mathbb{S}^{t+1}$:

$$P[\mathcal{X}_0^t = (u_0, \ldots, u_t)] = P[\mathcal{X}_0 = u_0] \cdot \prod_{i=0}^{t-1} P[\mathcal{X}_{i+1} = u_{i+1} | \mathcal{X}_0^i = (u_0, \ldots, u_i)].$$

Sin embargo, si consideramos una cadena de Markov, entonces la fórmula anterior se reduce a:

$$P[\mathcal{X}_0^t = (u_0, \ldots, u_t)] = P[\mathcal{X}_0 = u_0] \cdot \prod_{i=0}^{t-1} P[\mathcal{X}_{i+1} = u_{i+1} | \mathcal{X}_i = u_i]. \tag{2.3}$$

En (2.3) vemos la importancia del valor:

$$P[\mathcal{X}_{t+1} = u | \mathcal{X}_t = v]$$

al que podemos identificar como una función de tres variables: el estado "actual" $v \in \mathbb{S}$, el estado "siguiente" $u \in \mathbb{S}$ y el "tiempo actual" $t \in \mathbb{N}_0$. Así, teniendo en cuenta que $\mathbb{S} = \{s_1, ..., s_N\}$, definimos la probabilidad de transición:

$$a_{ij}(t) := P[\mathcal{X}_{t+1} = s_j | \mathcal{X}_t = s_i], \quad \forall t \in \mathbb{N}_0. \tag{2.4}$$

Por tanto, $a_{ij}(t)$ es la probabilidad de realizar una transición desde el estado actual s_i al estado siguiente s_j en el instante t.

Definición 2.3. Sea $\mathcal{X}_t$ una cadena de Markov, la matriz cuadrada de dimensión N, $A(t) = [a_{ij}(t)]$, es la **matriz de transición** de $\mathcal{X}_t$ en el instante t. Una cadena de Markov es **homogénea** si $A(t)$ es constante para todo $t \in \mathbb{N}_0$; en otro caso, es **no homogénea**.

Sea $\mathcal{X}_t$ una cadena de Markov que toma valores en un conjunto finito $\mathbb{S} = \{s_1, ..., s_N\}$ y sea $A(t)$ su matriz de transición en el instante t. Puesto que los elementos de una fila i de $A(t)$ son todas las probabilidades de realizar una transición desde el estado s_i, deben sumar 1 y por lo tanto, $A(t)$ es una matriz estocástica para todo t.

Para continuar con el estudio de las cadenas de Markov definimos el siguiente conjunto:

Definición 2.4. El **N-símplex estándar** es el subconjunto de $\mathbb{R}^{N+1}$ dado por:

$$\Delta^N = \{(t_1, ..., t_{N+1}) \in \mathbb{R}^{N+1} \mid \sum_{i=1}^{N+1} t_i = 1 \text{ y } t_i \geq 0 \text{ para todo } i\}.$$

Puesto que para todo t, $\sum_{i=1}^{N} P[\mathcal{X}_t = s_i] = 1$, podemos representar estas probabilidades con un vector $c^t \in \Delta^{N-1}$, siendo $c^t = (P[\mathcal{X}_t = s_1], \dots, P[\mathcal{X}_t = s_N])$.

Teorema 2.5. Sea $\{\mathcal{X}_t\}$ una cadena de Markov con valores en $\mathbb{S} = \{s_1, ..., s_N\}$ y sea $A(t)$ su matriz de transición en el instante t. Supongamos que $\mathcal{X}_0$ se distribuye de acuerdo con $c^0 = (c_1^0, \dots, c_N^0) \in \Delta^{N-1}$, esto es:

$$P[\mathcal{X}_0 = s_i] = c_i^0, \quad \forall i \in \{1, ..., N\}.$$

Entonces para todo $t \in \mathbb{N}$, $\mathcal{X}_t$ se distribuye de acuerdo con:

$$c^t = c^0 A(0) A(1) \cdots A(t-1). \tag{2.5}$$

Demostración. Vamos a demostrarlo por inducción. Para $t = 1$, por el teorema de la probabilidad total tenemos que:

$$P[\mathcal{X}_1 = s_j] = \sum_{i=1}^{N} P[\mathcal{X}_0 = s_i] \cdot P[\mathcal{X}_1 = s_j | \mathcal{X}_0 = s_i] = \sum_{i=1}^{N} c_i^0 \cdot a_{ij}(0).$$

Esto es, el producto del vector c^0 con la columna j-ésima de $A(0)$. Luego:

$$c^1 = (P[\mathcal{X}_1 = s_1], \dots, P[\mathcal{X}_1 = s_N]) = c^0 A(0).$$

Supongamos cierta la relación (2.5) para $t-1$, es decir, $c^{t-1} = c^0 A(0) A(1) \cdots A(t-2)$. Utilizando de nuevo el mismo teorema se tiene:

$$P[\mathcal{X}_t = s_j] = \sum_{i=1}^{N} P[\mathcal{X}_{t-1} = s_i] \cdot P[\mathcal{X}_t = s_j | \mathcal{X}_{t-1} = s_i] = \sum_{i=1}^{N} c_i^{t-1} \cdot a_{ij}(t-1)$$

que es el producto del vector c^{t-1} con la columna j-ésima de $A(t-1)$. Aplicando la hipótesis de inducción:

$$c^t = c^{t-1} A(t-1) = c^0 A(0) A(1) \cdots A(t-2) A(t-1). \qquad \square$$

Ejemplo 2.1. En este ejemplo presentamos una variación del juego de cartas "black-jack". En este caso, tenemos un dado de cuatro caras con valores 0, 1, 2 y 3, y con probabilidad uniforme en cada lanzamiento. Un jugador lanza el dado de forma repetida y $\mathcal{X}_t$ representa el valor acumulado tras t lanzamientos. Si el total es igual a nueve, el jugador gana; en otro caso se considera que pierde. Podemos asumir que el resultado de cada lanzamiento es independiente de los lanzamientos anteriores.

Tenemos entonces que $\{\mathcal{X}_t\}$ toma valores en el conjunto $\mathbb{S} := \{0, 1, ..., 8, W, L\}$ de cardinalidad 11. Sea $\mathcal{Y}_t$ el resultado del lanzamiento en el instante t:

$$P[\mathcal{Y}_t = 0] = P[\mathcal{Y}_t = 1] = P[\mathcal{Y}_t = 2] = P[\mathcal{Y}_t = 3] = 1/4$$

Examinemos ahora la variación de $\mathcal{X}_t$: puesto que el valor de $\mathcal{X}_t$ se va acumulando tras cada lanzamiento, tenemos que $\mathcal{X}_t = \mathcal{X}_{t-1} + \mathcal{Y}_t$. En el caso de que $\mathcal{X}_{t-1} + \mathcal{Y}_t = 9$, consideraremos $\mathcal{X}_t = W$ (ganar) y, si $\mathcal{X}_{t-1} + \mathcal{Y}_t > 9$, consideraremos $\mathcal{X}_t = L$ (perder). Si $\mathcal{X}_t = W$ o L, consideraremos que el juego está acabado y $\mathcal{X}_{t+1} = \mathcal{X}_t$. Estas observaciones se pueden resumir en las siguientes reglas:

- Si $\mathcal{X}_{t-1} \leq 5$:

$$P[\mathcal{X}_t = \mathcal{X}_{t-1}] = P[\mathcal{X}_t = \mathcal{X}_{t-1} + 1] = P[\mathcal{X}_t = \mathcal{X}_{t-1} + 2] \\ = P[\mathcal{X}_t = \mathcal{X}_{t-1} + 3] = 1/4.$$

- Si $\mathcal{X}_{t-1} = 6$:

$$P[\mathcal{X}_t = 6] = P[\mathcal{X}_t = 7] = P[\mathcal{X}_t = 8] = P[\mathcal{X}_t = W] = 1/4.$$

- Si $\mathcal{X}_{t-1} = 7$:

$$P[\mathcal{X}_t = 7] = P[\mathcal{X}_t = 8] = P[\mathcal{X}_t = W] = P[\mathcal{X}_t = L] = 1/4.$$

- Si $\mathcal{X}_{t-1} = 8$:

$$P[\mathcal{X}_t = 8] = P[\mathcal{X}_t = W] = 1/4.$$

$$P[\mathcal{X}_t = L] = 1/2.$$

- Si $\mathcal{X}_{t-1} = W$ o L:

$$P[\mathcal{X}_t = \mathcal{X}_{t-1}] = 1.$$

$\{\mathcal{X}_t\}$ es una cadena de Markov pues la distribución de $\mathcal{X}_t$ depende únicamente del valor de $\mathcal{X}_{t-1}$ y no de cómo se ha alcanzado dicho valor. Notemos que las probabilidades anteriores no dependen de t, con lo cual la matriz de transición de $\mathcal{X}_t$ es una matriz fija y $\mathcal{X}_t$ es homogénea.

La matriz de transición de $\mathcal{X}_t$ es entonces una matriz 11×11 dada por:

$$A = \begin{pmatrix}
1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1/4 & 1/4 & 2/4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{pmatrix}$$

Es natural que el juego comience con el valor inicial igual a cero. Por lo tanto, la distribución de $\mathcal{X}_0$ está representada por $c_0 \in \mathbb{R}^{11}$ con un 1 en la primera componente y ceros en el resto. Aplicando repetidamente la fórmula (2.5) obtendremos las distribuciones de $\mathcal{X}_1$, $\mathcal{X}_2$, etc. Así, sea c_t la distribución de $\mathcal{X}_t$, tenemos:

$$c_0 = \begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix},$$

$$c_1 = c_0 A = \begin{pmatrix} 1/4 & 1/4 & 1/4 & 1/4 & 0 & \dots & 0 \end{pmatrix},$$

$$c_2 = c_1 A = \begin{pmatrix} 1/16 & 1/8 & 3/16 & 1/4 & 3/16 & 1/8 & 1/16 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Cabe destacar que, si examinamos la distribución c_t, observamos que $P[\mathcal{X}_t \in \{0 \dots 8\}]$ tiende a cero cuando $t \to \infty$. Esto es natural pues el juego terminará eventualmente en victoria (W) o en pérdida (L) y todos los otros estados son transitorios.

2.2 ESTADOS DE UNA CADENA DE MARKOV

A partir de ahora, vamos a centrarnos en el estudio de cadenas de Markov cuyas matrices de transición son constantes. En consecuencia, las probabilidades de transición son independientes del instante t. Nos referiremos a ellas directamente como cadenas de Markov, asumiendo homogeneidad.

Está claro que los estados juegan un papel importante en el estudio de las cadenas de Markov. Para describir el comportamiento de una cadena de Markov estudiaremos las propiedades de sus estados. Antes de hacer este análisis empezaremos observando cómo evoluciona una cadena de Markov tras n instantes.

Definición 2.6. Sea $\{\mathcal{X}_t\}$ una cadena de Markov, $s_i, s_j \in \mathbb{S}$, $n, m \in \mathbb{N}_0$, denotamos:

$$P_{ij}^{m,m+n} := P[\mathcal{X}_{n+m} = s_j | \mathcal{X}_m = s_i].$$

Si $n = 0$:

$$P_{ij}^{m,m} = P[\mathcal{X}_m = s_j | \mathcal{X}_m = s_i] = \delta_{ij} = \begin{cases} 1, & \text{si } i = j \\ 0, & \text{si } i \neq j \end{cases}$$

Teorema 2.7 (Ecuación de Chapman-Kolmogorov)**.** En condiciones anteriores, sea $r \in \mathbb{N}_0$:

$$P_{ij}^{m,m+n+r} = \sum_{s_k \in \mathbb{S}} P_{ik}^{m,m+n} P_{kj}^{m+n,m+n+r}.$$

Demostración.

$$\begin{aligned} P_{ij}^{m,m+n+r} &= P[\mathcal{X}_{m+n+r} = s_j | \mathcal{X}_m = s_i] \\ &= \sum_{s_k \in \mathbb{S}} P[\mathcal{X}_{m+n+r} = s_j | \mathcal{X}_{m+n} = s_k, \mathcal{X}_m = s_i] P[\mathcal{X}_{m+n} = s_k | \mathcal{X}_m = s_i]. \end{aligned}$$

Aplicando la propiedad de Markov:

$$P[\mathcal{X}_{m+n+r} = s_j | \mathcal{X}_{m+n} = s_k, \mathcal{X}_m = s_i] = P[\mathcal{X}_{m+n+r} = s_j | \mathcal{X}_{m+n} = s_k] = P_{kj}^{m+n,m+n+r}.$$

Por lo tanto:

$$P_{ij}^{m,m+n+r} = \sum_{s_k \in \mathbb{S}} P_{ik}^{m,m+n} P_{kj}^{m+n,m+n+r}. \qquad \square$$

Notemos que por ser $\{\mathcal{X}_t\}$ homogénea, $P_{ij}^{m,m+1}$ es independiente de m, por lo que aplicando inductivamente la ecuación de Chapman-Kolmogorov, tenemos que las probabilidades $P_{ij}^{m,m+n}$ son independientes de m.

Definición 2.8. Sea $\{\mathcal{X}_t\}$ una cadena de Markov, $s_i, s_j \in \mathbb{S}$, $n, m \in \mathbb{N}_0$, se definen las probabilidades de transición en n pasos como:

$$a_{ij}^{(n)} := P_{ij}^{m,m+n} = P[\mathcal{X}_{m+n} = s_j | \mathcal{X}_m = s_i],$$

y la matriz de las probabilidades de transición en n pasos como $A^{(n)} = [a_{ij}^{(n)}]$.

Lema 2.9. La matriz de transición en n pasos cumple que $A^{(n)} = A^n, \forall n \in \mathbb{N}$. Por la observación a la proposición 1.8, $A^{(n)}$ es estocástica.

Demostración. Expresando la ecuación de Chapman-Kolmogorov en forma matricial tenemos que $A^{(n+r)} = A^{(n)} A^{(r)}$. Por ser $A^{(1)} = A$:

$$A^{(n)} = A^{(n-1)} A = A^{(n-2)} A^2 = \cdots = A^n. \qquad \square$$

2.2.1 *Estados accesibles y comunicables*

Definición 2.10. El estado s_j se dice **alcanzable** o **accesible** desde el estado s_i, representado por $i \longrightarrow j$, si existe $n \in \mathbb{N}_0$ tal que $a_{ij}^{(n)} > 0$. Dos estados s_i y s_j mutuamente alcanzables se dice que son **comunicables** y se representa por $i \longleftrightarrow j$.

La definición anterior tiene el significado siguiente: si s_j es accesible desde el estado s_i, entonces existirá un $n \in \mathbb{N}_0$ tal que $P[\mathcal{X}_{m+n} = s_j] > 0$ siempre que $\mathcal{X}_m = s_i$. Es decir, empezando desde el estado s_i, hay una probabilidad positiva de que, en un número finito de transiciones, alcancemos el estado s_j.

Teorema 2.11. La propiedad de comunicación, $\longleftrightarrow$, es una relación de equivalencia sobre el conjunto de estados $\mathbb{S}$.

Demostración.

- **Reflexividad:** $a_{ii}^{(0)} = \delta_{ii} = 1 > 0$. Por tanto, $i \longleftrightarrow i$.

- **Simetría:** si $i \longleftrightarrow j$, existen $n, m \in \mathbb{N}_0$ tales que $a_{ij}^{(n)}, a_{ji}^{(m)} > 0$, escogiendo los mismos n y m, tenemos que $j \longleftrightarrow i$.
- **Transitividad:** sean $i \longleftrightarrow j$ y $j \longleftrightarrow k$:
  - Por ser $i \longleftrightarrow j$, existen $n, m \in \mathbb{N}_0$ tales que $a_{ij}^{(n)}, a_{ji}^{(m)} > 0$.
  - Por ser $j \longleftrightarrow k$, existen $r, s \in \mathbb{N}_0$ tales que $a_{jk}^{(r)}, a_{kj}^{(s)} > 0$.

  Aplicando entonces la ecuación de Chapman-Kolmogorov tenemos que:
  - $a_{ik}^{(n+r)} = \sum_{s_l \in \mathbb{S}} a_{il}^{(n)} a_{lk}^{(r)} \geq a_{ij}^{(n)} a_{jk}^{(r)} > 0$.
  - $a_{ki}^{(s+m)} = \sum_{s_l \in \mathbb{S}} a_{kl}^{(s)} a_{li}^{(m)} \geq a_{kj}^{(s)} a_{ji}^{(m)} > 0$.

  Por lo tanto, $i \longleftrightarrow k$. □

Como resultado, podemos dividir el conjunto de los estados $\mathbb{S}$ en clases de equivalencia atendiendo a las comunicaciones entre estados. Esto nos lleva a la siguiente definición:

Definición 2.12. Sea $\{\mathcal{X}_t\}$ una cadena de Markov sobre un conjunto finito $\mathbb{S}$, $\{\mathcal{X}_t\}$ se dice **irreducible** si hay solo una clase de equivalencia sobre $\mathbb{S}$ mediante la relación $\longleftrightarrow$.

Es decir, una cadena de Markov es irreducible si todos los estados se comunican unos con otros. Por lo tanto, tenemos la siguiente proposición relacionado con las matrices:

Proposición 2.13. Sea $\{\mathcal{X}_t\}$ una cadena de Markov, $\{\mathcal{X}_t\}$ es irreducible si y sólo la matriz de transición asociada A es una matriz irreducible.

Demostración. Si tenemos una cadena irreducible, para todo par de estados s_i, s_j existe $n \in \mathbb{N}$ tal que $a_{ij}^{(n)} > 0$. Aplicando inductivamente la ecuación de Chapman-Kolmogorov, existe al menos una secuencia de productos con n términos $a_{ik} \cdots a_{lj} > 0$. En consecuencia, para todo par de nodos $i, j \in V_A$ existe un camino que los conecta.

Análogamente, si G_A es fuertemente conectada, entonces para todo par de i, j existe una sucesión finita $\{a_{ik}, \ldots, a_{lj}\}$ con todos los elementos mayores que 0. Aplicando la ecuación de Chapman-Kolmogorov, $i \longleftrightarrow j$. □

Ejemplo 2.2. Volviendo a tomar el ejemplo de "blackjack", podemos apreciar que:

- Un estado $s_i \in \{0 \ldots 8\}$ no es comunicable con un estado s_j con valor inferior $\implies a_{ij}^{(n)} = 0, \forall n \in \mathbb{N}_0, j < i$.
- Los estados W y L no son modificables $\implies a_{Wi}^{(n)} = a_{Li}^{(n)} = 0, \forall n \in \mathbb{N}_0, i \in \{0 \ldots 8, W \text{ o } L\}$.

Por simetría, cada estado es únicamente comunicable consigo mismo. En consecuencia, hay 11 clases de equivalencia en $\mathbb{S}$, una por cada estado y la cadena de Markov no es irreducible.

2.2.2 *Periodicidad de una cadena de Markov*

Definición 2.14. Sea $a_{ii}^{(n)}$ la probabilidad de transición en n pasos al estado s_i desde s_i, el periodo $\lambda(i)$ de un estado s_i es el máximo común divisor de todos los $n \in \mathbb{N}$ con $a_{ii}^{(n)} > 0$, esto es:

$$\lambda(i) := m.c.d(\{n \in \mathbb{N} \mid a_{ii}^{(n)} > 0\}).$$

Si $a_{ii}^{(n)} = 0$ para todo $n \in \mathbb{N}$, entonces definimos $\lambda(i) := 0$.

A continuación indicamos que la periodicidad también es una propiedad de clase. Esto es, si el estado s_i en una clase tiene periodo T, entonces todos los estados de esa clase tienen periodo T.

Teorema 2.15. Si $i \longleftrightarrow j$, entonces $\lambda(i) = \lambda(j)$, es decir, el periodo es constante en cada clase de equivalencia.

Demostración ([26]). Si $i = j$ el resultado es trivial. Supongamos que $i \neq j$, entonces existen $n, m \in \mathbb{N}$ tales que $a_{ij}^{(n)}, a_{ji}^{(m)} > 0$, por la ecuación de Chapman-Kolmogorov:

$$a_{ii}^{(n+m)} = \sum_{s_l \in S} a_{il}^{(n)} a_{li}^{(m)} \geq a_{ij}^{(n)} a_{ji}^{(m)} > 0 \Longrightarrow \lambda(i) \mid (n+m).$$

Sea $s \in \mathbb{N}$ tal que $a_{jj}^{(s)} > 0$:

$$a_{ii}^{(n+m+s)} \geq a_{ij}^{(n)} a_{jj}^{(s)} a_{ji}^{(m)} > 0 \Longrightarrow \lambda(i) \mid (n+m+s).$$

Por lo tanto, $\lambda(i) \mid s$. Como s es arbitrario, $\lambda(i)$ es un divisor común de $\{n \in \mathbb{N} \mid a_{jj}^{(n)} > 0\}$ y por definición de periodo, $\lambda(i) \mid \lambda(j)$. Realizando la misma discusión intercambiando los papeles de i y j, obtenemos que $\lambda(j) \mid \lambda(i)$. Como consecuencia, $\lambda(i) = \lambda(j)$. □

Definición 2.16. Un estado s_i se dice **no periódico** o **aperiódico** si $\lambda(i) = 1$.

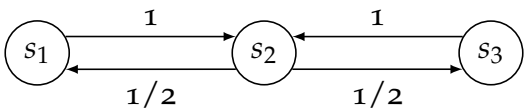
Definición 2.17. Una cadena de Markov se llama **no periódica** o **aperiódica** si todos sus estados son aperiódicos. En caso contrario, se dice **periódica**. Si la cadena de Markov es irreducible, entonces podemos hablar de **periodo de la cadena**.

La periodicidad es una propiedad que se puede apreciar fácilmente si representamos el grafo dirigido asociado a la matriz de transición. Veamos en el siguiente ejemplo:

Ejemplo 2.3. Sea una cadena de Markov con la siguiente matriz de transición:

$$A = \begin{pmatrix} 0 & 1 & 0 \\ 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \end{pmatrix}$$

Si representamos el grafo asociado:



podemos ver que todos los estados tienen periodo 2. Es más, todos los estados son comunicables entre sí luego la cadena es irreducible. En definitiva, tenemos una cadena de Markov periódica con periodo 2.

2.2.3 *Tiempos de transición*

Al inicio de esta sección habíamos visto las probabilidades de transición en n pasos del estado s_i al estado s_j. El número de transiciones necesarias para ir del estado s_i al estado s_j, se denomina **tiempo de transición de s_i hasta s_j**.

Cuando $s_i = s_j$, este tiempo es justo el número de transiciones que se necesita para regresar al estado inicial s_i. En este caso, este tiempo lo denominaremos **tiempo de recurrencia para el estado s_i**.

Definición 2.18. Sea $\{\mathcal{X}_t\}$ una cadena de Markov. La variable aleatoria:

$$\tau_{ij} := \text{mín}\{t \in \mathbb{N} \mid \mathcal{X}_t = s_j \text{ dado } \mathcal{X}_0 = s_i\}$$

considerando mín $\emptyset = \infty$, representa el tiempo mínimo que la cadena necesita para ir desde s_i hasta s_j y se conoce como **tiempo de transición de s_i hasta s_j**. La variable τ_{ii} se llama **tiempo de recurrencia para el estado s_i**.

Definición 2.19. Sea $\{\mathcal{X}_t\}$ una cadena de Markov y $s_i, s_j \in \mathbb{S}$. Para todo $n \in \mathbb{N}$ se define la **probabilidad de tiempo de transición en n pasos** como la probabilidad de que $\{\mathcal{X}_t\}$ alcance s_j en n pasos partiendo desde s_i:

$$f_{ij}^{(0)} := 0,$$
$$f_{ij}^{(n)} := P[\tau_{ij} = n] = P[\mathcal{X}_n = s_j, \mathcal{X}_r \neq s_j, 1 \leq r \leq n-1 | \mathcal{X}_0 = s_i].$$

Cuando $i = j$, hablamos de **probabilidad de tiempo recurrencia en n pasos**.

Como ya se dijo en la definición 2.18, los tiempos de transición son variables aleatorias y sus distribuciones dependen de las probabilidades de transición. En particular, $f_{ij}^{(n)}$ denota la probabilidad de que el tiempo de transición del estado s_i al s_j sea igual a n. Este tiempo de transición es n si la primera transición es del estado s_i a algún estado $s_k \neq s_j$ y el tiempo de transición del estado s_k al estado s_j es $n-1$.

Por lo tanto, estas probabilidades verifican la relación recursiva mostrada en la siguiente proposición:

Proposición 2.20. Las probabilidades de tiempo de transición en n pasos $f_{ij}^{(n)}$, con $n \in \mathbb{N}$, verifican la siguiente relación recursiva:

$$\begin{aligned} f_{ij}^{(1)} &= a_{ij}, \\ f_{ij}^{(n)} &= \sum_{s_k \neq s_j} a_{ik} f_{kj}^{(n-1)}, \ n > 1. \end{aligned}$$

Para s_i y s_j fijos, las $f_{ij}^{(n)}$ son números no negativos tales que $\sum_{n=1}^{\infty} f_{ij}^{(n)} \leq 1$. Si esta suma es estrictamente menor que 1, significa que una cadena que al inicio se encuentra en el estado s_i puede no alcanzar nunca el estado s_j. Cuando la suma sí es igual a 1, las $f_{ij}^{(n)}$ pueden considerarse como la función de masa de probabilidad de la variable aleatoria tiempo de transición τ_{ij}.

Definición 2.21. Sea $\{\mathcal{X}_t\}$ una cadena de Markov, se denomina **probabilidad de tiempo de transición** a la probabilidad de que la cadena alcance s_j empezando por s_i, es decir:

$$f_{ij}^{*} := \sum_{n=1}^{\infty} f_{ij}^{(n)} \leq 1.$$

Si consideramos la variable aleatoria τ_{ij}, entonces la probabilidad de que la cadena nunca alcance s_j, empezando desde s_i, es $P[\tau_{ij} = \infty] = 1 - f_{ij}^{*}$. Y está claro que f_{ii}^{*} es la probabilidad de que la cadena vuelva por lo menos una vez al estado s_i empezando por s_i. Para calcular estas probabilidades, podemos utilizar el siguiente resultado:

Teorema 2.22. Las probabilidades de tiempo de transición f_{ij}^{*} cumplen la ecuación:

$$f_{ij}^{*} = a_{ij} + \sum_{s_k \neq s_j} a_{ik} f_{kj}^{*}. \tag{2.6}$$

Demostración. Por la proposición 2.20:

$$\begin{aligned} f_{ij}^{*} &= \sum_{n=1}^{\infty} f_{ij}^{(n)} = a_{ij} + \sum_{n=2}^{\infty} f_{ij}^{(n)} = a_{ij} + f_{ij}^{(2)} + f_{ij}^{(3)} + \dots \\ &= a_{ij} + \sum_{s_k \neq s_j} a_{ik} f_{kj}^{(1)} + \sum_{s_k \neq s_j} a_{ik} f_{kj}^{(2)} + \dots \\ &= a_{ij} + \sum_{s_k \neq s_j} a_{ik} \sum_{n=1}^{\infty} f_{kj}^{(n)} = a_{ij} + \sum_{s_k \neq s_j} a_{ik} f_{kj}^{*}. \end{aligned}$$

□

Ejemplo 2.4. Volviendo al ejemplo de "blackjack", ya vimos que la cadena convergía a W o L con probabilidad 1. Calculemos ahora la probabilidad de ganar o de perder dado un estado inicial. Es claro que $f_{WW}^{*} = f_{LL}^{*} = 1$ pues una vez que se gana o se

pierde no se modifica más el estado. Por el mismo motivo, tenemos que $f^*_{WL} = f^*_{LW} = 0$. Para calcular las otras probabilidades podemos proceder de forma recursiva sobre los posibles valores de mayor a menor y usando (2.6):

$$f^*_{8W} = a_{8W} + \sum_{s_k \neq W} a_{8k} f^*_{kW} = a_{8W} + a_{88} f^*_{8W} + a_{8L} f^*_{LW}$$
$$= \frac{1}{4} + \frac{1}{4} f^*_{8W} + \frac{2}{4} f^*_{LW} = \frac{1}{4} + \frac{1}{4} f^*_{8W}.$$

Despejando tenemos que:

$$f^*_{8W} = \frac{1}{3}.$$

De forma similar:

$$f^*_{8L} = a_{8L} + \sum_{s_k \neq L} a_{8k} f^*_{kL} = a_{8L} + a_{88} f^*_{8L} + a_{8W} f^*_{WL}$$
$$= \frac{2}{4} + \frac{1}{4} f^*_{8L} + \frac{1}{4} f^*_{WL} = \frac{1}{2} + \frac{1}{4} f^*_{8L}.$$

Despejando:

$$f^*_{8L} = \frac{2}{3}.$$

No es de extrañar que $f^*_{8W} + f^*_{8L} = 1$, pues el juego siempre acabará ganando o perdiendo. Es más, para cualquier estado inicial s_i, es cierto que $f^*_{iW} + f^*_{iL} = 1$. Para calcular la probabilidad de ganar desde el estado 7:

$$f^*_{7W} = a_{7W} + \sum_{s_k \neq W} a_{7k} f^*_{kW} = a_{7W} + a_{77} f^*_{7W} + a_{78} f^*_{8W} + a_{7L} f^*_{LW}$$
$$= \frac{1}{4} \left(1 + \frac{1}{3} + f^*_{7W} \right) = \frac{1}{3} + \frac{1}{4} f^*_{7W}.$$

Despejando:

$$f^*_{7W} = \frac{4}{9}.$$

Si seguimos, obtenemos la siguiente tabla:

i	f^*_{iW}	f^*_{iL}
8	1/3	2/3
7	4/9	5/9
6	16/27	11/27
5	37/81	44/81
4	121/243	122/243
3	376/729	353/729
2	1072/2187	1115/2187
1	3289/6561	3272/6561
0	9889/19683	9794/19683

Podemos ver que la probabilidad de ganar es mayor o menor dependiendo de cada estado inicial. En particular, el estado 6 ofrece la mejor opción para ganar. Esto se debe a que desde el estado 6 no es posible perder en la siguiente ronda pero sí es posible ganar.

Ahora que conocemos la probabilidad de alcanzar un estado s_j desde un estado s_i, nos interesa saber cuánto tarda de media:

Definición 2.23. Sea $\{\mathcal{X}_t\}$ una cadena de Markov, el **tiempo de transición medio** del estado s_i al estado s_j se define por:

$$\mu_{ij} := E[\tau_{ij}] = \begin{cases} \infty, & \text{si } f_{ij}^* < 1 \\ \sum_{n=1}^{\infty} n f_{ij}^{(n)}, & \text{si } f_{ij}^* = 1 \end{cases}$$

Para el caso i=j, se llamará **tiempo medio de recurrencia** para el estado s_i.

Este tiempo representa el número medio de transiciones que se necesita para pasar de un estado s_i a otro estado s_j. Para calcularlo, podemos emplear el siguiente teorema:

Teorema 2.24. Supongamos que para todo $s_k \in \mathbb{S}$, $\mu_{kj} < \infty$. Entonces, si $s_i \in \mathbb{S}$, el tiempo de transición medio μ_{ij} satisface la ecuación:

$$\mu_{ij} = 1 + \sum_{s_k \neq s_j} a_{ik}\mu_{kj}. \tag{2.7}$$

Demostración. Puesto que $\mu_{ij} < \infty$, debe ser $\mu_{ij} = \sum_{n=1}^{\infty} n f_{ij}^{(n)}$. Por la proposición 2.20:

$$\begin{aligned} \mu_{ij} &= a_{ij} + \sum_{n=2}^{\infty} n f_{ij}^{(n)} = a_{ij} + 2f_{ij}^{(2)} + 3f_{ij}^{(3)} + \dots \\ &= a_{ij} + \sum_{s_k \neq s_j} a_{ik} 2 f_{kj}^{(1)} + \sum_{s_k \neq s_j} a_{ik} 3 f_{kj}^{(2)} + \cdots = a_{ij} + \sum_{s_k \neq s_j} a_{ik} \sum_{n=2}^{\infty} n f_{kj}^{(n-1)} \\ &= a_{ij} + \sum_{s_k \neq s_j} a_{ik} \sum_{n=1}^{\infty} (n+1) f_{kj}^{(n)} = a_{ij} + \sum_{s_k \neq s_j} a_{ik} \sum_{n=1}^{\infty} \left(n f_{kj}^{(n)} + f_{kj}^{(n)} \right). \end{aligned}$$

Por hipótesis, $\mu_{kj} < \infty$, luego $\sum_{n=1}^{\infty} n f_{kj}^{(n)}$ es una serie convergente por ser una serie de términos no negativos y mayorada. De mismo modo, por definición de μ_{kj} tenemos que $f_{kj}^* = \sum_{n=1}^{\infty} f_{kj}^{(n)} = 1$ es convergente, y en consecuencia, la serie de sumas

de términos es convergente con $\sum_{n=1}^{\infty} n f_{kj}^{(n)} + \sum_{n=1}^{\infty} f_{kj}^{(n)} = \sum_{n=1}^{\infty} \left(n f_{kj}^{(n)} + f_{kj}^{(n)} \right)$. Empleando esto:

$$\begin{aligned}\mu_{ij} &= a_{ij} + \sum_{s_k \neq s_j} a_{ik} \left(\sum_{n=1}^{\infty} n f_{kj}^{(n)} + \sum_{n=1}^{\infty} f_{kj}^{(n)} \right) = a_{ij} + \sum_{s_k \neq s_j} a_{ik} \left(\mu_{kj} + 1 \right) \\ &= a_{ij} + \sum_{s_k \neq s_j} a_{ik} \mu_{kj} + \sum_{s_k \neq s_j} a_{ik} = \sum_{s_k \in \mathbb{S}} a_{ik} + \sum_{s_k \neq s_j} a_{ik} \mu_{kj} \\ &= 1 + \sum_{s_k \neq s_j} a_{ik} \mu_{kj}. \end{aligned}$$

□

Los conceptos anteriores son extensibles a subconjuntos de $\mathbb{S}$. Si tomamos el ejemplo de "blackjack", podemos considerar el subconjunto $E = \{W, L\}$. Si $\mathcal{X}_t \in E$, implica que el juego ha terminado. Está claro que desde cualquier estado s_i, $f_{iE}^{*} = 1$ y $\mu_{iE} < \infty$. Podemos reescribir (2.7) como:

$$\mu_{iE} = 1 + \sum_{s_k \notin E} a_{ik} \mu_{kE}.$$

Es claro que $\mu_{WE} = \mu_{LE} = 1$. Si $\mathcal{X}_0 = 8$:

$$\mu_{8E} = 1 + a_{88} \mu_{8E} = 1 + \frac{1}{4} \mu_{8E}.$$

Despejando:

$$\mu_{8E} = \frac{4}{3}.$$

Para $\mathcal{X}_0 = 7$:

$$\mu_{7E} = 1 + a_{77} \mu_{7E} + a_{78} \mu_{8E} = 1 + \frac{1}{4} \mu_{7E} + \frac{1}{4} \frac{4}{3} = \frac{4}{3} + \frac{1}{4} \mu_{7E}.$$

Despejando:

$$\mu_{7E} = \frac{16}{9}.$$

Si seguimos procediendo de esta manera, obtenemos la siguiente tabla:

i	μ_{iE}	$\approx$
8	4/3	1.333
7	16/9	1.778
6	64/27	2.370
5	256/81	3.160
4	916/243	3.700
3	3232/729	4.433
2	11200/2187	5.121
1	37888/6561	5.775
0	126820/19683	6.443

que representa el número de rondas que se necesitan para terminar el juego iniciando desde cada estado.

2.2.4 *Estados recurrentes y transitorios*

Vamos a distinguir ahora entre diversos tipos de estados, lo cual nos va a permitir dividir el espacio de estados en varios grupos. Para ello vamos a utilizar la probabilidad de tiempo de transición f_{ij}^*:

Definición 2.25. Un estado $s_i \in \mathbb{S}$ se dice **recurrente** si y sólo si $f_{ii}^* = 1$, en otro caso, se llamará **transitorio**. Si todos los estados son recurrentes, entonces se hablará de una cadena de Markov recurrente, en otro caso, transitoria.

Un estado es transitorio si, después de haber entrado a este estado la cadena puede no regresar nunca a él. Por consiguiente, el estado s_i es transitorio si y sólo si existe un estado $s_j \neq s_i$ que es accesible desde s_i pero no viceversa. Así, si el estado s_i es transitorio y la cadena alcanza dicho estado, existe una probabilidad positiva de que la cadena se mueva al estado s_j y no regrese nunca al estado s_i.

La otra posibilidad es que iniciando en el estado s_i, la cadena siempre regrese a ese estado. En este caso, decimos que el estado es recurrente.

Estas propiedades también se pueden definir utilizando las probabilidades de transición en n pasos:

Proposición 2.26. Sea $s_i \in \mathbb{S}$:

- s_i es recurrente si y sólo si $\sum_{n=1}^{\infty} a_{ii}^{(n)} = \infty$.
- s_i es transitorio si y sólo si $\sum_{n=1}^{\infty} a_{ii}^{(n)} < \infty$.

Demostración. Para demostrarlo, primero veamos la relación que existe entre $a_{ij}^{(n)}$ y $f_{ij}^{(n)}$, recordemos que $a_{ij}^{(n)}$ es la probabilidad de que se produzca una transición de s_i a s_j en n pasos, lo cual incluye posibles transiciones en los pasos $1, 2, 3, \ldots n-1$. De esta forma:

$$a_{ij}^{(1)} = f_{ij}^{(1)},$$
$$a_{ij}^{(2)} = f_{ij}^{(2)} + f_{ij}^{(1)} a_{jj}^{(1)},$$
$$a_{ij}^{(3)} = f_{ij}^{(3)} + f_{ij}^{(2)} a_{jj}^{(1)} + f_{ij}^{(1)} a_{jj}^{(2)}.$$

Así, en general:

$$a_{ij}^{(n)} = \sum_{k=1}^{n} f_{ij}^{(k)} a_{jj}^{(n-k)} = f_{ij}^{(n)} + f_{ij}^{(n-1)} a_{jj}^{(1)} \cdots + f_{ij}^{(1)} a_{jj}^{(n-1)}. \tag{2.8}$$

Usando esto:

$$\sum_{n=1}^{\infty} a_{ii}^{(n)} = \sum_{n=1}^{\infty} \left[f_{ii}^{(n)} + f_{ii}^{(n-1)} a_{ii}^{(1)} \cdots + f_{ii}^{(1)} a_{ii}^{(n-1)} \right]$$
$$= \sum_{n=1}^{\infty} f_{ii}^{(n)} + \sum_{n=1}^{\infty} f_{ii}^{(n)} \sum_{k=1}^{\infty} a_{ii}^{(k)} = f_{ii}^{*} + f_{ii}^{*} \sum_{n=1}^{\infty} a_{ii}^{(n)}.$$

Despejando:

$$\sum_{n=1}^{\infty} a_{ii}^{(n)} = \frac{f_{ii}^{*}}{1 - f_{ii}^{*}}.$$

Luego esta claro que:

- s_i es recurrente $\iff f_{ii}^{*} = 1 \iff \sum_{n=1}^{\infty} a_{ii}^{(n)} = \frac{f_{ii}^{*}}{1 - f_{ii}^{*}} = \infty.$
- s_i es transitorio $\iff f_{ii}^{*} < 1 \iff \sum_{n=1}^{\infty} a_{ii}^{(n)} = \frac{f_{ii}^{*}}{1 - f_{ii}^{*}} < \infty.$

Con esta proposición, ya podemos demostrar que las propiedades de recurrencia y transitoriedad son de clase:

Teorema 2.27. Sean $s_i, s_j \in \mathbb{S}$, si s_i es recurrente e $i \longrightarrow j$, entonces $f_{ji}^{*} = 1$, s_j es recurrente y $f_{ij}^{*} = 1$.

La siguiente demostración se puede consultar en [26, Página 38]:

Demostración. Si $i \longrightarrow j, \exists n \in \mathbb{N}$ tal que $a_{ij}^{(n)} > 0$. Por (2.8), $\exists k \in \{1, \ldots, n\}$ tal que $f_{ij}^{(k)} > 0$, luego $f_{ij}^{*} > 0$. Si fuese $f_{ji}^{*} < 1$, con probabilidad $1 - f_{ji}^{*} > 0$ partiendo desde s_j no pasaríamos nunca por s_i. Así que con probabilidad al menos $f_{ij}^{*}(1 - f_{ji}^{*}) > 0$ saliendo de s_i no volveríamos a pasar nunca por s_i, lo cual contradice con que s_i sea recurrente.

Puesto que $f_{ji}^{*} > 0$, existirá $m \in \mathbb{N}$ tal que $f_{ji}^{(m)} > 0$, por (2.8), $a_{ji}^{(m)} > 0$, luego también $j \longrightarrow i$ e $i \longleftrightarrow j$.

Veamos ahora que s_j ha de ser recurrente: puesto que $i \longleftrightarrow j$, existen $n, m \in \mathbb{N}$ tales que $a_{ij}^{(n)} > 0$ y $a_{ji}^{(m)} > 0$. Para todo $r \geq m + n$ se tiene:

$$a_{jj}^{(r)} \geq a_{ji}^{(m)} a_{ii}^{(r-m-n)} a_{ij}^{(n)}.$$

Por lo tanto:

$$\sum_{r=1}^{\infty} a_{jj}^{(r)} \geq \sum_{r=1}^{n+m} a_{jj}^{(r)} + \sum_{r=n+m+1}^{\infty} a_{ji}^{(m)} a_{ii}^{(r-m-n)} a_{ij}^{(n)} = \sum_{r=1}^{n+m} a_{jj}^{(r)} + a_{ji}^{(m)} a_{ij}^{(n)} \sum_{r=1}^{\infty} a_{ii}^{(r)}.$$

Puesto que s_i es recurrente, $\sum_{r=1}^{\infty} a_{ii}^{(r)} = \infty$ luego también $\sum_{r=1}^{\infty} a_{jj}^{(r)} = \infty$. Por la proposición anterior, s_j es recurrente.

Finalmente, $f_{ij}^{*} = 1$ es clara utilizando que $j \longrightarrow i$, s_j es recurrente y la primera parte de esta demostración. □

De la demostración, podemos apreciar también que si s_i es recurrente y s_j es transitoria entonces no es posible acceder desde s_i a s_j ($i \not\longrightarrow j$). En consecuencia:

Corolario 2.28. Sean $s_i, s_j \in \mathbb{S}$ con $i \longleftrightarrow j$, entonces o son ambos transitorios o son ambos recurrentes.

Vamos a presentar ahora un tipo especial de estado recurrente:

Definición 2.29. Un estado $s_i \in \mathbb{S}$ se llamará **absorbente** si $a_{ii} = 1$.

Si una cadena de Markov ha alcanzado un estado absorbente s_i, permanecerá allí para siempre pues $a_{ij} = 0$ para todo $s_j \neq s_i$. En consecuencia la clase de equivalencia $[s_i]$ estará formado únicamente por s_i.

Con los resultados anteriores seremos capaces de dividir el espacio de estados $\mathbb{S}$ en dos subconjuntos disjuntos: uno constituido por los estados transitorios y otro por los estados recurrentes. Los estados transitorios son inaccesibles desde los recurrentes. Los estados recurrentes se pueden dividir de manera única en clases de equivalencia mediante la relación de equivalencia $i \longleftrightarrow j$. Notemos que si s_i y s_j están en clases distintas entonces $i \not\longrightarrow j$ y $j \not\longrightarrow i$ por el teorema 2.27.

De acuerdo con este resultado, podemos hacer una reordenación de los estados de $\mathbb{S}$ (es decir, una reordenación de las filas y columnas de la matriz de transición) que coloque los estados transitorios al final y agrupe los estados de cada una de las clases de equivalencia de los estados recurrentes. Tendremos así, la siguiente estructura para la matriz de transición de una cadena de Markov:

$$\left(\begin{array}{cccc|c} P_1 & 0 & \dots & 0 & \\ 0 & P_2 & \dots & 0 & \\ \vdots & \vdots & \ddots & \vdots & 0 \\ 0 & 0 & \dots & P_r & \\ \hline \multicolumn{4}{c|}{R} & Q \end{array}\right)$$

donde las submatrices P_i están asociadas a cada clase de equivalencia, R proporciona las probabilidades para pasar desde los estados transitorios a los recurrentes y Q da las probabilidades entre estados transitorios.

2.3 COMPORTAMIENTO ASINTÓTICO DE UNA CADENA DE MARKOV

Aparte de las definiciones del sección anterior, que nos permiten calcular directamente probabilidades relacionadas con los estados, también nos interesa el comportamien-

to de una cadena de Markov a largo plazo. Para ello, vamos a estudiar la matriz de transición en n pasos cuando n tiende a infinito.

En primer lugar, vamos a presentar una distribución especial:

Definición 2.30. Sea A la matriz de transición de una cadena de Markov $\{\mathcal{X}_t\}$, diremos que $\pi \in \Delta^{N-1}$ es una distribución estacionaria si $\pi A = \pi$.

Supongamos que $\{\mathcal{X}_t\}$ es una cadena de Markov con matriz de transición A. Hemos visto que, dependiendo de la distribución inicial, el proceso resultante $\{\mathcal{X}_t\}$ evoluciona de forma distinta a lo largo del tiempo. Pero, si existe una distribución estacionaria π y en un instante t' se da $c^{t'} = (P[\mathcal{X}_{t'} = s_1, \ldots, \mathcal{X}_{t'} = s_N]) = \pi$, entonces $\forall t \in \mathbb{N}, t \geq t', c^t = \pi$.

Notemos también que π es un vector propio a la izquierda de A con valor propio 1, que sabemos que siempre existe por la proposición 1.8. Una distribución estacionaria se puede entender como un punto de equilibrio de la cadena. Es posible que existan varias distribuciones estacionarias pero bajo ciertas condiciones, podemos afirmar que existe una única distribución estacionaria y que la cadena converge hacia ella. Para justificar estas condiciones, necesitamos introducir algunas propiedades de las cadenas irreducibles y aperiódicas (véase [14, Capítulo 4]):

Proposición 2.31. Sea $\{\mathcal{X}_t\}$ una cadena de Markov aperiódica, entonces existe $H \in \mathbb{N}$ tal que $a_{ii}^{(m)} > 0$ para todo estado $s_i \in \mathbb{S}$ y todo número natural $m \geq H$.

Para demostrarlo utilizaremos el siguiente lema de teoría de números:

Lema 2.32. Sea D un conjunto de enteros no negativos tal que:

1. es cerrado para la suma, es decir, si $a, b \in D \implies a + b \in D$.
2. $m.c.d(D) = 1$.

entonces D contiene a todos los enteros no negativos salvo un subconjunto finito. En consecuencia, existe $H \in \mathbb{N}$ tal que para todo número natural $m \geq H$, $m \in D$.

Demostración. Véase [14, Página 26]. □

Demostración (Proposición 2.31)**.** Para cada estado s_i, consideramos:

$$D_i = \{n \in \mathbb{N} \mid a_{ii}^{(n)} > 0\}$$

puesto que la cadena es aperiódica, todos los estados son aperiódicos y $m.c.d(D_i) = 1$. Sean t, s elementos de D_i, luego $a_{ii}^{(t)} > 0$ y $a_{ii}^{(s)} > 0$. Como:

$$a_{ii}^{(t+s)} \geq a_{ii}^{(t)} a_{ii}^{(s)} > 0 \implies t + s \in D_i \implies D_i \text{ es cerrado para la suma}$$

por el lema anterior, existe $H_i \in \mathbb{N}$ tal que $\forall m \geq H_i, m \in D_i$. Puesto que el espacio de estados $\mathbb{S}$ es finito, existe $H = \text{máx}\{H_i \mid i \in \{1, \dots N\}\}$ tal que $\forall m \geq H, m \in \mathbb{N}, a_{ii}^{(m)} > 0, \forall s_i \in \mathbb{S}$. □

Proposición 2.33. Sea $\{\mathcal{X}_t\}$ una cadena de Markov irreducible y aperiódica, entonces existe $M \in \mathbb{N}$ tal que $a_{ij}^{(m)} > 0, \forall s_i, s_j \in \mathbb{S}$ y $\forall m \in \mathbb{N}, m \geq M$.

Demostración. Puesto que la cadena es aperiódica, por la proposición 2.31, existe $H \in \mathbb{N}$ tal que $a_{ii}^{(m)} > 0$ para todo estado $s_i \in \mathbb{S}$ y todo número natural $m \geq H$. Además, como la cadena es irreducible, para todo par de estados s_i, s_j, existe $n_{ij} \in \mathbb{N}$ tal que $a_{ij}^{(n_{ij})} > 0$. Por lo tanto, para $m \geq H + n_{ij}$:

$$a_{ij}^{(m)} \geq a_{ii}^{(m-n_{ij})} a_{ij}^{(n_{ij})} > 0.$$

Puesto que el espacio de los estados es finito, basta elegir $M = H + \text{máx}\{n_{ij} \mid i, j \in \{1, \dots N\}\}$. □

Definición 2.34. Una cadena de Markov $\{\mathcal{X}_t\}$ es **regular** si su matriz de transición A es primitiva. Recordemos que esto quiere decir que existe $k \in \mathbb{N}$ tal que A^k tiene sólo elementos estrictamente positivos.

Como resultado de esta definición y de la proposición 2.33 tenemos el siguiente corolario:

Corolario 2.35. Una cadena de Markov es regular si y sólo si es irreducible y aperiódica.

Demostración. Si la cadena es irreducible y aperiódica, entonces es regular como consecuencia directa de la proposición 2.33.

Supongamos que la cadena es regular, existe entonces $k \in \mathbb{N}$ tal que $a_{ij}^{(k)} > 0, \forall s_i, s_j \in \mathbb{S}$. En consecuencia, todos los estados son comunicables y la cadena es irreducible. Para demostrar que la cadena es aperiódica basta calcular el periodo de un estado s_i ahora que sabemos que es irreducible. Notemos que:

$$a_{ii}^{(3k)} = \sum_{s_j \in \mathbb{S}} a_{ij}^{(k)} a_{jj}^{(k)} a_{ji}^{(k)} > 0.$$

Por otro lado, puesto que $\sum_{s_j \in \mathbb{S}} a_{ij}^{(k+1)} = 1$, existe $a_{ij}^{(k+1)} > 0$. Por lo tanto:

$$a_{ii}^{(3k+1)} = \sum_{s_j \in \mathbb{S}} a_{ij}^{(k+1)} a_{jj}^{(k)} a_{ji}^{(k)} > 0.$$

Puesto $m.c.d(3k, 3k+1) = 1$, tenemos que la cadena es aperiódica. □

Las anteriores propiedades nos proporcionan una condición suficiente para que exista una única distribución estacionaria, que se muestra en el siguiente teorema:

Teorema 2.36. Si $\{\mathcal{X}_t\}$ es una cadena de Markov irreducible, entonces existe una única distribución estacionaria π asociada a $\{\mathcal{X}_t\}$. Si además la cadena es aperiódica, para cada distribución inicial c^0:

$$\lim_{t\to\infty} c^t = \lim_{t\to\infty} c^0 A^t = \pi.$$

Por este teorema, podemos asegurar que si una cadena es irreducible, entonces existe un único punto de equilibrio. Es más, si la cadena es también aperiódica, entonces a largo plazo convergerá hacia dicho equilibrio, sea cual sea la distribución inicial. Para la demostración, utilizaremos los resultados sobre matrices que vimos en el anterior capítulo:

Demostración. Por el teorema 1.14 y las proposiciones 1.8 y 2.13, tenemos que para una cadena de Markov irreducible el vector propio asociado a 1 es único y por lo tanto existe una única distribución estacionaria. Además, por el teorema 1.14, sabemos que cada componente de la distribución estacionaria es estrictamente positiva.

Si además la cadena es aperiódica, entonces regular. Por lo tanto, usando el corolario 1.17 tenemos que:

$$\lim_{t\to\infty} \frac{1}{||c^0 A^t||_1} c^0 A^t = \frac{1}{||\pi||_1}\pi.$$

Puesto que para todo t, $c^t = c^0 A^t \in \Delta^{N-1}$ y $\pi \in \Delta^{N-1}$, sus normas tienen valor 1 y obtenemos la igualdad del teorema 2.36. □

Veamos en el siguiente ejemplo [8, Página 102] la utilidad del teorema 2.36:

Ejemplo 2.5. Supongamos que una ciudad tiene tres cadenas de supermercados $\{A, B, C\}$. Considerando un determinado periodo de tiempo, observamos que, por diferentes razones como el precio, la calidad, etc, algunos habitantes deciden cambiar de cadena.

Para estudiar este cambio a largo plazo, se utiliza la cadena $\{\mathcal{X}_t\}$ = la cadena de supermercado escogida por el cliente en el día t. Se supone también que la proporción de clientes que cambian de supermercado al día es constante con la siguiente matriz de transición:

$$A = \begin{pmatrix} 0{,}8 & 0{,}1 & 0{,}1 \\ 0{,}2 & 0{,}7 & 0{,}1 \\ 0{,}1 & 0{,}3 & 0{,}6 \end{pmatrix}$$

Está claro que la cadena es regular pues la matriz de transición sólo contiene elementos positivos, el vector propio asociado a 1 es:

$$\pi = \begin{pmatrix} 0{,}45 & 0{,}35 & 0{,}2 \end{pmatrix}$$

Lo cual nos indica que a largo plazo, el 45 % de los clientes se quedarán en el supermercado A, el 35 % en el supermercado B y el 20 % en el supermercado C.

Notemos que para este resultado no ha sido necesario saber cual es la proporción de clientes que acuden a cada supermercado en el momento del que se inicia el estudio. Para comprobar la previsión anterior, podemos suponer una distribución inicial alejada de la distribución estacionaria:

$$c^0 = \begin{pmatrix} 0{,}1 & 0{,}2 & 0{,}7 \end{pmatrix}$$

Tras 5 días:

$$c^5 = c^0 A^5 = \begin{pmatrix} 0{,}3995 & 0{,}3848 & 0{,}2156 \end{pmatrix}$$

Tras 14 días:

$$c^{14} = c^0 A^{14} = \begin{pmatrix} 0{,}44937 & 0{,}3506 & 0{,}20003 \end{pmatrix}$$

3

MODELOS DE MARKOV OCULTOS

En este capítulo, estudiaremos un tipo especial de proceso estocástico llamado modelo de Markov oculto (HMM). Empezaremos introduciendo estos modelos, para después seguir discutiendo sobre los problemas y algoritmos que conllevan. En adelante, utilizaremos la abreviatura HMM para referirnos a los modelos de Markov ocultos.

Este capítulo se basa principalmente en [19], [24, Capítulo 2] y [15].

3.1 EXTENSIÓN A MODELOS DE MARKOV OCULTOS

Hasta ahora hemos considerado cadenas de Markov en las cuales cada estado es un evento observable (o material). Este modelo es demasiado restrictivo para aplicar a numerosos problemas en los que no podemos observar directamente los acontecimientos que nos interesan. Para estudiar este tipo de problema extendemos el concepto de cadena de Markov para incluir los casos en los que la observación es una función probabilística del estado. Como resultado, obtenemos un proceso estocástico conjunto formado por una cadena de Markov homogénea que no es observable (es decir, oculta) pero que produce una serie de consecuencias perceptibles mediante otro proceso estocástico. Es decir, tendremos una cadena $\{\mathcal{X}_t\}_{t=0}^{\infty}$ que representa los sucesos ocultos y un proceso $\{\mathcal{Y}_t\}_{t=0}^{\infty}$ que representa las consecuencias observadas de $\{\mathcal{X}_t\}$.

Para aclarar esta idea, consideramos el siguiente ejemplo [21]:

Ejemplo 3.1. Un guardia de seguridad trabaja en una instalación subterránea, sin conexión con el exterior. Cada día, no puede saber si está lloviendo o no, pero por las mañanas ve llegar al director con o sin paraguas.

En este caso, $\mathcal{X}_t$ indica si llueve o no en el día t e $\mathcal{Y}_t$ indica si el director lleva o no paraguas. Está claro que $\mathcal{Y}_t$ es consecuencia directa de $\mathcal{X}_t$ y asumiendo que la posibilidad de que llueva en un día determinado depende únicamente del tiempo del día anterior, tenemos que $\{\mathcal{X}_t\}$ es una cadena de Markov homogénea.

Una representación común de la estructura de HMM es la siguiente:

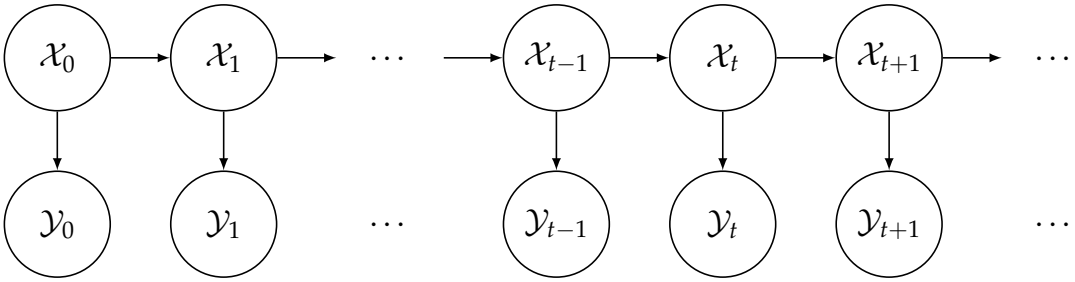


Figura 1: Estructura de un HMM

La representación anterior y el ejemplo 3.1 nos dan una idea de lo que es un HMM. Para concretarlo, damos la siguiente definición:

Definición 3.1. Sean $\{\mathcal{X}_t\}_{t=0}^{\infty}$ e $\{\mathcal{Y}_t\}_{t=0}^{\infty}$ procesos estocásticos que toman valores en conjuntos finitos $\mathbb{S} = \{s_1, \ldots, s_N\}$ y $\mathbb{V} = \{v_1, \ldots, v_M\}$ respectivamente. El proceso conjunto $\{(\mathcal{X}_t, \mathcal{Y}_t)\}$ es un modelo de Markov oculto si:

- $\{\mathcal{X}_t\}$ es una cadena de Markov homogénea.
- $P[\mathcal{Y}_t = y_t | \mathcal{X}_0 = x_0, \ldots, \mathcal{X}_t = x_t, \mathcal{Y}_0 = y_0, \ldots, \mathcal{Y}_{t-1} = y_{t-1}] = P[\mathcal{Y}_t = y_t | \mathcal{X}_t = x_t]$, es decir, la observación en el instante t depende únicamente del estado que se encuentra en dicho momento. Llamaremos a esta probabilidad la probabilidad de emisión.

En distintas fuentes, en lugar de dar una definición explícita de HMM nombran los elementos que lo caracterizan. Puesto que son de enorme importancia, vamos a presentarlos a continuación. Un HMM se caracteriza por:

1. El conjunto de estados $\mathbb{S} = \{s_1, \ldots, s_N\}$ que, a pesar de no ser observables, suelen conllevar un significado físico del problema.
2. El conjunto de posibles observaciones $\mathbb{V} = \{v_1, \ldots, v_M\}$ que corresponden a las salidas materiales del sistema.
3. La matriz de transición A asociada a $\{\mathcal{X}_t\}$ con:
$$a_{ij} = P[\mathcal{X}_{t+1} = s_j | \mathcal{X}_t = s_i].$$
4. La matriz de emisiones $B \in [0,1]^{N \times M}$ estocástica con:
$$b_{jk} = P[\mathcal{Y}_t = v_k | \mathcal{X}_t = s_j] \text{ para todo } t \geq 0.$$
Para reducir la confusión, en adelante utilizaremos la notación $b_{s_j}(v_k)$ para referirnos a estas probabilidades.
5. Una distribución inicial $\pi \in \Delta^{N-1}$ tal que:
$$P[\mathcal{X}_0 = s_i] = \pi_i.$$

Es frecuente (véase [19]) utilizar la notación:

$$\lambda = (A, B, \pi) \tag{3.1}$$

para representar un HMM.

3.2 LOS TRES PROBLEMAS BÁSICOS DE LOS HMMS

A partir de los conceptos anteriores, podemos identificar 3 entidades: el modelo, la secuencia de observaciones o de salidas y la secuencia de estados. Existen 3 problemas básicos de interés que involucran a estas entidades:

1. Dado un modelo, ¿cuál es la probabilidad de observar una secuencia particular de salidas? En este caso no nos interesa la secuencia de estados, tan sólo queremos conocer la probabilidad de que ocurran ciertas observaciones.
2. Dado un modelo y una secuencia de salidas, ¿cuál es la secuencia de estados más probable que genera dichas salidas?
3. Dada una secuencia de salidas y conocido el espacio de estados, ¿cuál es el modelo que maximiza la probabilidad de observar dichas salidas?

El **problema 1** es un problema de evaluación donde calculamos la probabilidad de observar una secuencia de salidas conocido el modelo. También nos permite conocer si el modelo se ajusta a dicha secuencia. Esto puede ser útil, por ejemplo, si estamos considerando varios modelos posibles. En ese caso, la solución del **problema 1** nos permitiría elegir el modelo que más se ajuste a las observaciones.

En el caso del ejemplo 3.1, un ejemplo del **problema 1** podría ser calcular la probabilidad de que el director lleve paraguas dos días seguidos y no en el tercero.

El **problema 2** es donde intentamos cubrir la parte oculta del modelo, es decir, encontrar la secuencia "correcta" de estados. Está claro que en realidad no existe una secuencia "correcta". Por ello, utilizaremos criterios de optimalidad para resolver este problema de la mejor manera posible.

En el caso del ejemplo 3.1, si se observa el paraguas en los dos primeros días y no en el tercero, parece lógico pensar que ha llovido en esos dos primeros días y no en el tercero. Veremos que es efectivamente así resolviendo este problema mediante el algoritmo de Viterbi.

En el **problema 3** pretendemos optimizar los parámetros del modelo dada una secuencia de salidas. La secuencia de observaciones usada para ajustar los parámetros se suele denominar secuencia de entrenamiento. El entrenamiento de HMM es importante, pues así podemos adaptar los parámetros a los datos percibidos. A partir de los resultados, podemos formular mejores modelos para describir fenómenos reales.

Como ejemplo, vamos considerar un problema de reconocimiento de voz, una de las aplicaciones más conocidas de HMM. Podemos utilizar las soluciones del **problema 3** para entrenar un HMM λ_0 (usando la notación (3.1)) que reconoce la pronunciación de la palabra "no" y otro HMM λ_1 que reconoce la pronunciación del "sí". Entonces dada la pronunciación de una palabra desconocida, podemos calcular la probabilidad de dicha pronunciación en cada uno de los dos modelos usando la solución del **problema 1**. Así, podemos determinar si la palabra se asemeja más al "sí" o al "no".

En las siguientes subsecciones vamos a intentar solucionar estos problemas siguiendo principalmente la metodología descrita en [19]. Notemos que, por ser $\{\mathcal{X}_t\}$ homogénea y por el hecho de que $\mathcal{Y}_t$ depende únicamente del estado en el instante t, el instante en el que se comienza a observar las salidas es indiferente. Por lo tanto, podemos suponer siempre que las observaciones inician en el instante $t = 0$.

3.2.1 *Solución al problema 1*

Queremos calcular la probabilidad de una secuencia de observaciones concreta, $O = (O_0, O_1, \ldots, O_r)$ conocido el modelo. La forma más directa de hacerlo es mediante enumeración de todas las posibles secuencias de estados de longitud $r+1$. Consideramos una de ellas:

$$Q = (q_0, q_1, \ldots, q_r) \in \mathbb{S}^{r+1}$$

siendo q_0 el estado inicial. Para facilitar la escritura, introducimos la siguiente notación:

$$\mathcal{Y}_k^l := (\mathcal{Y}_k, \mathcal{Y}_{k+1}, \ldots, \mathcal{Y}_{l-1}, \mathcal{Y}_l).$$

Además, dada la secuencia Q, para representar las probabilidades de transición a partir de los estados de la secuencia pondremos:

$$a_{q_i q_j} = P[\mathcal{X}_{t+1} = q_j | \mathcal{X}_t = q_i].$$

Usando la notación anterior, la probabilidad de la secuencia de observaciones O dada la secuencia de estados Q es:

$$P[\mathcal{Y}_0^r = O | \mathcal{X}_0^r = Q] = \prod_{t=0}^{r} P[\mathcal{Y}_t = O_t | \mathcal{X}_t = q_t]$$

donde aplicamos la independencia entre las observaciones. Por lo tanto:

$$P[\mathcal{Y}_0^r = O | \mathcal{X}_0^r = Q] = b_{q_0}(O_0) \cdot b_{q_1}(O_1) \cdots b_{q_r}(O_r).$$

Y la probabilidad de dicha secuencia de estados Q se puede calcular como:

$$P[\mathcal{X}_0^r = Q] = \pi_{q_0} \cdot a_{q_0 q_1} \cdot a_{q_1 q_2} \cdots a_{q_{r-1} q_r}.$$

Es claro que:

$$P[\mathcal{Y}_0^r = O, \mathcal{X}_0^r = Q] = P[\mathcal{Y}_0^r = O | \mathcal{X}_0^r = Q] \cdot P[\mathcal{X}_0^r = Q].$$

Y la probabilidad de O se puede obtener sumando esta probabilidad mediante todas las posibles secuencias de estados:

$$P[\mathcal{Y}_0^r = O] = \sum_{Q \in \mathbb{S}^{r+1}} P[\mathcal{Y}_0^r = O | \mathcal{X}_0^r = Q] \cdot P[\mathcal{X}_0^r = Q]$$

$$= \sum_{(q_0,q_1,\ldots,q_r)\in \mathbb{S}^{r+1}} \pi_{q_0} \cdot b_{q_0}(O_0) \cdot a_{q_0 q_1} \cdot b_{q_1}(O_1) \cdots a_{q_{r-1} q_r} \cdot b_{q_r}(O_r).$$

Esta manera de calcular, involucra un orden de $2 \cdot (r+1) \cdot N^{(r+1)}$ operaciones, puesto que existen $N^{(r+1)}$ posibles secuencias de estados y para cada una de estas secuencias hay que realizar $2 \cdot (r+1)$ cálculos. Esto hace imposible calcular esta probabilidad, pues incluso para un modelo de 5 estados, si se quiere calcular la probabilidad de una secuencia con 100 observaciones ($r = 99$) se necesitarían $2 \cdot 100 \cdot 5^{100} \approx 10^{72}$ operaciones. Afortunadamente, existe una forma más eficiente de resolver el **problema 1** y es mediante el conocido como **algoritmo de avance-retroceso**.

Definición 3.2. Definimos la **variable de avance** $\alpha_t(i)$ como la probabilidad de observar la secuencia parcial $(O_0, O_1, \ldots, O_t)$ y que el estado en el instante t sea s_i:

$$\alpha_t(i) = P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_t = s_i].$$

En el instante inicial $t = 0$, para todo $i \in \{1, \ldots, N\}$:

$$\alpha_0(i) = P[\mathcal{Y}_0 = O_0, \mathcal{X}_0 = s_i] = P[\mathcal{Y}_0 = O_0 | \mathcal{X}_0 = s_i] \cdot P[\mathcal{X}_0 = s_i] = b_{s_i}(O_0) \cdot \pi_i.$$

Suponiendo que conocemos las $\alpha_t(i)$ para todo i, podemos calcular fácilmente $\alpha_{t+1}(j)$, que es la probabilidad de observar $(O_0, O_1, \ldots, O_{t+1})$ y $\mathcal{X}_{t+1} = s_j$. Puesto que queremos que $\mathcal{X}_{t+1} = s_j$, primero calculamos la probabilidad de mantener la secuencia parcial $(O_0, \ldots, O_t)$ actualizado el estado, esto no es más que la suma de las variables de avance en t multiplicados por las probabilidades de transición:

$$\begin{aligned} &P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_{t+1} = s_j] = \\ &= \sum_{i=1}^{N} P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_t = s_i] \cdot P[\mathcal{X}_{t+1} = s_j | \mathcal{X}_t = s_i] \\ &= \sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}. \end{aligned}$$

Dado que $\mathcal{Y}_{t+1}$ depende únicamente de $\mathcal{X}_{t+1}$, una vez conocida la suma anterior:

$$\begin{aligned} \alpha_{t+1}(j) &= P[\mathcal{Y}_0^{t+1} = (O_0, \ldots, O_t, O_{t+1}), \mathcal{X}_{t+1} = s_j] \\ &= P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_{t+1} = s_j] \cdot P[\mathcal{Y}_{t+1} = O_{t+1} | \mathcal{X}_{t+1} = s_j] \\ &= \left(\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij} \right) \cdot b_{s_j}(O_{t+1}). \end{aligned}$$

Luego podemos calcular las variables de avance de forma recursiva:

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij} \right) \cdot b_{s_j}(O_{t+1}), \qquad 0 \le t \le r-1, \quad 1 \le j \le N. \tag{3.2}$$

Finalmente, notemos que:

$$P[\mathcal{Y}_0^r = O] = \sum_{i=1}^{N} P[\mathcal{Y}_0^r = O, \mathcal{X}_r = s_i] = \sum_{i=1}^{N} \alpha_r(i). \tag{3.3}$$

Si revisamos el cálculo de las variables de avance $\alpha_t(j)$, podemos ver que para cada estado se necesitan $2N$ operaciones en una etapa $t > 0$, puesto que hay N estados. Podemos concluir que el cálculo de todas las variables de avance requiere un orden de $2rN^2$ operaciones. Si $N = 5$ y $r = 99$, necesitaríamos alrededor de 5000 operaciones usando el algoritmo de avance, en comparación con 10^{72} operaciones que requiere el cálculo directo.

A continuación vamos a utilizar el sencillo caso del ejemplo 3.1 para poner en práctica lo que acabamos de ver:

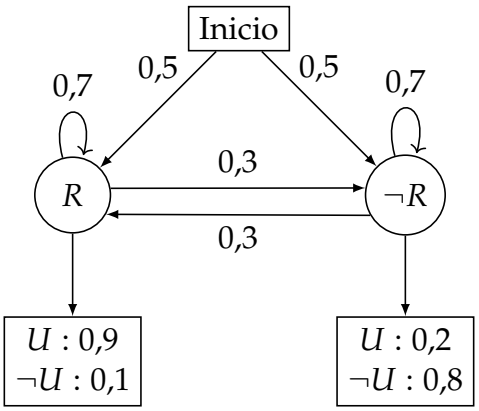
Ejemplo 3.2 ([20])**.** En primer lugar vamos a concretar el modelo: representamos el conjunto de estados como $\mathbb{S} = \{R, \neg R\}$ entendiendo R como lluvia. El conjunto de observaciones también tiene cardinalidad 2: $\mathbb{V} = \{U, \neg U\}$ entendiendo U como presencia de paraguas. Además, vamos a concretar los parámetros del modelo como sigue:

$$A = \begin{pmatrix} 0{,}7 & 0{,}3 \\ 0{,}3 & 0{,}7 \end{pmatrix}$$

$$B = \begin{pmatrix} 0{,}9 & 0{,}1 \\ 0{,}2 & 0{,}8 \end{pmatrix}$$

$$\pi = \begin{pmatrix} 0{,}5 & 0{,}5 \end{pmatrix}$$

Con estos datos, podemos representar el modelo mediante la siguiente gráfica:



donde los nodos circulares representan los estados. Los valores de los arcos desde el inicio a los estados corresponden con las probabilidades de la distribución inicial, los valores entre los estados corresponden con las probabilidades de transición y

los valores de los recuadros rectangulares corresponden con las probabilidades de emisión desde cada estado.

Volviendo a nuestro ejemplo, supongamos que queremos calcular la probabilidad de observar la secuencia $O = (O_0, O_1, O_2) = (U, U, \neg U)$, calculamos las variables de avance hasta $r = 2$ teniendo en cuenta que $s_1 = R$ y $s_2 = \neg R$:

- $t = 0$:

$$\alpha_0(1) = b_{s_1}(O_0) \cdot \pi_1 = 0{,}9 \cdot 0{,}5 = 0{,}45$$
$$\alpha_0(2) = b_{s_2}(O_0) \cdot \pi_2 = 0{,}2 \cdot 0{,}5 = 0{,}1$$

- $t = 1$:

$$\begin{aligned}\alpha_1(1) &= b_{s_1}(O_1) \cdot (\alpha_0(1) \cdot a_{11} + \alpha_0(2) \cdot a_{21}) \\ &= 0{,}9 \cdot (0{,}45 \cdot 0{,}7 + 0{,}1 \cdot 0{,}3) = 0{,}3105\end{aligned}$$
$$\begin{aligned}\alpha_1(2) &= b_{s_2}(O_1) \cdot (\alpha_0(1) \cdot a_{12} + \alpha_0(2) \cdot a_{22}) \\ &= 0{,}2 \cdot (0{,}45 \cdot 0{,}3 + 0{,}1 \cdot 0{,}7) = 0{,}041\end{aligned}$$

- $t = 2$:

$$\begin{aligned}\alpha_2(1) &= b_{s_1}(O_2) \cdot (\alpha_1(1) \cdot a_{11} + \alpha_1(2) \cdot a_{21}) \\ &= 0{,}1 \cdot (0{,}3105 \cdot 0{,}7 + 0{,}041 \cdot 0{,}3) = 0{,}022965\end{aligned}$$
$$\begin{aligned}\alpha_2(2) &= b_{s_2}(O_2) \cdot (\alpha_1(1) \cdot a_{12} + \alpha_1(2) \cdot a_{22}) \\ &= 0{,}8 \cdot (0{,}3105 \cdot 0{,}3 + 0{,}041 \cdot 0{,}7) = 0{,}09748\end{aligned}$$

Así, la probabilidad de que el director lleve paraguas dos días seguidos y no en el tercero es:

$$P[\mathcal{Y}_0^2 = (U, U, \neg U)] = \alpha_2(1) + \alpha_2(2) = 0{,}022965 + 0{,}09748 = 0{,}120445$$

La parte de retroceso del algoritmo no es necesaria para resolver **problema 1**, pero se va a usar en las soluciones a los **problemas 2 y 3**, así que vamos a presentarla aquí.

Definición 3.3. Definimos **la variable de retroceso** $\beta_t(i)$ como la probabilidad de observar la secuencia parcial $(O_{t+1}, O_{t+1}, \ldots, O_r)$ condicionada a que en el instante t, el estado sea s_i. Es decir:

$$\beta_t(i) = P[\mathcal{Y}_{t+1}^r = (O_{t+1}, O_{t+2}, \ldots, O_r) | \mathcal{X}_t = s_i].$$

Puesto que la secuencia de salidas acaba en O_r, $\beta_r(i)$ no se puede determinar usando la definición anterior. En este caso, se define:

$$\beta_r(i) = 1, \qquad \forall i \in \{1, \ldots, N\}.$$

De forma similar a las variables de avance, podemos calcular $\beta_t(i)$ en base a $\beta_{t+1}(j)$. Puesto que conocemos éstos últimos, solo tenemos que preocuparnos por O_{t+1}. De nuevo, dado que $\mathcal{Y}_t$ depende únicamente de $\mathcal{X}_t$:

$$\begin{aligned}&P[\mathcal{Y}_{t+1}^r = (O_{t+1}, O_{t+2}, \ldots, O_r) | \mathcal{X}_{t+1} = s_j] = \\ &= P[\mathcal{Y}_{t+1} = O_{t+1} | \mathcal{X}_{t+1} = s_j] \cdot P[\mathcal{Y}_{t+2}^r = (O_{t+2}, O_{t+3}, \ldots, O_r) | \mathcal{X}_{t+1} = s_j] \\ &= b_{s_j}(O_{t+1}) \cdot \beta_{t+1}(j).\end{aligned}$$

Además, puesto que $\mathcal{X}_{t+1}$ depende de $\mathcal{X}_t$:

$$\begin{aligned}\beta_t(i) &= P[\mathcal{Y}_{t+1}^r = (O_{t+1}, O_{t+2}, \ldots, O_r)|\mathcal{X}_t = s_i] \\ &= \sum_{j=1}^{N} P[\mathcal{X}_{t+1} = s_j|\mathcal{X}_t = s_i] \cdot P[\mathcal{Y}_{t+1}^r = (O_{t+1}, O_{t+2}, \ldots, O_r)|\mathcal{X}_{t+1} = s_j] \\ &= \sum_{j=1}^{N} a_{ij} \cdot b_{s_j}(O_{t+1}) \cdot \beta_{t+1}(j).\end{aligned}$$

Luego también podemos calcular las variables de retroceso de forma recursiva:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \cdot b_{s_j}(O_{t+1}) \cdot \beta_{t+1}(j), \qquad 0 \leq t \leq r-1, \quad 1 \leq i \leq N. \tag{3.4}$$

Para cada estado, se necesitan $3N - 1$ operaciones en una etapa con $0 \leq t \leq r - 1$. Dado que existen N estados, se requiere un orden de $3rN^2$ operaciones para calcular todas las variables de retroceso.

Veremos en los siguientes apartados, que las variables de avance y de retroceso se usarán para resolver los **problemas 2 y 3**.

3.2.2 *Solución al problema 2*

Existen varias formas de resolver el **problema 2** en el que, a diferencia del **problema 1**, no es posible dar una solución exacta. Se trata ahora de encontrar una secuencia de estados "óptima" dada una secuencia de observaciones y un determinado modelo. En primer lugar, debemos definir lo que es una secuencia de estados óptima. Existen varios criterios de optimalidad, uno de ellos consiste en escoger estados que son más probables individualmente. Con este criterio se pretende maximizar el número estimado de estados individuales correctos. Para implementar esta solución al **problema 2**, definimos la siguiente variable:

$$\gamma_t(i) = P[\mathcal{X}_t = s_i|\mathcal{Y}_0^r = (O_0, O_1, \ldots, O_r)]$$

que es la probabilidad de que el estado en el instante t sea s_i condicionada a observar la secuencia de salidas completa $O = (O_0, O_1, \ldots, O_r)$.

Proposición 3.4. $\gamma_t(i)$ se puede expresar en función de las variables de avance y de retroceso de la siguiente forma:

$$\gamma_t(i) = \frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j) \cdot \beta_t(j)}.$$

Demostración. Por definición de las variables:

$$\begin{aligned}\alpha_t(i) \cdot \beta_t(i) &= P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_t = s_i] \cdot P[\mathcal{Y}_{t+1}^r = (O_{t+1}, O_{t+2}, \ldots, O_r) | \mathcal{X}_t = s_i] \\ &= P[\mathcal{Y}_0^r = (O_0, O_1, \ldots, O_r), \mathcal{X}_t = s_i].\end{aligned}$$

Por lo tanto:

$$\begin{aligned}\frac{\alpha_t(i) \cdot \beta_t(i)}{\sum_{j=1}^{N} \alpha_t(j) \cdot \beta_t(j)} &= \frac{P[\mathcal{Y}_0^r = (O_0, O_1, \ldots, O_r), \mathcal{X}_t = s_i]}{\sum_{j=1}^{N} P[\mathcal{Y}_0^r = (O_0, O_1, \ldots, O_r), \mathcal{X}_t = s_j]} \\ &= \frac{P[\mathcal{Y}_0^r = (O_0, O_1, \ldots, O_r), \mathcal{X}_t = s_i]}{P[\mathcal{Y}_0^r = (O_0, O_1, \ldots, O_r)]}.\end{aligned}$$

Aplicando la definición de probabilidad condicionada tenemos la igualdad del enunciado. □

Usando estas variables, podemos definir los estados más probables individualmente dada la secuencia de observaciones O:

Definición 3.5. Sea la secuencia de observaciones $O = (O_0, O_1, \ldots, O_r)$, definimos el estado más probable individualmente en el instante t como:

$$q_t = \underset{1 \leq i \leq N}{\arg\max}\{\gamma_t(i)\} = \{s_i \in \mathbb{S} \mid \forall s_j \in \mathbb{S} : \gamma_t(j) \leq \gamma_t(i)\}. \tag{3.5}$$

A pesar de que (3.5) maximiza el número estimado de estados correctos, puede haber problemas con la secuencia de estados resultante. Por ejemplo, si existen estados inalcanzables desde una de ellas, la secuencia de estados "óptima" puede ser inválida. Esto se debe a que la solución proporcionada por (3.5) sólo determina los estados más probables en cada instante, sin tener en cuenta la probabilidad de existencia de la secuencia resultante en ningún momento.

Una posible solución a este problema consiste en modificar el criterio. Se pueden considerar secuencias de estados que maximizan el número estimado de parejas (q_t, q_{t+1}) o de ternas (q_t, q_{t+1}, q_{t+2}) de estados correctas. Estos criterios pueden ser razonables para ciertas aplicaciones concretas, pero el criterio más utilizado es el de encontrar la secuencia $Q = (q_0, q_1, \ldots, q_r)$ que maximiza $P[\mathcal{X}_0^r = Q | \mathcal{Y}_0^r = O]$. Lo cual es equivalente a maximizar $P[\mathcal{X}_0^r = Q, \mathcal{Y}_0^r = O]$.

Una técnica formal para encontrar dicha secuencia Q existe, se basa en métodos de programación dinámica y se llama **algoritmo de Viterbi**. En primer lugar definimos la variable de Viterbi:

$$\begin{aligned}\delta_t(i) :&= \max_{(q_0, q_1 \cdots, q_{t-1}) \in \mathbb{S}^t} P[\mathcal{X}_0^{t-1} = (q_0, q_1, \ldots, q_{t-1}), \mathcal{X}_t = s_i, \mathcal{Y}_0^t = (O_0, \ldots, O_t)] \\ &= \max_{(q_0, q_1 \cdots, q_{t-1}) \in \mathbb{S}^t} P[\mathcal{X}_0^t = (q_0, q_1, \ldots, q_{t-1}, s_i), \mathcal{Y}_0^t = (O_0, \ldots, O_t)].\end{aligned}$$

En cada instante, $\delta_t(i)$ nos proporciona la probabilidad de la secuencia de estados más probable $(q_0, q_1, \ldots, q_{t-1}, q_t)$ de longitud $t+1$ con $q_t = s_i$, habiendo además observado $(O_0, \ldots, O_t)$.

En el instante inicial $t = 0$, para todo $i \in \{1, \ldots, N\}$:

$$\delta_0(i) = P[\mathcal{X}_0 = s_i, \mathcal{Y}_0 = O_0] = P[\mathcal{X}_0 = s_i] \cdot P[\mathcal{Y}_0 = O_0 | \mathcal{X}_0 = s_i] = b_{s_i}(O_0) \cdot \pi_i.$$

Notemos que cada $\delta_t(i)$ con $t > 0$ se puede calcular recursivamente en función de $\delta_{t-1}(j), 1 \leq j \leq N$. La idea de la recursividad se debe a la propiedad de Markov, pues la secuencia más probable de estados hasta llegar a $\mathcal{X}_t = s_i$ está compuesta por la secuencia más probable $(q_0, q_1 \ldots, q_{t-1})$ con q_{t-1} igual a un cierto estado s_j y la transición de s_j a s_i. Esta idea se puede observar con la Figura 2.

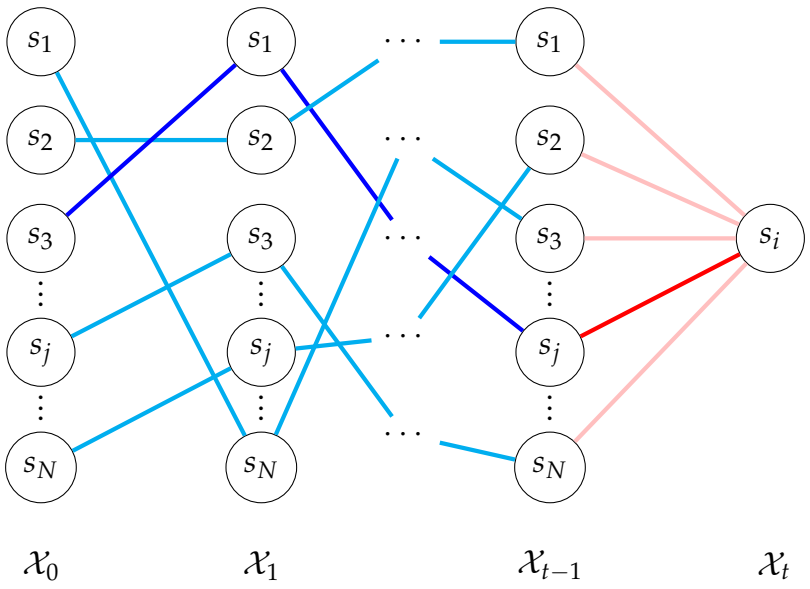


Figura 2: Recursión de algoritmo de Viterbi [20]

Por lo tanto, tenemos que hallar las secuencias más probables hasta $t-1$ y ver desde cuál se obtiene la probabilidad máxima dando un paso más:

$$\begin{aligned}\delta_t(i) &= \max_{(q_0,\ldots,q_{t-1}) \in \mathbb{S}^t} P[\mathcal{X}_0^{t-1} = (q_0, \ldots, q_{t-1}), \mathcal{X}_t = s_i, \mathcal{Y}_0^t = (O_0, \ldots, O_t)] \\ &= \max_{(q_0,\ldots,q_{t-1}) \in \mathbb{S}^t} \big(P[\mathcal{X}_0^{t-1} = (q_0, \ldots, q_{t-1}), \mathcal{X}_t = s_i, \mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1})] \\ &\quad \cdot P[\mathcal{Y}_t = O_t | \mathcal{X}_0^{t-1} = (q_0, \ldots, q_{t-1}), \mathcal{X}_t = s_i, \mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1})]\big).\end{aligned}$$

Aplicando que $\mathcal{Y}_t$ depende únicamente de $\mathcal{X}_t$:

$$\begin{aligned}\delta_t(i) = \max_{(q_0,\ldots,q_{t-1}) \in \mathbb{S}^t} &P[\mathcal{X}_0^{t-1} = (q_0, \ldots, q_{t-1}), \mathcal{X}_t = s_i, \mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1})] \\ &\cdot P[\mathcal{Y}_t = O_t | \mathcal{X}_t = s_i].\end{aligned}$$

Utilizando la idea de recursión:

$$\begin{aligned}
&\max_{(q_0,\ldots,q_{t-1})\in S^t} P[\mathcal{X}_0^{t-1} = (q_0,\ldots,q_{t-1}), \mathcal{X}_t = s_i, \mathcal{Y}_0^{t-1} = (O_0,\ldots,O_{t-1})] \\
&= \max_{j=1,\ldots,N} \big(P[\mathcal{X}_t = s_i | \mathcal{X}_{t-1} = s_j] \cdot \max_{(q_0,\ldots,q_{t-2})\in S^t} P[\mathcal{X}_0^{t-2} = (q_0,\ldots,q_{t-2}), \\
&\quad \mathcal{X}_{t-1} = s_j, \mathcal{Y}_0^{t-1} = (O_0,\ldots,O_{t-1})]\big) \\
&= \max_{j=1,\ldots,N} \left(a_{ji} \cdot \delta_{t-1}(j)\right).
\end{aligned}$$

Por lo tanto:

$$\delta_t(i) = \max_{j=1,\ldots,N} \left(a_{ji} \cdot \delta_{t-1}(j)\right) \cdot b_{s_i}(O_t). \tag{3.6}$$

Además, necesitamos tener constancia de los estados que forman parte de la secuencia óptima. Para ello, definimos $\psi_t(i)$ como el estado en el instante $t-1$ que maximiza $\delta_t(i)$. Resumiendo:

- En el instante inicial:

$$\delta_0(i) = b_{s_i}(O_0) \cdot \pi_i, \qquad 1 \leq i \leq N.$$

- Aplicando recursión:

$$\begin{aligned}
\delta_t(i) &= \max_{j=1,\ldots,N} \left(a_{ji} \cdot \delta_{t-1}(j)\right) \cdot b_{s_i}(O_t), && 1 \leq t \leq r, \\
\psi_t(i) &= \arg\max_{j=1,\ldots,N} \left(a_{ji} \cdot \delta_{t-1}(j)\right), && 1 \leq i \leq N.
\end{aligned}$$

- Finalmente:

$$\begin{aligned}
P^* &= \max_{i=1,\ldots,N} \delta_r(i), \\
q_r^* &= \arg\max_{i=1,\ldots,N} \delta_r(i).
\end{aligned}$$

- Para encontrar la secuencia de estados óptima:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = r-1, r-2, \ldots, 0.$$

Notemos que el esquema recursivo del algoritmo de Viterbi es similar al del algoritmo de avance. La mayor diferencia se encuentra en que, en este caso, se toma el máximo en lugar de calcular una sumatoria.

Ejemplo 3.3 ([20])**.** Bajo los parámetros del ejemplo 3.2, ahora podemos hallar la secuencia de estados más probable habiendo observado el director con paraguas en los dos primeros días y no en el tercero. Recordemos que $O = (O_0, O_1, O_2) = (U, U, \neg U)$ y los estados son $s_1 = L$ y $s_2 = \neg L$:

- $t = 0$:

$$\delta_0(1) = b_{s_1}(O_0) \cdot \pi_1 = 0{,}9 \cdot 0{,}5 = 0{,}45$$

$$\delta_0(2) = b_{s_2}(O_0) \cdot \pi_2 = 0{,}2 \cdot 0{,}5 = 0{,}1$$

- $t = 1$:

$$\begin{aligned}\delta_1(1) &= b_{s_1}(O_1) \cdot \text{máx}\{a_{11} \cdot \delta_0(1),\, a_{21} \cdot \delta_0(2)\} \\ &= 0{,}9 \cdot \text{máx}\{(0{,}7 \cdot 0{,}45),\, (0{,}3 \cdot 0{,}1)\} \\ &= 0{,}9 \cdot \text{máx}\{\underline{0{,}315}, 0{,}03\} = 0{,}2835 \\ \psi_1(1) &= L\end{aligned}$$

$$\begin{aligned}\delta_1(2) &= b_{s_2}(O_1) \cdot \text{máx}\{a_{12} \cdot \delta_0(1),\, a_{22} \cdot \delta_0(2)\} \\ &= 0{,}2 \cdot \text{máx}\{(0{,}3 \cdot 0{,}45),\, (0{,}7 \cdot 0{,}1)\} \\ &= 0{,}2 \cdot \text{máx}\{\underline{0{,}135}, 0{,}07\} = 0{,}027 \\ \psi_1(2) &= L\end{aligned}$$

- $t = 2$:

$$\begin{aligned}\delta_2(1) &= b_{s_1}(O_2) \cdot \text{máx}\{a_{11} \cdot \delta_1(1),\, a_{21} \cdot \delta_1(2)\} \\ &= 0{,}1 \cdot \text{máx}\{(0{,}7 \cdot 0{,}2835),\, (0{,}3 \cdot 0{,}027)\} \\ &= 0{,}1 \cdot \text{máx}\{\underline{0{,}19845}, 0{,}0081\} = 0{,}019845 \\ \psi_2(1) &= L\end{aligned}$$

$$\begin{aligned}\delta_2(2) &= b_{s_2}(O_2) \cdot \text{máx}\{a_{12} \cdot \delta_1(1),\, a_{22} \cdot \delta_1(2)\} \\ &= 0{,}8 \cdot \text{máx}\{(0{,}3 \cdot 0{,}2835),\, (0{,}7 \cdot 0{,}027)\} \\ &= 0{,}8 \cdot \text{máx}\{\underline{0{,}08505}, 0{,}0189\} = 0{,}06804 \\ \psi_2(2) &= L\end{aligned}$$

Puesto que $\delta_2(2) > \delta_2(1)$, tomamos $q_2^* = s_2 = \neg L$, reconstruyendo la secuencia tenemos que:

$$q_1^* = \psi_2(2) = L = s_1, \quad q_0^* = \psi_1(1) = L.$$

Luego la secuencia de estados resultante de aplicar el algoritmo de Viterbi es $Q = (L, L, \neg L)$, lo cual coincide con nuestra intuición.

3.2.3 *Solución al problema 3*

El tercer problema, y el más complicado, consiste en tratar de determinar un método para ajustar los parámetros del modelo de manera que se maximice la probabilidad de una secuencia de observaciones dada.

Sin embargo, no existe ninguna forma analítica de resolver este problema de optimización. Dada una secuencia finita de observaciones $O = (O_0, \ldots, O_r)$, no existe una manera óptima de estimar los parámetros del modelo.

No obstante, podemos elegir $\lambda = (A, B, \pi)$ de forma que, con estos parámetros, $P[\mathcal{Y}_0^r = O]$ se maximice de forma local. Para determinar estos parámetros vamos a utilizar un algoritmo llamado **algoritmo de Baum-Welch**. En primer lugar, dado un par de estados $s_i, s_j \in \mathrm{S}$, definimos:

$$\xi_t(i,j) := P[\mathcal{X}_t = s_i, \mathcal{X}_{t+1} = s_j | \mathcal{Y}_0^r = O], \qquad 0 \leq t \leq r-1.$$

que es la probabilidad de que los estados en los instantes t y $t+1$ sean s_i y s_j respectivamente, condicionada a observar la secuencia de salidas completa $O = (O_0, O_1, \ldots, O_r)$.

Proposición 3.6. $\xi_t(i,j)$ se puede expresar en función de las variables de avance y de retroceso de la siguiente forma:

$$\xi_t(i,j) = \frac{\alpha_t(i) \cdot a_{ij} \cdot b_{s_j}(O_{t+1}) \cdot \beta_{t+1}(j)}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) \cdot a_{ij} \cdot b_{s_j}(O_{t+1}) \cdot \beta_{t+1}(j)}. \tag{3.7}$$

Demostración. Aplicando la definición de probabilidad condicionada:

$$\xi_t(i,j) = \frac{P[\mathcal{X}_t = s_i, \mathcal{X}_{t+1} = s_j, \mathcal{Y}_0^r = O]}{P[\mathcal{Y}_0^r = O]}.$$

Veamos que los numeradores coinciden. Por definiciones de las variables de avance y de retroceso:

$$\begin{aligned}
\alpha_t(i) \cdot a_{ij} \cdot b_{s_j}(O_{t+1}) \cdot \beta_{t+1}(j) &= P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_t = s_i] \cdot P[\mathcal{X}_{t+1} = s_j | \mathcal{X}_t = s_i] \\
&\quad \cdot P[\mathcal{Y}_{t+1} = O_{t+1} | \mathcal{X}_{t+1} = s_j] \\
&\quad \cdot P[\mathcal{Y}_{t+2}^r = (O_{t+2}, \ldots, O_r) | \mathcal{X}_{t+1} = s_j] \\
&= P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_t = s_i, \mathcal{X}_{t+1} = s_j, \\
&\quad \mathcal{Y}_{t+1} = O_{t+1}, \mathcal{Y}_{t+2}^r = (O_{t+2}, \ldots, O_r)] \\
&= P[\mathcal{Y}_0^r = (O_0, \ldots, O_r), \mathcal{X}_t = s_i, \mathcal{X}_{t+1} = s_j].
\end{aligned}$$

Teniendo esto, podemos ver que los denominadores también coinciden:

$$\begin{aligned}
P[\mathcal{Y}_0^r = O] &= \sum_{i=1}^{N} \sum_{j=1}^{N} P[\mathcal{Y}_0^r = O, \mathcal{X}_t = s_i, \mathcal{X}_{t+1} = s_j] \\
&= \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i) \cdot a_{ij} \cdot b_{s_j}(O_{t+1}) \cdot \beta_{t+1}(j).
\end{aligned}$$

□

Recordemos que previamente habíamos definido la variable $\gamma_t(i)$ como:

$$\gamma_t(i) = P[\mathcal{X}_t = s_i | \mathcal{Y}_0^r = O].$$

No es difícil ver a partir de las definiciones, que:

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j), \qquad 0 \leq t \leq r-1.$$

Si sumamos $\gamma_t(i)$ con t desde 0 hasta r, obtenemos una cantidad que se puede interpretar como la cantidad esperada de visitas al estado s_i. Si excluimos $t = r$, se puede interpretar como el número esperado de transiciones que se realizan a partir de s_i. De forma similar, la suma de $\xi_t(i,j)$ con t de 0 a $r-1$ se puede interpretar como el número esperado de transiciones de s_i a s_j. En resumen:

$$\sum_{t=0}^{r} \gamma_t(i) = \text{número esperado de visitas a } s_i.$$

$$\sum_{t=0}^{r-1} \gamma_t(i) = \text{número esperado de transiciones desde } s_i.$$

$$\sum_{t=0}^{r-1} \xi_t(i,j) = \text{número esperado de transiciones de } s_i \text{ a } s_j.$$

Usando lo anterior, podemos dar un método razonable para reestimar los parámetros de un HMM. Definimos:

$$\overline{\pi}_i = \gamma_0(i) = P[\mathcal{X}_0 = s_i | \mathcal{Y}_0^r = O].$$

$$\begin{aligned} \overline{a}_{ij} &= \frac{\text{número esperado de transiciones de } s_i \text{ a } s_j}{\text{número esperado de transiciones desde } s_i} \\ &= \frac{\sum_{t=0}^{r-1} \xi_t(i,j)}{\sum_{t=0}^{r-1} \gamma_t(i)}. \end{aligned}$$

$$\begin{aligned} \overline{b}_j(k) &= \frac{\text{número esperado de visitas a } s_j \text{ habiendo observado } v_k}{\text{número esperado de visitas a } s_j} \\ &= \frac{\sum_{\substack{t=0 \\ O_t = v_k}}^{r} \gamma_t(j)}{\sum_{t=0}^{r} \gamma_t(j)}. \end{aligned}$$

Si definimos el modelo actual como $\lambda = (A, B, \pi)$, y lo utilizamos para calcular los parámetros anteriores, podemos definir un modelo reestimado $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$. Entonces está probado que puede ocurrir una de las siguientes opciones [19]:

- el modelo inicial λ es un punto crítico de la función de probabilidad, en ese caso $\overline{\lambda} = \lambda$.
- la probabilidad de observar la secuencia de salidas O bajo el modelo $\overline{\lambda}$ es mayor que bajo λ. Es decir, $P[\mathcal{Y}_0^r = O|\overline{\lambda}] > P[\mathcal{Y}_0^r = O|\lambda]$.

Con el proceso anterior, podemos iterar reemplazando $\overline{\lambda}$ por λ. De esta forma, podemos aumentar la probabilidad de observar O hasta llegar a un valor límite. Cabe destacar que este algoritmo es susceptible de caer en un máximo local y en diversos problemas de interés, pueden existir varios máximos locales.

Recordemos que el algoritmo de Baum-Welch se utiliza para reestimar los parámetros de $\lambda = (A, B, \pi)$ con la intención de maximizar $P[\mathcal{Y}_0^r = O|\lambda]$ con $O = (O_0, \cdots, O_r)$ una secuencia de observaciones conocida. Pero también podemos usarlo para determinar los estados de un espacio $\mathbb{S}$ al que sólo se conoce la cardinalidad del espacio: ante problemas en los que no tenemos suficiente información, podemos partir de un modelo inicial estimado y aplicar el algoritmo de Baum-Welch para construir un modelo ajustado mediante una secuencia de observaciones suficientemente amplia. A partir de los resultados, podemos dar un significado físico a los estados del espacio $\mathbb{S}$. Como ejemplo, examinamos el siguiente caso:

Ejemplo 3.4. Consideramos el ejemplo ilustrado en [24], en el que se pretende estudiar las propiedades básicas del inglés. Para ello, toma el libro "Brown Corpus" con alrededor de un millón de palabras, eliminó las puntuaciones, números y caracteres especiales y convirtió todas las letras a minúscula obteniendo 26 posibles letras y el espacio, un total de 27 símbolos que consideró como posibles observaciones.

Con estas 27 posibles observaciones y considerando un espacio de estados $\mathbb{S}$ con cardinalidad $N = 2$, toma las primeras las primeras 50000 observaciones y supone que todas las probabilidades se distribuyen de forma aleatoriamente uniforme. En especial, asigna a cada probabilidad de emisión un valor aleatorio cercano a $1/27$. Aplicando 100 veces el algoritmo de Baum-Welch obtuvo las siguiente probabilidades de emisión:

	Inicial		Final	
	s_1	s_2	s_1	s_2
a	0,03735	0,03909	0,13845	0,00075
b	0,03408	0,03537	0,00000	0,02311
c	0,03455	0,03537	0,00062	0,05614
d	0,03828	0,03909	0,00000	0,06937
e	0,03782	0,03583	0,21404	0,00000
f	0,03922	0,03630	0,00000	0,03559
g	0,03688	0,04048	0,00081	0,02724
h	0,03408	0,03537	0,00066	0,07278

i	0,03875	0,03816	0,12275	0,00000
j	0,04062	0,03909	0,00000	0,00365
k	0,03735	0,03490	0,00182	0,00703
l	0,03968	0,03723	0,00049	0,07231
m	0,03548	0,03537	0,00000	0,03889
n	0,03735	0,03909	0,00000	0,11461
o	0,04062	0,03397	0,13156	0,00000
p	0,03595	0,03397	0,00040	0,03674
q	0,03641	0,03816	0,00000	0,00153
r	0,03408	0,03676	0,00000	0,10225
s	0,04062	0,04048	0,00000	0,11042
t	0,03548	0,03443	0,01102	0,14392
u	0,03922	0,03537	0,04508	0,00000
v	0,04062	0,03955	0,00000	0,01621
w	0,03455	0,03816	0,00000	0,02303
x	0,03595	0,03723	0,00000	0,00447
y	0,03408	0,03769	0,00019	0,02587
z	0,03408	0,03955	0,00000	0,00110
espacio	0,03688	0,03397	0,33211	0,01298

Tabla 1: Probabilidades de emisión antes y después de aplicar el algoritmo de Baum-Welch [24]

Si analizamos estas probabilidades, podemos ver que los estados separan las vocales de las consonantes, de forma que s_1 representa el conjunto de vocales y s_2 representa el conjunto de consonantes.

Este resultado también se ha alcanzado utilizando la traducción del Quijote a inglés [1] y no debería de sorprender a una persona que tenga conocimiento sobre el idioma. Sin embargo, permite a personas que carecen de ese conocimiento llegar a conclusiones analizando los resultados obtenidos de un HMM asociado al problema.

3.3 MEJORA DE LAS SOLUCIONES

No es difícil darse cuenta de que las variables $\alpha_t(i), \beta_t(i)$ y $\delta_t(i)$ tienden a 0 cuando $t \to \infty$ por ser productos de probabilidades. Para evitar multiplicar por cero, necesitamos tomar medidas que dependerán del problema que estamos tratando de resolver.

3.3.1 *Normalización de $\alpha_t(i)$ y $\beta_t(i)$*

En primer lugar, consideramos el cálculo de $\alpha_t(i)$. Recordemos que por (3.2):

$$\alpha_t(i) = \left(\sum_{j=1}^{N} \alpha_{t-1}(j) \cdot a_{ji} \right) \cdot b_{s_i}(O_t), \qquad 1 \leq t \leq r, \quad 1 \leq i \leq N.$$

Para resolver el problema de multiplicar por $\alpha_t(i)$ cuando éste tiende a 0, parece lógico normalizar $\alpha_t(i)$ dividiéndolo por la suma de las $\alpha_t(j)$ con $j \in \{1, \ldots, N\}$ y utilizar el resultado de la división. Sin embargo, necesitamos verificar que con esta reestimación, no estamos alterando el resultado del algoritmo de avance. Para ello, definimos las siguientes variables:

- Para $t = 0$, consideramos:

$$\tilde{\alpha}_0(i) = \alpha_0(i), \qquad c_0 = \frac{1}{\sum_{j=1}^{N} \tilde{\alpha}_0(j)}, \qquad \hat{\alpha}_0(i) = c_0 \cdot \tilde{\alpha}_0(i), \qquad 1 \leq i \leq N.$$

- Para $1 \leq t \leq r$, $1 \leq i \leq N$:

$$\tilde{\alpha}_t(i) = \left(\sum_{j=1}^{N} \hat{\alpha}_{t-1}(j) \cdot a_{ji} \right) \cdot b_{s_i}(O_t), \qquad c_t = \frac{1}{\sum_{j=1}^{N} \tilde{\alpha}_t(j)}, \qquad \hat{\alpha}_t(i) = c_t \cdot \tilde{\alpha}_t(i).$$

Proposición 3.7. La variable $\hat{\alpha}_t(i)$ cumple que:

$$\hat{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)}, \qquad 0 \leq t \leq r, \qquad 1 \leq i \leq N.$$

Demostración. Por definición $\hat{\alpha}_0(i) = c_0 \cdot \alpha_0(i)$, supongamos que:

$$\hat{\alpha}_t(i) = c_0 \cdot c_1 \cdots c_t \cdot \alpha_t(i). \tag{3.8}$$

Entonces:

$$\begin{aligned}
\hat{\alpha}_{t+1}(i) &= c_{t+1} \cdot \tilde{\alpha}_{t+1}(i) \\
&= c_{t+1} \cdot \left(\sum_{j=1}^{N} \hat{\alpha}_t(j) \cdot a_{ji} \right) \cdot b_{s_i}(O_{t+1}) \\
&= c_0 \cdot c_1 \cdots c_t \cdot c_{t+1} \cdot \left(\sum_{j=1}^{N} \alpha_t(j) \cdot a_{ji} \right) \cdot b_{s_i}(O_{t+1}) \\
&= c_0 \cdot c_1 \cdots c_t \cdot c_{t+1} \cdot \alpha_{t+1}(i).
\end{aligned}$$

Luego por inducción, (3.8) es cierto para todo $t \leq r$. En consecuencia, por (3.8) y definición de $\hat{\alpha}_t(i)$ y $\tilde{\alpha}_t(i)$:

$$\tilde{\alpha}_t(i) = \frac{\hat{\alpha}_t(i)}{c_t} = \frac{c_0 \cdots c_t \cdot \alpha_t(i)}{c_t} = c_0 \cdots c_{t-1} \cdot \alpha_t(i),$$
$$\hat{\alpha}_t(i) = \frac{\tilde{\alpha}_t(i)}{\sum_{j=1}^{N} \tilde{\alpha}_t(j)} = \frac{c_0 \cdots c_{t-1} \cdot \alpha_t(i)}{c_0 \cdots c_{t-1} \cdot \sum_{j=1}^{N} \alpha_t(j)} = \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)}.$$ □

Corolario 3.8. Se cumple que:

$$\hat{\alpha}_t(i) = P[\mathcal{X}_t = s_i | \mathcal{Y}_0^t = (O_0, \ldots, O_t)], \qquad 0 \leq t \leq r, \qquad 1 \leq i \leq N.$$

Demostración. Por la proposición 3.7 y la definición de $\alpha_t(i)$:

$$\hat{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)} = \frac{P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_t = s_i]}{\sum_{j=1}^{N} P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_t = s_j]}$$
$$= \frac{P[\mathcal{Y}_0^t = (O_0, \ldots, O_t), \mathcal{X}_t = s_i]}{P[\mathcal{Y}_0^t = (O_0, \ldots, O_t)]} = P[\mathcal{X}_t = s_i | \mathcal{Y}_0^t = (O_0, \ldots, O_t)].$$ □

Por lo tanto, $\hat{\alpha}_t(i)$ es el resultado de dividir $\alpha_t(i)$ por la suma de las $\alpha_t(j)$ con $j \in \{1, \ldots, N\}$ como queríamos. Además, con este proceso hemos evitado calcular directamente $\alpha_t(i)$. Ahora debemos comprobar que utilizando $\hat{\alpha}_t(i)$ podemos calcular $P[\mathcal{Y}_0^r = O]$. Notemos que de la proposición 3.7 tenemos:

$$\sum_{j=1}^{N} \hat{\alpha}_t(j) = 1, \qquad \forall t \in \{0, \ldots, r\}.$$

Entonces, aplicando (3.8):

$$1 = \sum_{j=1}^{N} \hat{\alpha}_r(j) = c_0 \cdot c_1 \cdots c_r \cdot \sum_{j=1}^{N} \alpha_r(j) = c_0 \cdot c_1 \cdots c_r \cdot P[\mathcal{Y}_0^r = O].$$

En consecuencia:

$$P[\mathcal{Y}_0^r = O] = \frac{1}{\prod_{t=0}^{r} c_t}.$$

Para evitar obtener 0 por desbordamiento, podemos aplicar el logaritmo:

$$\log\left(P[\mathcal{Y}_0^r = O]\right) = -\sum_{t=0}^{r} \log c_t.$$

Antes de pasarnos a la normalización de $\beta_t(i)$, vamos a pararnos a examinar la variable c_t. Tal como hemos definido c_t, uno puede pensar que depende de la variable $\tilde{\alpha}_t(i)$ y por lo tanto depende de $\alpha_t(i)$. Pero con la siguiente proposición vamos a ver que c_t es independiente de $\alpha_t(i)$:

Proposición 3.9. La variable c_t cumple que:

$$c_0 = P[\mathcal{Y}_0 = O_0]^{-1},$$
$$c_t = P[\mathcal{Y}_t = O_t | \mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1})]^{-1}, \qquad 1 \leq t \leq r.$$

Demostración. De la proposición 3.7 y (3.8) tenemos que:

$$\hat{\alpha}_t(i) = \frac{\alpha_t(i)}{\sum_{j=1}^{N} \alpha_t(j)} = c_0 \cdot c_1 \cdots c_t \cdot \alpha_t(i). \tag{3.9}$$

Para $t = 0$ tenemos:

$$\hat{\alpha}_0(i) = \frac{\alpha_0(i)}{\sum_{j=1}^{N} \alpha_0(j)} = c_0 \alpha_0(i).$$

Por lo tanto:

$$c_0 = \frac{1}{\sum_{j=1}^{N} \alpha_0(j)} = \frac{1}{\sum_{j=1}^{N} P[\mathcal{Y}_0 = O_0, \mathcal{X}_0 = s_j]} = P[\mathcal{Y}_0 = O_0]^{-1}.$$

Para $t > 0$, utilizamos (3.9):

$$\prod_{k=0}^{t-1} c_k = \frac{1}{\sum_{j=1}^{N} \alpha_{t-1}(j)} = \frac{1}{\sum_{j=1}^{N} P[\mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1}), \mathcal{X}_{t-1} = s_j]}$$
$$= P[\mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1})]^{-1}.$$

Y en consecuencia:

$$c_t = \frac{P[\mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1})]}{\sum_{j=1}^{N} P[\mathcal{Y}_0^{t} = (O_0, \ldots, O_{t}), \mathcal{X}_t = s_j]} = \frac{P[\mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1})]}{P[\mathcal{Y}_0^{t} = (O_0, \ldots, O_{t})]}$$
$$= \left(\frac{P[\mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1}), \mathcal{Y}_t = O_t]}{P[\mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1})]} \right)^{-1} = P[\mathcal{Y}_t = O_t | \mathcal{Y}_0^{t-1} = (O_0, \ldots, O_{t-1})]^{-1}.$$

□

Así, c_t es la inversa de una probabilidad. Luego $c_t \geq 1$ y podemos utilizarla para normalizar $\beta_t(i)$ de mismo modo que con $\alpha_t(i)$. Recordemos que por (3.4):

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} \cdot b_{s_j}(O_{t+1}) \cdot \beta_{t+1}(j), \qquad 0 \leq t \leq r-1, \quad 1 \leq i \leq N.$$

Definimos la variable normalizada $\hat{\beta}_t(i)$ de siguiente manera:

- Para $t = r$, consideramos:

$$\hat{\beta}_r(i) = c_r, \qquad 1 \leq i \leq N.$$

- Para $0 \leq t \leq r-1$:

$$\hat{\beta}_t(i) = c_t \cdot \sum_{j=1}^{N} a_{ij} \cdot b_{s_j}(O_{t+1}) \cdot \hat{\beta}_{t+1}(j), \qquad 1 \leq i \leq N.$$

No es difícil ver que la variable $\hat{\beta}_t(i)$ cumple lo siguiente:

$$\hat{\beta}_t(i) = c_t \cdot c_{t+1} \cdots c_r \cdot \beta_t(i), \qquad 0 \leq t \leq r, \qquad 1 \leq i \leq N. \tag{3.10}$$

En efecto, para $t = r$ está claro que:

$$\hat{\beta}_r(i) = c_r = c_r \cdot \beta_r(i).$$

Supongamos cierto para $t > 0$, veamos que la proposición se cumple para $t-1$:

$$\begin{aligned}
\hat{\beta}_{t-1}(i) &= c_{t-1} \cdot \sum_{j=1}^{N} a_{ij} \cdot b_{s_j}(O_t) \cdot \hat{\beta}_t(j) \\
&= c_{t-1} \cdot \sum_{j=1}^{N} a_{ij} \cdot b_{s_j}(O_t) \cdot [c_t \cdot c_{t+1} \cdots c_r \cdot \beta_t(j)] \\
&= c_{t-1} \cdot c_t \cdot c_{t+1} \cdots c_r \cdot \sum_{j=1}^{N} a_{ij} \cdot b_{s_j}(O_t) \cdot \beta_t(j) \\
&= c_{t-1} \cdot c_t \cdot c_{t+1} \cdots c_r \cdot \beta_{t-1}(i).
\end{aligned}$$

Por lo tanto, $\hat{\beta}_t(i)$ cumple una propiedad inductiva similar a la de $\hat{\alpha}_t(i)$.

En realidad, cualquier otra forma de normalización hubiese sido válida [7]. El motivo por el que utilizamos c_t para normalizar $\beta_t(i)$ es que de esta forma, podemos utilizar las variables normalizadas en (3.7) y en el cálculo de los parámetros reestimados en el algoritmo de Baum-Welch.

Con esta normalización, presentamos los pseudo-códigos que resuelven los **problemas 1** y **3**:

Algoritmo 1 Algoritmo de avance-retroceso normalizado

Input: $\lambda = (A, B, \pi)$
Input: $O = (O_0, \ldots, O_r)$ = secuencia de observaciones
Output: $\alpha_t(i), \beta_t(i), c_t, \forall i \in \{1, \ldots N\}, \forall t \in \{0, \ldots r\}$
Output: $logProb = \log\left(P[\mathcal{Y}_0^r = O|\lambda]\right)$

1: //Calcular $\alpha_0(i)$
2: $c_0 \leftarrow 0$
3: **for** $i \leftarrow 1$ **to** N **do**

```
    α_0(i) = π_i · b_{s_i}(O_0)
    c_0 = c_0 + α_0(i)
end for

//Normalizar α_0(i)
c_0 ← 1/c_0
for i ← 1 to N do
    α_0(i) = c_0 · α_0(i)
end for

//Calcular α_t(i)
for t ← 1 to r do
    c_t ← 0
    for i ← 1 to N do
        α_t(i) ← 0
        for j ← 1 to N do
            α_t(i) ← α_t(i) + α_{t−1}(j) · a_{ji}
        end for
        α_t(i) ← α_t(i) · b_{s_i}(O_t)
        c_t ← c_t + α_t(i)
    end for

    //Normalizar α_t(i)
    c_t ← 1/c_t
    for i ← 1 to N do
        α_t(i) = c_t · α_t(i)
    end for
end for

//Inicializamos β_r(i) con los valores de c_r
for i ← 1 to N do
    β_r(i) = c_r
end for

//Calcular β_t(i)
for t ← r − 1 to 0 by −1 do
    for i ← 1 to N do
        β_t(i) ← 0
        for j ← 1 to N do
            β_t(i) ← β_t(i) + a_{ij} · b_{s_j}(O_{t+1}) · β_{t+1}(j)
        end for

        //Normalizar β_t(i)
```

```
        β_t(i) ← c_t · β_t(i)
    end for
end for

//Calcular log(P[Y_0^r = O])
logProb ← 0
for t ← 0 to r do
    logProb ← logProb + log(c_t)
end for
logProb ← −logProb
```

Algoritmo 2 Algoritmo de Baum-Welch

Input: $\lambda = (A, B, \pi)$
Input: $maxIters$ = número máximo de iteraciones
Input: $O = (O_0, \ldots, O_r)$ = secuencia de entrenamiento
Output: $\overline{\lambda} = (\overline{A}, \overline{B}, \overline{\pi})$

```
iters ← 0
oldLogProb ← −∞
Llamar a algoritmo 1

while iters < maxIters y logProb > oldLogProb do

    //Calcular ξ_t(i, j) y γ_t(i)
    for t ← 0 to r − 1 do
        for i ← 1 to N do
            γ_t(i) ← 0
            for j ← 1 to N do
                ξ_t(i, j) ← α_t(i) · a_ij · b_{s_j}(O_{t+1}) · β_{t+1}(j)
                γ_t(i) ← γ_t(i) + ξ_t(i, j)
            end for
        end for
    end for

    //Para γ_r(i) notemos que coinciden con α_r(i) normalizados
    for i ← 1 to N do
        γ_r(i) ← α_r(i)
    end for

    //Reestimación de π
    for i ← 1 to N do
        π̄_i ← γ_0(i)
    end for
```

```
    //Reestimación de A y B
    for i ← 1 to N do
        denom ← 0
        for t ← 0 to r − 1 do
            denom ← denom + γ_t(i)
        end for

        for j ← 1 to N do
            numerA ← 0
            for t ← 0 to r − 1 do
                numerA ← numerA + ξ_t(i, j)
            end for
            ā_ij ← numerA/denom
        end for

        denom ← denom + γ_r(i)
        for k ← 1 to M do
            numerB ← 0
            for t ← 0 to r do
                if O_t == v_k then
                    numerB ← numerB + γ_t(i)
                end if
            end for
            b̄_i(k) ← numerB/denom
        end for
    end for

    λ ← λ̄
    oldLogProb ← logProb
    Llamar a algoritmo 1
    iters ← iters + 1
end while

return λ̄ ← λ
```

3.3.2 *Mejora del algoritmo de Viterbi*

En el caso del algoritmo de Viterbi, podemos tomar el logaritmo de $\delta_t(i)$. Al ser el logaritmo una función creciente, no afecta al cálculo de los máximos. De manera que:

- En el instante inicial:

$$\hat{\delta}_0(i) = \log(b_{s_i}(O_0) \cdot \pi_i), \qquad 1 \leq i \leq N.$$

- Aplicando recursión:

$$\hat{\delta}_t(i) = \max_{j=1,\dots,N} \left(\hat{\delta}_{t-1}(j) + \log(a_{ji})\right) + \log(b_{s_i}(O_t)), \qquad 1 \leq t \leq r,$$
$$\hat{\psi}_t(i) = \arg\max_{j=1,\dots,N} \left(\hat{\delta}_{t-1}(j) + \log(a_{ji})\right), \qquad 1 \leq i \leq N.$$

- Finalmente:

$$\hat{P}^* = \max_{i=1,\dots,N} \hat{\delta}_r(i),$$
$$\hat{q}_r^* = \arg\max_{i=1,\dots,N} \hat{\delta}_r(i).$$

- Para encontrar la secuencia de estados óptima:

$$\hat{q}_t^* = \hat{\psi}_{t+1}(\hat{q}_{t+1}^*), \quad t = r-1, r-2, \dots, 0.$$

Así, tenemos el siguiente pseudo-código que resuelve el **problema 2**:

Algoritmo 3 Algoritmo de Viterbi

Input: $\lambda = (A, B, \pi)$
Input: $O = (O_0, \dots, O_r)$ = secuencia de observaciones
Output: $Q = (q_0, \dots, q_r)$ = secuencia de estados que maximiza $P[\mathcal{X}_0^r = Q | \mathcal{Y}_0^r = O]$

1: //Calcular $\delta_0(i)$
2: **for** $i \leftarrow 1$ **to** N **do**
3: $\quad \delta_0(i) \leftarrow \log(b_{s_i}(O_0) \cdot \pi_i)$
4: **end for**
5:
6: //Calcular $\delta_t(i)$ y $\psi_t(i)$
7: **for** $t \leftarrow 1$ **to** r **do**
8: $\quad$ **for** $i \leftarrow 1$ **to** N **do**
9: $\quad\quad \delta_t(i) \leftarrow \max_{j=1,\dots,N} \left(\delta_{t-1}(j) + \log(a_{ji})\right) + \log(b_{s_i}(O_t))$
10: $\quad\quad \psi_t(i) \leftarrow \arg\max_{j=1,\dots,N} \left(\delta_{t-1}(j) + \log(a_{ji})\right)$
11: $\quad$ **end for**
12: **end for**
13: //Obtener la secuencia óptima
14: $q_r = \arg\max_{i=1,\dots,N} \delta_r(i)$
15: **for** $t \leftarrow r-1$ **to** 0 **by** -1 **do**
16: $\quad q_t \leftarrow \psi_{t+1}(q_{t+1})$
17: **end for**

4

APLICACIONES A LA BIOLOGÍA

En este capítulo, introduciremos algunos problemas de bioinformática que pueden ser resueltos mediante el uso de HMM. Empezaremos presentado algunos conceptos básicos de biología relacionados con nuestro estudio, discutiremos sobre la naturaleza y la importancia que tienen los problemas y cómo podemos adaptar el modelo y los algoritmos vistos en el capítulo anterior para resolverlos.

Para este capítulo, las fuentes principales son [9], [27] y [25, Capítulo 8].

4.1 NOCIONES BÁSICAS DE BIOLOGÍA

Para nuestro estudio necesitamos introducir algunas nociones básicas de biología, en concreto, presentaremos conceptos relacionados con el ADN y las proteínas. Cabe destacar que, por el carácter que tiene este trabajo, no entraremos en detalle sobre estos conceptos y sólo vamos a exponer la información relevante para poder introducir los problemas. Por lo tanto, el lector puede encontrar en determinadas ocasiones, una falta de rigor en el sentido de biología. Por último, este apartado se basa esencialmente en [25, Capítulo 8] y [10, Apéndice A].

El material genético para la mayoría de los seres vivos es el ácido desoxirribonucleico, conocido generalmente como ADN. Consiste en un polímero (conjunto) de nucleótidos, en los que cada nucleótido está compuesto por un glúcido (la desoxirribosa), un grupo fosfato y una base nitrogenada de uno de los siguientes cuatro tipos: adenina, guanina, citosina y timina. En general, se denota a cada nucleótido por la letra inicial de la base que contiene, es decir, por A, G, C y T respectivamente.

Nucleótidos adyacentes en una de las cadenas del ADN se conectan mediante un enlace químico entre el glúcido de uno y el grupo fosfato del siguiente. La estructura clásica de doble hélice del ADN se forma cuando se conectan las dos cadenas de nucleótidos mediante puentes de hidrógeno. Estas conexiones sólo se forman para pares de nucleótidos concretos (conocidos como par de bases): la adenina con la timina ($A \leftrightarrow T$) y la guanina con la citosina ($G \leftrightarrow C$). Por lo tanto, las dos cadenas de ADN son complementarias pues si una cadena contiene una A, entonces estará

conectada con una *T* en la cadena contraria. Análogamente, si una contiene una *C*, la otra contendrá una *G*.

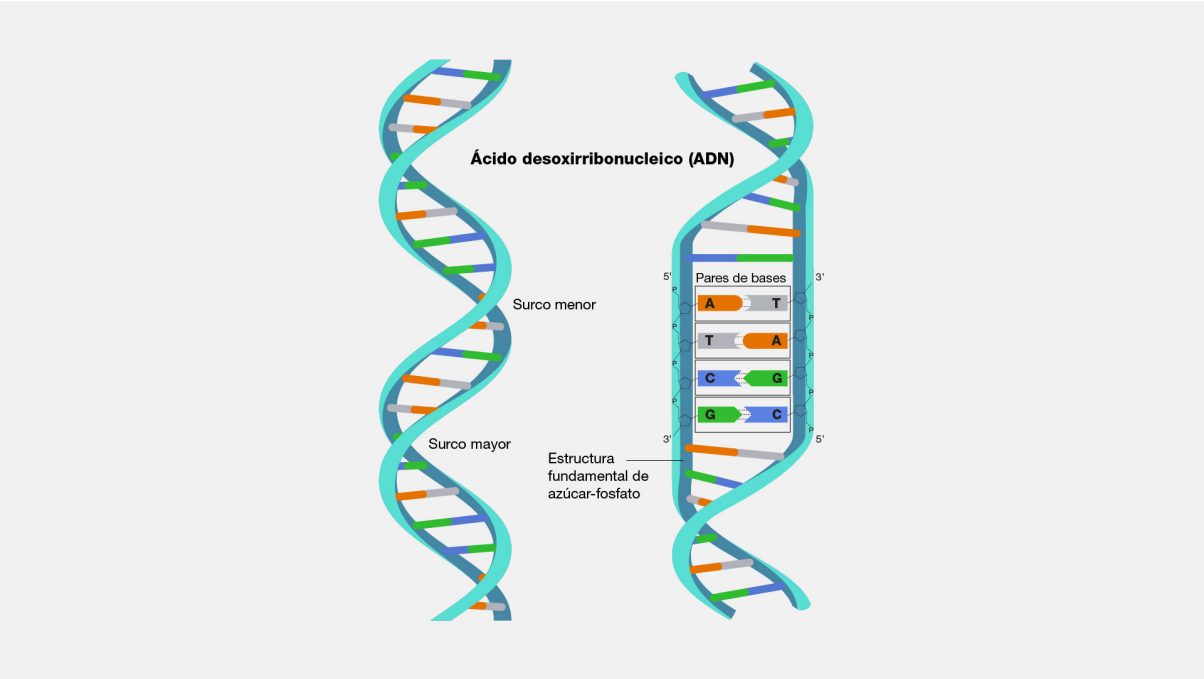


Figura 3: Estructura del ADN [17]

La secuencia que constituye la sucesión de bases en una de las cadenas del ADN representa la información genética codificada en dicha cadena. Por la complementariedad de las cadenas, a partir de una de ellas se puede también determinar la información de la otra. Además, cada cadena tiene definida una dirección espacial contraria a la otra, de modo que sólo podemos interpretar una cadena en la dirección correspondiente.

A nivel celular, el ADN se organiza en cromosomas, cada uno de los cuales contiene ADN superenrollado que puede tener cientos de millones de pares de bases. La mayoría de las células humanas contienen 23 pares de cromosomas, uno heredado del padre y el otro de la madre. Los dos cromosomas de un par son prácticamente idénticos, con la excepción del cromosoma sexual, para el cual existen dos tipos, X e Y. Casi todas las células del cuerpo humano contienen copias idénticas del conjunto completo de 23 pares de cromosomas. El conjunto total de ADN de un organismo se conoce como su genoma, cabe destacar que el genoma humano contiene más de tres mil millones de pares de bases.

Un cromosoma humano está compuesto principalmente por ADN no codificante, cuya función apenas se está empezando a entender. En el ADN se encuentran genes que codifican proteínas. Estos genes representan aproximadamente el 2 % del genoma humano, sin embargo, son el foco de atención de los genetistas. Los genes a menudo están organizados en exones, que son secuencias que eventualmente serán utilizadas

por la célula, alternado con intrones, que serán descartados en el proceso de codificación. La información de estos genes se transcribirá a ARN (ácido ribonucleico) y en muchos casos, se traducirá a proteínas.

En el proceso de transcripción de un gen a ARN, se utiliza la secuencia de ADN del gen (eliminano los intrones) como plantilla. Al igual que el ADN, el ARN está también compuesto por una serie de nucleótidos, pero con ciertas diferencias: el ARN está formado en general por una única cadena y sustituye la timina (T) por el uracilo (U). Un caso particular de ARN, el ARNm (ARN mensajero), será finalmente traducido a proteína.

Una proteína está compuesta por una secuencia de 20 posibles aminoácidos. Cada uno de estos aminoácidos está representado por una o más secuencias de tres nucleótidos de ARN conocidas como codones. La combinación de cuatro posibles nucleótidos en grupos de tres resulta en $4^3 = 64$ codones, lo que significa que la mayoría de los aminoácidos están codificados por más de un codón. La función de una proteína depende finalmente tanto de su secuencia de aminoácidos como de la estructura tridimensional que ha adquirido de su transformación a partir de un ARNm.

4.2 SOFTWARE UTILIZADO

Antes de presentar las aplicaciones de HMM en la biología, presentamos los recursos de software que vamos a utilizar para ilustrar algunos ejemplos. Como recurso principal se van a utilizar dos librerías de Python:

- **NumPy**: es una de las librerías más utilizadas de Python, proporciona la capacidad de tratar elementos matemáticos de forma sencilla y eficiente. En este caso se ha utilizado la versión más reciente hasta el momento, la 1.24.3. Se puede consultar la documentación en [18].
- **hmmlearn**: es una librería de códigos abiertos para Python que implementa modelos de Markov ocultos. Utiliza códigos escritos en C++ para los algoritmos, de forma que son más eficientes que si son implementados directamente en Python. También implementa modelos que no hemos visto en este trabajo. Se ha utilizado para este trabajo la versión 0.3.0. Se puede consultar la documentación en [12] y el código en [13]

A partir de estas herramientas se implementarán archivos de Jupyter Notebook que nos servirán para simular los modelos que presentaremos a continuación. Estos desarrollos se encuentran en `https://github.com/vdeq79/TFG/tree/main/src`.

4.3 ISLAS CPG

En biología computacional, la predicción de genes es un problema en el que se busca identificar regiones codificadoras o genes en un ADN. Puesto que estas regiones poseen ciertas periodicidades y propiedades estadísticas, los HMMs son utilizados para el estudio de este problema. Considerando las estructuras de los genes como estados ocultos y los pares de bases del ADN como observaciones, se puede hacer una predicción de genes aplicando el algoritmo de Viterbi. Este razonamiento es aplicable también a otros problemas de análisis biológico como la búsqueda de regiones funcionales, extracción de patrones, búsqueda de motivos de secuencia e identificación de islas CpG. Este último será el objetivo de nuestro estudio en este apartado [16].

Las islas CpG son regiones de ADN con una gran concentración de dinucletidos CpG, que son citosinas (C) seguido de guaninas (G). Se definen formalmente como regiones de al menos 200 pares de bases con una proporción de C o G superior al 50 % y con un ratio CpG de observado/esperado superior al 60 %. Están íntimamente relacionadas con el inicio de un gen en numerosos genomas de mamíferos, por tanto la presencia de una isla CpG es importante en la predicción de genes. Podemos utilizar HMM para determinar si un fragmento corto de ADN proviene de una isla CpG o para encontrar todas las islas CpG en un segmento largo.

Existen diversos modelos que sirven para este problema, vamos a presentar un modelo casi trivial que consta de 2 estados (con los símbolos + y - presentando la pertenencia a una isla CpG o no):

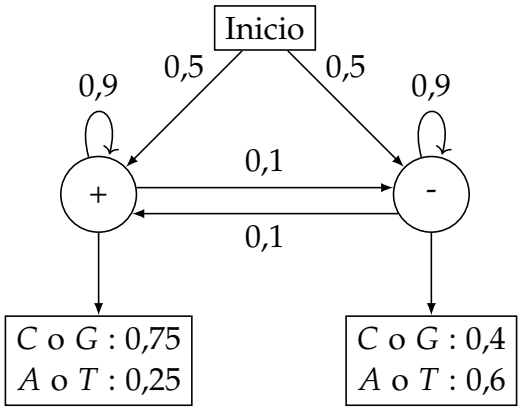


Figura 4: HMM sencillo para identificar islas CpG [16]

Usando este modelo podemos utilizar el algoritmo de Viterbi para identificar islas CpG. Como ejemplo, consideramos la siguiente secuencia:

TCTCGCTGCCGCCAACCCTCGGCGCCGTCGGGTTCGCCGCGGCTCTGATAAG
TCCCGTTTATGGTACCCGGGCCGATCTCTGGTGGGAATCGGAGACCTGTGTAC
CCTGACGCATCCGTTTGTGTTCCCTACACGGCCGACGCAGACCGGGCGCGCG
GCGCCACCCAACGAAGCCCGGGTATGGCACGTGCCCCAGGCGGTGCCCTAC

CCGTATTTCGGGACAAGTTCCCGGATCGGGTGAAAGTTAACGGAAGGATGCC
AAGCAATAGCGGCCACAGGACCCGCCTGGCGACGCATGGACTGGATCCGGA
GGTCTGGCCAACAGTTGATTTCATGGGTTACAGCCCCGGTGTAGATCCCCTCA
TGGTCTCCCGAACCGATTAGTTTGAAAACTGTATCTCCTGGCCGCCTAACAGG
TATAAAGAGCCGGCTCACACTGGGGTGAGGGGGCGCGTGGCCCCCTT

Para determinar si proviene de una isla CpG aplicamos el algoritmo de Viterbi utilizando funcionalidades de **hmmlearn**. Obtenemos la siguiente secuencia de estados:

```
#Construimos nuestro modelo
model=hmm.CategoricalHMM(n_components=2, n_features=2)
model.startprob_=np.array([0.5,0.5])
model.transmat_=np.array([[0.9,0.1],[0.1,0.9]])
model.emissionprob_=np.array([[.75, 0.25],[0.4,0.6]])
#Declaramos la secuencia que queremos tratar
sample="TCTCGCTGCCGCCAACCCTCGGCGCCGTCGGGTTCGCCGCGGCTCTGATAAGTCCCGTTTATGGTACCCGG
GCCGATCTCTGGTGGGAATCGGAGACCTGTGTACCCTGACGCATCCGTTTGTGTTCCCTACACGGCCGACGCAGACCGG
GCGCGCGGCGCCACCCAACGAAGCCCGGGTATGGCACGTGCCCCAGGCGGTGCCCTACCCGTATTTCGGGACAAGTTCC
CGGATCGGGTGAAAGTTAACGGAAGGATGCCAAGCAATAGCGGCCACAGGACCCGCCTGGCGACGCATGGACTGGATCC
GGAGGTCTGGCCAACAGTTGATTTCATGGGTTACAGCCCCGGTGTAGATCCCCTCATGGTCTCCCGAACCGATTAGTTT
GAAAACTGTATCTCCTGGCCGCCTAACAGGTATAAAGAGCCGGCTCACACTGGGGTGAGGGGGCGCGTGGCCCCCTT"
#Consideramos una emisión como la presencia de C o G y otra la presencia de A o
T
sample_sym=np.array([[0 if letter == 'C' or letter == 'G' else 1] for letter in
sample])
log_prob, result=model.decode(sample_sym)
#Tranformamos la salida a los estados
result_sym=np.array(['+' if state==0 else '-' for state in result])
print("".join(result_sym))
>> ++++++++++++++++++++++++++++++++++++++++++----------------------++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
----------------------------------++++++++++++++++++++++++++++++++++++++++++++++
++++----------------------++++++++++++++++++++++++++++++++++++++++----------------
----++++++++++----------++++++++++++++++++++++++++++++++++++++++++++
```

Podemos usar este resultado para distinguir los estados correspondientes en la secuencia de observaciones:

TCTCGCTGCCGCCAACCCTCGGCGCCGTCGGGTTCGCCGCGGCTCTGATAAGT
CCCGTTTATGGTACCCGGGCCGATCTCTGGTGGGAATCGGAGACCTGTGTAC
CCTGACGCATCCGTTTGTGTTCCCTACACGGCCGACGCAGACCGGGCGCGCG
GCGCCACCCAACGAAGCCCGGGTATGGCACGTGCCCCAGGCGGTGCCCTAC

CCGTATTTCGGGACAAGTTCCCGGATCGGGTGAAAGTTAACGGAAGGATGCC
AAGCAATAGCGGCCACAGGACCCGCCTGGCGACGCATGGACTGGATCCGGA
GGTCTGGCCAACAGTTGATTTCATGGGTTACAGCCCCGGTGTAGATCCCCTCA
TGGTCTCCCGAACCGATTAGTTTGAAAACTGTATCTCCTGGCCGCCTAACAGG
TATAAAGAGCCGGCTCACACTGGGGTGAGGGGGCGCGTGGCCCCCTT

Donde el color azul representa que las letras provienen de un estado + y el color rojo representa que provienen de un estado - como resultado de aplicar el algoritmo de Viterbi. Si analizamos las secuencias en rojo, podemos ver que contienen una proporción de T o A superior al 50 %. Pero en definitiva, tenemos una mayor proporción de estados + en la secuencia de estados resultante, lo que sugiere que la secuencia de observaciones puede provenir de una isla CpG.

4.4 ALINEAMIENTO DE PARES DE SECUENCIAS

En análisis de secuencias, muchas veces es importante comparar dos secuencias para determinar si están funcionalmente relacionadas. Por ejemplo, genes con funciones similares en diferentes organismos suelen tener secuencias de ADN muy parecidas. Si un gen nuevo es suficientemente similar a otro gen de otro organismo cuya función es conocida, entonces es razonable esperar que el nuevo gen ejerza la misma función. Lo mismo ocurre con las proteínas, si se descubre una nueva proteína de la que se desconoce su estructura tridimensional pero con una similitud sustancial entre su secuencia de aminoácidos y la secuencia de una proteína de la que sí se conoce la estructura tridimensional, entonces es razonable esperar que la estructura de la nueva proteína sea de alguna forma similar a la de la proteína conocida.

Para estudiar este problema, comenzamos estableciendo algunas notaciones. Vamos a considerar un par de secuencias, x e y de longitudes n y m respectivamente. Sea x_i el i-ésimo símbolo de x e y_j el j-ésimo símbolo de y, estos símbolos pertenecen a un cierto alfabeto $\mathcal{A}$ que, en el caso del ADN, serán las 4 bases $\{A, C, G, T\}$ y, en el caso de las proteínas, serán los 20 aminoácidos.

Si sólo consideramos las secuencias originales el problema sería trivial: sólo existe un único alineamiento en el caso de que $n = m$ y en otro caso, el problema se reduciría en encontrar la mejor posición para incluir una secuencia en otra. El problema real tiene en cuenta posibles inserciones de huecos (*gaps*) en cualquiera de las dos secuencias para obtener alineamientos entre símbolos iguales. No permitiendo insertar hueco en ambas secuencias al mismo tiempo ni insertar dos huecos en diferentes secuencias de forma seguida.

Estas consideraciones surgen debido a que, al comparar dos secuencias, buscamos evidencias que provengan de un ancestro común a través de un proceso de mutación y selección. Las mutaciones que se tienen en cuenta incluyen sustituciones, que modi-

fican elementos de una secuencia, así como inserciones y eliminaciones, que agregan o eliminan elementos. A continuación, consideremos el siguiente ejemplo:

Ejemplo 4.1. Tomamos dos secuencias de ADN:

$$x = CACGAAT, y = AGTTCAA$$

Podemos considerar un alineamiento entre estas dos secuencias como sigue:

$$\begin{array}{ccccccccc} C & A & - & - & - & C & G & A & A & T \\ - & A & G & T & T & C & - & A & A & - \end{array}$$

donde cada $-$ representa un hueco.

Como podemos apreciar en el ejemplo, las inserciones producen nuevos alineamientos a tener en cuenta. Para valorar estos alineamientos, se define una matriz de puntuaciones que asigna para cada alineamiento de símbolos un valor. Esta matriz se conoce usualmente como la matriz de sustitución. Un ejemplo de matriz de sustitución para secuencias de ADN podría ser la siguiente:

$$S = \begin{array}{c} \\ A \\ C \\ G \\ T \end{array} \begin{array}{c} \begin{array}{cccc} A & C & G & T \end{array} \\ \begin{pmatrix} 10 & -3 & -2 & 1 \\ -2 & 8 & 1 & -2 \\ -3 & 1 & 9 & -3 \\ 0 & -3 & -2 & 6 \end{pmatrix} \end{array} \tag{4.1}$$

En esta matriz, cada elemento representa la puntuación que obtendría un alineamiento entre el símbolo de la fila y el símbolo de la columna. Cada uno de estos valores, se puede calcular de la siguiente manera:

$$s(a, b) = \log\left(\frac{p_{ab}}{q_a \cdot q_b}\right)$$

siendo a y b elementos del alfabeto $\mathcal{A}$, p_{ab} la probabilidad de que se produzca un emparejamiento entre a y b y q_a la frecuencia relativa esperada de que se produzca un símbolo a en una secuencia.

Estos valores son relevantes pues afectan a la significación final del análisis: si tenemos un sistema de puntuación ajustado a la realidad, podremos afirmar con mayor seguridad las similitudes entre dos secuencias. Por esta razón, existen métodos para derivar estos valores a partir de datos conocidos y matrices de sustitución utilizadas ampliamente como las matrices PAM o BLOSUM.

Además de los alineamientos entre los símbolos del alfabeto, se tienen en cuenta también los alineamientos entre un símbolo y un hueco. Estos alineamientos tienen una puntuación negativa, lo que se conoce generalmente como penalizaciones por

hueco (*gap penalties*). El coste asociado a una consecución de huecos de longitud g suele venir dado por una de las dos siguientes funciones:

$$\gamma_1(g) = -gd,$$

$$\gamma_2(g) = -d - (g-1)e,$$

donde d se considera la penalización por iniciar la secuencia de huecos y e se considera la penalización por extender la secuencia, usualmente con un valor menor que d. Mientras que en γ_1 se trata todos los huecos por igual, en γ_2 se penaliza menos las secuencias de huecos con mayores longitudes. Esto es deseable cuando se prevé que las inserciones de varios huecos sean tan frecuentes como inserciones de un único hueco. En la práctica, los valores d y e se escogen empíricamente una vez elegidos los valores de la matriz de sustitución.

Con estos elementos, podemos determinar el mejor alineamiento con huecos entre dos secuencias entendiéndolo como aquel con mayor puntuación dada la matriz de sustitución. Si consideramos las dos secuencias enteras, estaremos hablando del alineamiento global de pares de secuencias. Si lo que buscamos es el mejor alineamiento entre subsecuencias de x e y, entonces estaremos hablando del alineamiento local.

Para nuestro estudio, vamos a centrarnos en el problema de alineamiento global. Este problema se puede resolver empleando el algoritmo de Needleman-Wunsch, un algoritmo de programación dinámica que garantiza encontrar el alineamiento óptimo entre dos secuencias con posibles huecos. Pero también podemos utilizar un tipo específico de HMM para resolver este problema, los *Pair HMMs*.

4.4.1 *Pair HMM*

A diferencia de los HMMs que habíamos presentado hasta ahora, los *pair HMMs* generan dos salidas en cada estado en lugar de uno. Por esta razón, podemos utilizar este modelo para el problema de alineamiento de pares. Existen distintas variaciones de este modelo, vamos a presentar el modelo ilustrado en [9].

El espacio de estados consiste principalmente en tres estados: en el estado M se generan dos símbolos del alfabeto $\mathcal{A}$ mientras que en los estados X e Y se emite un símbolo de la salida correspondiente dejando un hueco en la otra. Llamaremos p_{ab} a la probabilidad de que se emitan los símbolos a y b desde el estado M y q_a la probabilidad de emitir un símbolo a y un hueco desde los estados X e Y.

A los estados X e Y se les conoce como estados de inserción (o estado de inserción y eliminación en algunas fuentes, por ejemplo [2]) y al estado M, el estado de alineamiento (*match state*). Para entender mejor los estados volvemos a tomar el alineamiento del ejemplo 4.1, la secuencia de estados que produce dicho alineamiento es la siguiente:

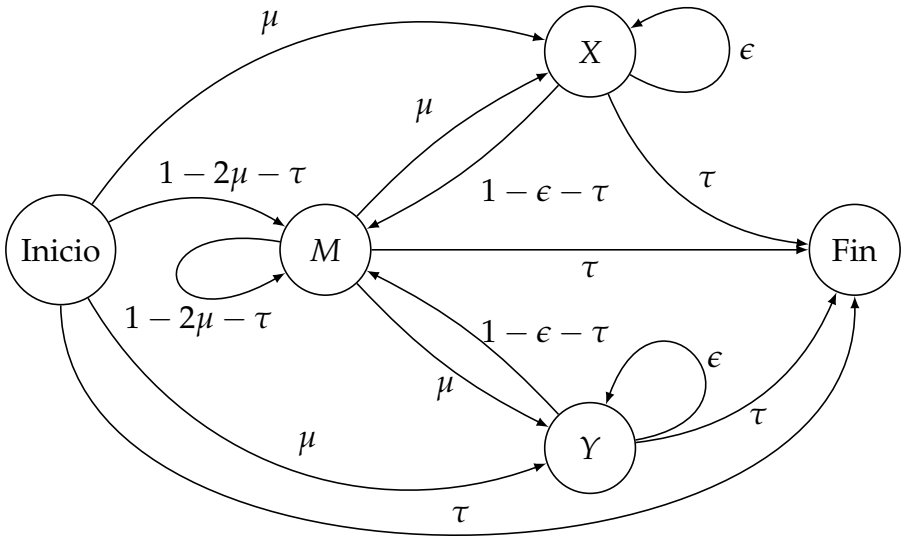


Figura 5: Estructura de *pair HMM* [9]

Ejemplo 4.2.

$x \longrightarrow$	C	A	$-$	$-$	$-$	C	G	A	A	T
$y \longrightarrow$	$-$	A	G	T	T	C	$-$	A	A	$-$
Estado	X	M	Y	Y	Y	M	X	M	M	X

Asumiendo que las transiciones a los estados de inserción se tratan de forma análoga, existen dos parámetros para definir las probabilidades de transición:

$\mu =$ probabilidad de pasar del estado M a un estado de inserción (X o Y).
$\epsilon =$ probabilidad de permanecer en un estado de inserción.

Puesto que estamos tratando con secuencias espaciales en lugar de temporales, tiene sentido definir un estado de inicio y un estado de fin. Ambos estados son estados silenciosos, estados que no producen ninguna salida. El hecho de añadir un estado de inicio facilita posteriormente la modificación de los algoritmos. Este estado sustituye la función de la distribución inicial, de modo que el modelo siempre empezará por el estado de inicio. Para este problema, podemos considerar que el estado de inicio tiene las mismas probabilidades de transición que M.

Por otro lado, al añadir un estado de fin necesitamos introducir un nuevo parámetro, la probabilidad de pasar a dicho estado desde los otros estados. Asumimos que dicha probabilidad es la misma desde cualquiera de los otros estados y lo denotamos por τ. Este valor influirá en la longitud media de los alineamientos generados por el modelo. Con estos elementos, tenemos el modelo que se muestra en la figura 5.

Una vez conocido el modelo, podemos aplicar el algoritmo de Viterbi adaptado para encontrar el alineamiento óptimo. En este caso, la entrada consiste en dos secuencias $x = (x_1, \ldots, x_n)$ e $y = (y_1, \ldots, y_m)$ y la salida es la secuencia de estados que conduce al alineamiento óptimo.

Utilizaremos los índices i y j para referirnos a los elementos de las secuencias x e y. Puesto que x_i e y_j son elementos del alfabeto $\mathcal{A}$, usaremos la notación $p_{x_i y_j}$, q_{x_i}, q_{y_j} para distinguir los alineamientos entre los símbolos y los posibles huecos de x e y.

Recordemos que en el algoritmo original:

$$\delta_t(i) = \max_{(q_0, q_1 \cdots, q_{t-1}) \in \mathbb{S}^t} P[\mathcal{X}_0^t = (q_0, q_1, \ldots, q_{t-1}, s_i), \mathcal{Y}_0^t = (O_0, \ldots, O_t)].$$

Y se calcula de la siguiente forma:

$$\begin{aligned} &\delta_0(i) = b_{s_i}(O_0) \cdot \pi_i, \\ &\delta_t(i) = \max_{j=1,\ldots,N} \left(a_{ji} \cdot \delta_{t-1}(j)\right) \cdot b_{s_i}(O_t), \qquad 1 \leq i \leq N, \quad 1 \leq t \leq r. \end{aligned} \tag{4.2}$$

En este caso, sólo tenemos que calcular las variables relacionadas con los estados M, X e Y. El estado de inicio se representará también por el estado M por simplicidad. Puesto que el estado fin es un estado silencioso y representa el final del alineamiento, tendrá un tratamiento especial.

Usando los índices i y j, la variable del algoritmo de Viterbi pasa a ser de forma $\delta_{i,j}(s)$ con $s \in \{M, X, Y\}$. Esta variable tiene el mismo significado que en el caso general: es la probabilidad de la secuencia de estados más probable terminado en el estado s, habiendo considerado hasta las posiciones i y j de las secuencias de entrada. Por lo tanto, el cálculo es idéntico al caso general, sustituyendo el índice temporal por los índices i y j.

Concretamente, inicializamos el algoritmo asignando los siguientes valores:

$$\delta_{0,0}(M) = 1.$$

$$\begin{aligned} \delta_{0,j}(X) = \delta_{i,0}(Y) = 0, &\qquad 0 \leq i \leq n,\ 0 \leq j \leq m. \\ \delta_{0,j}(M) = \delta_{i,0}(M) = 0, &\qquad 1 \leq i \leq n,\ 1 \leq j \leq m. \end{aligned}$$

El primer caso se debe a que siempre empezamos en el estado de inicio (que hemos representado en este caso por el estado M) y al ser silencioso podemos asumir que tiene la probabilidad de emisión 1. Los otros casos se deben a que son situaciones imposibles. Por ejemplo, $\delta_{0,1}(X)$ es la probabilidad de que se haya considerado y_1 y ningún símbolo de x habiendo tomado como último estado el estado X. Esto contradice con la propia definición de X, pues en dicho estado siempre se emite un símbolo de x con un hueco en y.

El resto de las variables se calculan de forma similar que en (4.2), iremos por casos:

- $\delta_{i,j}(M)$ indica que se ha alineado x_i e y_j tomando como último estado el estado M. Además, aplicando recursividad, tenemos que encontrar la probabilidad de la secuencia más probable habiendo considerado hasta las posiciones $i-1$ y $j-1$ terminado en un estado s y considerar la transición de s a M:

$$\delta_{i,j}(M) = p_{x_i y_j} \cdot \text{máx} \begin{cases} (1-2\mu-\tau)\cdot\delta_{i-1,j-1}(M) \\ (1-\epsilon-\tau)\cdot\delta_{i-1,j-1}(X) \\ (1-\epsilon-\tau)\cdot\delta_{i-1,j-1}(Y) \end{cases} \qquad 1 \leq i \leq n,\, 1 \leq j \leq m.$$

- $\delta_{i,j}(X)$ indica que se ha emitido x_i y un hueco tomando como último estado el estado X. Puesto que en este caso no se emite ningún símbolo de y, quiere decir que y_j ya había sido emitido anteriormente. Como no es posible pasar del estado Y al X, existen dos posibilidades: el alineamiento entre x_{i-1} e y_j o el alineamiento entre x_k e y_j con $k < i-1$. Este último caso implicaría la emisión de x_{i-1} con un hueco, por lo tanto:

$$\delta_{i,j}(X) = q_{x_i} \cdot \text{máx} \begin{cases} \mu \cdot \delta_{i-1,j}(M) \\ \epsilon \cdot \delta_{i-1,j}(X) \end{cases} \qquad 1 \leq i \leq n,\, 0 \leq j \leq m.$$

- $\delta_{i,j}(Y)$ se calcula de mismo modo que $\delta_{i,j}(X)$:

$$\delta_{i,j}(Y) = q_{y_j} \cdot \text{máx} \begin{cases} \mu \cdot \delta_{i,j-1}(M) \\ \epsilon \cdot \delta_{i,j-1}(Y) \end{cases} \qquad 0 \leq i \leq n,\, 1 \leq j \leq m.$$

Finalmente, se calcula:

$$\delta(Fin) = \tau \cdot \text{máx}\{\delta_{n,m}(M),\, \delta_{n,m}(X),\, \delta_{n,m}(Y)\}.$$

Para obtener la secuencia óptima mantenemos el estado correspondiente al argumento máximo de cada variable y lo recuperamos desde $\delta(Fin)$ del mismo modo que en el algoritmo original. Para evitar el problema de convergencia a 0, basta aplicar logaritmo a las variables tal como hacíamos en la versión original.

Una de las ventajas que nos proporciona la utilización de *pair HMM* respecto a algoritmos que emplean directamente programación dinámica, es que ahora podemos calcular la probabilidad de que dos secuencias estén relacionadas según el modelo, independientemente del alineamiento. Lo hacemos considerando todos los posibles alineamientos:

$$P[x,y] = \sum_{Alineamiento\, Q} P[x,y,Q].$$

Para calcular esta suma adaptamos el algoritmo de avance a este modelo. La variable de avance se puede calcular de forma recursiva similar a la variable de Viterbi, pero sumando las variables previas en lugar de calcular el máximo. Una vez obtenida esta

probabilidad, podemos calcular la probabilidad del alineamiento óptimo Q^*. Por la definición de la variable de Viterbi, esto es:

$$P[Q^*|x,y] = \frac{P[Q^*,x,y]}{P[x,y]} = \frac{\delta(Fin)}{P[x,y]}.$$

Para evitar el problema de convergencia a 0 en el algoritmo de avance, podemos tomar logaritmo de las variables. Recordemos que:

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}\right) \cdot b_{s_j}(O_{t+1}), \qquad 0 \leq t \leq r-1, \quad 1 \leq j \leq N.$$

Aplicando logaritmo:

$$\begin{aligned}\log(\alpha_{t+1}(j)) &= \log\left(\sum_{i=1}^{N} \alpha_t(i) \cdot a_{ij}\right) + \log(b_{s_j}(O_{t+1})) \\ &= \log\left(\sum_{i=1}^{N} \exp\left[\log(\alpha_t(i)) + \log(a_{ij})\right]\right) + \log(b_{s_j}(O_{t+1}))\end{aligned}$$

Utilizando la función **logsumexp** de la librería **SciPy**[23], podemos calcular fácilmente este logaritmo.

Ejemplo 4.3. Utilizando las librerías **hmmlearn** y **NumPy** se ha implementado la clase **PairHMM** en python. Esta clase tiene la misma estructura que hemos explicado anteriormente y también se ha implementado el algoritmo de Viterbi y el algoritmo de avance correspondiente. A continuación, vamos a alinear las secuencias del ejemplo 4.1 utilizando esta clase con los siguientes parámetros:

$$\mu = 0{,}2 \qquad \epsilon = 0{,}3 \qquad \tau = 0{,}1$$

$$p_{AA} = p_{CC} = p_{GG} = p_{TT} = 0{,}2$$
$$p_{AT} = p_{TA} = p_{CG} = p_{GC} = 0{,}025$$
$$p_{AC} = p_{AG} = p_{CA} = p_{CT} = p_{GA} = p_{GT} = p_{TC} = p_{TG} = 0{,}0125$$
$$q_A = q_C = q_G = q_T = 0{,}25$$

Tomando las secuencias $x = CACGAAT$ e $y = AGTTCAA$, aplicamos el algoritmo de Viterbi respecto del modelo que construimos con los anteriores parámetros:

```
1 model = PairHMM(alphabet=["A", "C", "G", "T"], mu=0.2, epsilon=0.5, tau=0.1,
  match_probabilities=[0.2, 0.0125, 0.0125, 0.025, 0.0125, 0.2, 0.025, 0.0125,
  0.0125, 0.025, 0.2, 0.0125, 0.025, 0.0125, 0.0125, 0.2],
  insert_probabilities=[0.25]*4)
2 x="CACGAAT"
3 y="AGTTCAA"
```

```
sequence, alignment, p_fin=model.modifiedViterbi( x, y )
print("\n".join(alignment))
>> CA---CGAAT
>> -AGTTC-AA-
```

Como se puede apreciar, nos devuelve el siguiente alineamiento:

$$\begin{array}{cccccccccc} C & A & - & - & - & C & G & A & A & T \\ - & A & G & T & T & C & - & A & A & - \end{array}$$

Este es el mismo alineamiento que vimos en el ejemplo 4.1, pero no se trata de una casualidad pues dicho alineamiento es el mejor alineamiento con la matriz de sustitución dada en (4.1) y usando la función de penalización de huecos $\gamma(g) = -g$ [25].

Observando esa matriz podemos ver que aparte de los alineamientos entre los mismos símbolos, los alineamientos entre A con T y C con G se valoran también positivamente, lo que se traduce en una mayor probabilidad que el resto.

Calculamos la probabilidad de que x e y estén relacionados independientemente del alineamiento:

```
total_prob=model.modifiedFoward(x, y)
print(total_prob)
>> 5.85602525347472e-12
```

Esta probabilidad nos puede parecer baja, pero si calculamos esta probabilidad con dos secuencias idénticas bajo el mismo modelo:

```
print(model.modifiedFoward("AAAAAAA", "AAAAAAA"))
>> 2.8039432167959317e-08
```

Podemos apreciar que esta probabilidad es en general baja y disminuye significativamente conforme aumenta la longitud de las secuencias. Calculamos ahora la probabilidad del alineamiento óptimo:

```
p_sequence = p_fin/total_prob
print(p_sequence)
>> 0.10373930673190596
```

En este caso, obtenemos un valor bastante significativo, lo cuál nos indica la "correctitud" del alineamiento. En general, la probabilidad del alineamiento óptimo suele ser un valor extremadamente pequeño debido a que existen numerosos alineamientos muy similares al óptimo que tienen una probabilidad ligeramente menor.

Finalmente, cabe enfatizar que el alineamiento que nos proporciona el algoritmo de Viterbi depende de los parámetros, un cambio como el de modificar el valor de ϵ a $\epsilon = 0{,}5$ produce un alineamiento diferente:

```
model = PairHMM(alphabet=["A", "C", "G", "T"], mu=0.2, epsilon=0.5, tau=0.1,
match_probabilities=[0.2, 0.0125, 0.0125, 0.025, 0.0125, 0.2, 0.025, 0.0125,
0.0125, 0.025, 0.2, 0.0125, 0.025, 0.0125, 0.0125, 0.2],
insert_probabilities=[0.25]*4)
x="CACGAAT"
y="AGTTCAA"
sequence, alignment, p_fin=model.modifiedViterbi( x, y )
print("\n".join(alignment))

>> CACG---AAT
>> -A-GTTCAA-
```

4.5 PROFILE HMM

En el apartado anterior hemos visto cómo podemos alinear dos secuencias de ADN o de proteínas para determinar si están relacionadas. En general, secuencias biológicas con funciones similares forman familias. Para caracterizar las propiedades de una familia, se acude al alineamiento múltiple de secuencias de la familia. Una vez obtenido dicho alineamiento, podemos determinar si una nueva secuencia x pertenece a la familia correspondiente, y en caso afirmativo, concluir que tiene una determinada funcionalidad.

Para modelar las similitudes resultantes del alineamiento, podemos usar un tipo particular de HMM, llamado *profile HMM*. En este apartado, vamos a asumir que tenemos dado el alineamiento múltiple de secuencias de una familia y vamos a presentar la forma de construir el *profile HMM* correspondiente.

A diferencia de los HMMs generales, los *profile HMMs* tienen una estructura estrictamente lineal y no cíclica. El primer paso para construir el modelo es determinar su longitud. Para ello, necesitamos separar el alineamiento en dos posibles regiones: la región de alineamiento, que está formada por posiciones cuyos elementos son mayoritariamente símbolos y la región de inserción formada por posiciones con una alta frecuencia de huecos. Aunque existen diferentes criterios, lo usual es que posiciones cuyos huecos son más de la mitad de los elementos formen parte de la región de inserción y en caso contrario, formen parte de la región de alineamiento. Veamos el siguiente ejemplo:

Ejemplo 4.4.

$$\begin{array}{cccccc} T & A & - & - & T & C \\ T & A & G & - & T & C \\ T & A & G & A & - & C \\ - & A & G & - & T & G \\ M & M & M & I & M & M \end{array}$$

Las posiciones con M en última fila indica que pertenecen a la región de alineamiento mientras que la posición con I indica que pertenece a la región de inserción.

Una vez identificadas las regiones, la longitud del modelo viene dada por el número de posiciones que forman parte de la región de alineamiento. Conocida la longitud, el siguiente paso es construir el espacio de estados, que está principalmente constituido por 3 tipos de estados: los estados de alineamiento M_k, los estados de inserción I_k y los estados de eliminación D_k. Cada uno de de los índices k, corresponde con la k-ésima posición dentro de la región de alineamiento comenzando desde 1. Para el ejemplo 4.4 tenemos:

$$\begin{array}{cccccc} & T & A & - & - & T & C \\ & T & A & G & - & T & C \\ & T & A & G & A & - & C \\ & - & A & G & - & T & G \\ \hline k = & 1 & 2 & 3 & & 4 & 5 \end{array}$$

A partir de la longitud n, construimos un HMM trivial formado por los estados de alineamiento M_k con $k = 1, \ldots, n$ y probabilidades de transición 1 al siguiente estado M_{k+1}. Los estados M_k representan el caso de que un símbolo de x se alinee con el símbolo más frecuente de la posición correspondiente. Las probabilidades de emisión de M_k corresponden con las frecuencias relativas de cada símbolo en la posición k-ésima del alineamiento. También consideramos los estados de inicio y fin, con los mismos significados que en el caso de *pair HMM* pero con probabilidades de transición triviales. Para el ejemplo 4.4 tenemos el siguiente HMM:

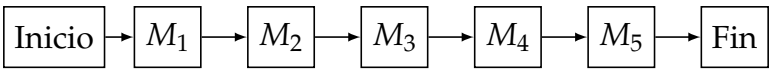


Figura 6: HMM trivial a partir del ejemplo 4.4

A partir del modelo trivial, añadimos los estados I_k y D_k. Los estados I_k modelan las inserciones, emisiones de símbolos adicionales a los n símbolos emitidos por el modelo trivial. En otras palabras, cada inserción representa el alineamiento de un símbolo en una posición fuera de la región de alineamiento y formada mayoritariamente o totalmente por huecos. En este caso, k indica la última posición de la región de alineamiento anterior a la inserción. Para modelar posibles inserciones anteriores a cualquier alineamiento de símbolos, añadimos un estado I_0. Cada estado de inserción

está asociado a 3 transiciones posibles: $M_k \to I_k$ que representa la inserción tras un alineamiento, $I_k \to I_k$ para permitir inserciones múltiples y la transición $I_k \to M_{k+1}$ para un nuevo alineamiento.

Por otra parte, los estados D_k modelan las eliminaciones, emisiones de huecos en la posición k de la región de alineamiento. Cada estado tiene 4 transiciones posibles: $M_{k-1} \to D_k$ representando la eliminación en la posición k de la región de alineamiento, $D_k \to D_{k+1}$ para permitir eliminaciones múltiples, $D_k \to M_{k+1}$ para alinear después de una eliminación.

Además, permitimos las transiciones $D_k \to I_k$ para inserciones tras eliminar e $I_k \to D_{k+1}$ para eliminaciones tras insertar. Estas transiciones son muy poco probables, pero las añadimos pues dejándolas fuera puede generar problemas a la hora de construir el modelo.

Tomando el alineamiento del ejemplo 4.4, añadiendo los estados y transiciones al modelo trivial, obtenemos el siguiente *profile HMM*:

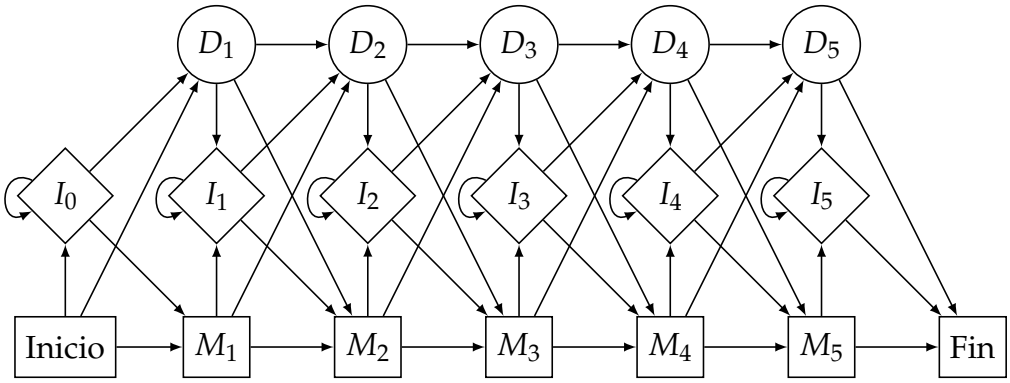


Figura 7: *Profile HMM* a partir del ejemplo 4.4

Determinada la estructura del modelo, ahora necesitamos hallar las probabilidades de transición y de emisión. Con los estados que hemos definido, conocemos para cada una de las secuencias del alineamiento múltiple, la secuencia de estados que la produce. Por lo tanto, podemos estimar las probabilidades contando las transiciones y las emisiones de símbolos de todas las secuencias de estados y calcular las frecuencias relativas correspondientes. Para permitir probabilidades de transición o de emisión que no han sido observadas en el alineamiento original, añadimos un valor ponderado denominado pseudoconteo a los recuentos para evitar probabilidades iguales a cero. Este valor depende de la probabilidad que estamos calculando y del conocimiento que tenemos sobre el problema; el más simple posible es añadir 1 a todas las observaciones. Veamos a continuación un ejemplo utilizando esta aproximación:

Ejemplo 4.5. Volvemos a tomar el alineamiento múltiple del ejemplo 4.4 con el modelo correspondiente a la figura 7. En primer lugar vamos a determinar los estados que

generan cada secuencia. Para evitar confusiones, pondremos M_0 para representar al inicio y M_6 para representar al fin.

$$
\begin{array}{cccccccc}
 & T & A & - & - & T & C & \Longrightarrow \quad M_0\, M_1\, M_2\, D_3\, M_4\, M_5\, M_6 \\
 & T & A & G & - & T & C & \Longrightarrow \quad M_0\, M_1\, M_2\, M_3\, M_4\, M_5\, M_6 \\
 & T & A & G & A & - & C & \Longrightarrow \quad M_0\, M_1\, M_2\, M_3\, I_3\, D_4\, M_5\, M_6 \\
 & - & A & G & - & T & G & \Longrightarrow \quad M_0\, D_1\, M_2\, M_3\, M_4\, M_5\, M_6 \\
\hline
k = & 1 & 2 & 3 & & 4 & 5 &
\end{array}
$$

Contamos ahora las frecuencias de cada símbolo en los estados M_k e I_k, no es necesario calcular para los D_k pues siempre emiten un hueco.

	A	C	G	T
M_1	0	0	0	3
M_2	4	0	0	0
M_3	0	0	3	0
M_4	0	0	0	3
M_5	0	3	1	0

Tabla 2: Emisiones desde M_k

	A	C	G	T
I_0	0	0	0	0
I_1	0	0	0	0
I_2	0	0	0	0
I_3	1	0	0	0
I_4	0	0	0	0
I_5	0	0	0	0

Tabla 3: Emisiones desde I_k

Para las transiciones entre estados, tenemos las siguientes frecuencias donde cada k representa el índice del estado que inicia la transición:

k	0	1	2	3	4	5
$M \to M$	3	3	3	2	3	4
$M \to D$	1	0	1	0	0	–
$M \to I$	0	0	0	1	0	0
$I \to M$	0	0	0	0	0	0
$I \to D$	0	0	0	1	0	–
$I \to I$	0	0	0	0	0	0
$D \to M$	–	1	0	1	1	0
$D \to D$	–	0	0	0	0	–
$D \to I$	–	0	0	0	0	0

Tabla 4: Frecuencias de transiciones

Como podemos ver, hay muchas transiciones y emisiones con frecuencia cero, para evitar probabilidades iguales a cero, sumamos 1 a todas las frecuencias y calculamos las probabilidades:

	A	C	G	T
M_1	$1/7$	$1/7$	$1/7$	$4/7$
M_2	$5/8$	$1/8$	$1/8$	$1/8$
M_3	$1/7$	$1/7$	$4/7$	$1/7$
M_4	$1/7$	$1/7$	$1/7$	$4/7$
M_5	$1/8$	$1/2$	$1/4$	$1/8$

Tabla 5: Probabilidades de emisión de M_k

	A	C	G	T
I_0	$1/4$	$1/4$	$1/4$	$1/4$
I_1	$1/4$	$1/4$	$1/4$	$1/4$
I_2	$1/4$	$1/4$	$1/4$	$1/4$
I_3	$2/5$	$1/5$	$1/5$	$1/5$
I_4	$1/4$	$1/4$	$1/4$	$1/4$
I_5	$1/4$	$1/4$	$1/4$	$1/4$

Tabla 6: Probabilidades de emisión de I_k

k	0	1	2	3	4	5
$M \to M$	$4/7$	$2/3$	$4/7$	$1/2$	$2/3$	$5/6$
$M \to D$	$2/7$	$1/6$	$2/7$	$1/6$	$1/6$	$-$
$M \to I$	$1/7$	$1/6$	$1/7$	$1/3$	$1/6$	$1/6$
$I \to M$	$1/3$	$1/3$	$1/3$	$1/4$	$1/3$	$1/2$
$I \to D$	$1/3$	$1/3$	$1/3$	$1/2$	$1/3$	$-$
$I \to I$	$1/3$	$1/3$	$1/3$	$1/4$	$1/3$	$1/2$
$D \to M$	$-$	$1/2$	$1/3$	$1/2$	$1/2$	$1/2$
$D \to D$	$-$	$1/4$	$1/3$	$1/4$	$1/4$	$-$
$D \to I$	$-$	$1/4$	$1/3$	$1/4$	$1/4$	$1/2$

Tabla 7: Probabilidades de transición

Una vez que conocemos la estructura y la forma de determinar las probabilidades, podemos utilizar el modelo resultante para saber si una secuencia pertenece a una familia. En este caso, el algoritmo de Viterbi y el algoritmo de avance se implementan de forma similar que en el caso de *pair HMM*. Usando la misma notación definiremos de forma recursiva las variables de Viterbi $\delta_i(M_k)$, $\delta_i(D_k)$, $\delta_i(I_k)$ con $i \in \{0, \ldots, m\}$ siendo m la longitud de la secuencia de entrada x. Para inicializar, usamos la notación M_0 para el estado de inicio. Puesto que siempre se empieza por dicho estado:

$$\delta_0(M_0) = 1$$

Ahora, analizando la figura 7 tenemos las siguientes relaciones de recursividad:

- Puesto que las transiciones a M_j llegan desde estados con índices $j-1$ y M_j emite un símbolo:

$$\delta_i(M_j) = b_{M_j}(x_i) \cdot \text{máx} \begin{cases} a_{M_{j-1}M_j} \cdot \delta_{i-1}(M_{j-1}) \\ a_{D_{j-1}M_j} \cdot \delta_{i-1}(D_{j-1}) \\ a_{I_{j-1}M_j} \cdot \delta_{i-1}(I_{j-1}) \end{cases} \qquad 1 \leq i \leq m,\ 1 \leq j \leq n.$$

- Para permitir emisiones de huecos en la región de alineamiento anterior a cualquier símbolo de x, calculamos también $\delta_0(D_j)$. Las transiciones a D_j llegan desde estados con índices $j-1$ y D_j únicamente emiten huecos:

$$\delta_i(D_j) = \text{máx} \begin{cases} a_{M_{j-1}D_j} \cdot \delta_i(M_{j-1}) \\ a_{D_{j-1}D_j} \cdot \delta_i(D_{j-1}) \\ a_{I_{j-1}D_j} \cdot \delta_i(I_{j-1}) \end{cases} \qquad 0 \leq i \leq m,\ 1 \leq j \leq n.$$

- Puesto que las transiciones a I_j llegan desde estados con índices j y I_j emite un símbolo:

$$\delta_i(I_j) = b_{I_j}(x_i) \cdot \text{máx} \begin{cases} a_{M_jI_j} \cdot \delta_{i-1}(M_j) \\ a_{D_jI_j} \cdot \delta_{i-1}(D_j) \\ a_{I_jI_j} \cdot \delta_{i-1}(I_j) \end{cases} \qquad 1 \leq i \leq m,\ 0 \leq j \leq n.$$

Para la implementación del algoritmo de avance, basta con sustituir la suma por el máximo en las variables de Viterbi.

Ejemplo 4.6. De la misma forma que para *pair HMM* se ha implementado la clase **ProfileHMM** en python. Para construir esta clase necesitamos proporcionar el alfabeto correspondiente, el símbolo para el hueco y el alineamiento dado. Utilizando el alineamiento del ejemplo anterior aplicado el pseudoconteo:

```
model = ProfileHMM(alphabet=["A", "C", "G", "T"], gap_symbol='-',
alignment=['TA--TC', 'TAG-TC', 'TAGA-C', '-AG-TG'], show_probabilities=True)
>> Probabilidades de emisión de estados de alineamiento:
#Se representa de la misma forma que en el ejemplo anterior.
>> [[0.14285714 0.14285714 0.14285714 0.57142857]
>>  [0.625      0.125      0.125      0.125     ]
>>  [0.14285714 0.14285714 0.57142857 0.14285714]
>>  [0.14285714 0.14285714 0.14285714 0.57142857]
>>  [0.125      0.5        0.25       0.125     ]]
>> Probabilidades de emisión de estados de inserción:
>> [[0.25 0.25 0.25 0.25]
>>  [0.25 0.25 0.25 0.25]
>>  [0.25 0.25 0.25 0.25]
>>  [0.4  0.2  0.2  0.2 ]
>>  [0.25 0.25 0.25 0.25]
>>  [0.25 0.25 0.25 0.25]]
>> Probabilidades de transición entre estados:
#Se representa de la misma forma que en el ejemplo anterior, se ordena los
estados de inicio de transición de forma: M, D, I. No se representan las
transiciones al estado fin.
```

```
>> [[[0.57142857 0.66666667 0.57142857 0.5        0.66666667]
>>  [0.28571429 0.16666667 0.28571429 0.16666667 0.16666667]
>>  [0.14285714 0.16666667 0.14285714 0.33333333 0.16666667]]
#Transiciones desde los estados de eliminación, ignoraremos la primera columna.
>> [[0.33333333 0.5        0.33333333 0.5        0.5       ]
>>  [0.33333333 0.25       0.33333333 0.25       0.25      ]
>>  [0.33333333 0.25       0.33333333 0.25       0.25      ]]
#Transiciones desde los estados de inserción.
>> [[0.33333333 0.33333333 0.33333333 0.25       0.33333333]
>>  [0.33333333 0.33333333 0.33333333 0.5        0.33333333]
>>  [0.33333333 0.33333333 0.33333333 0.25       0.33333333]]]
>>Probabilidades de transición al estado fin:
#Se ordena los estados de forma: M, D, I.
>> [[0.83333333 0.16666667]
>>  [0.5        0.5       ]
>>  [0.5        0.5       ]]
```

Con este modelo podemos aplicar el algoritmo de Viterbi a una nueva secuencia para obtener un nuevo alineamiento:

```
result, prob = model.modifiedViterbi("ATATGTC")
print(result)
>> ['I', 'M', 'M', 'I', 'M', 'M', 'M']
```

Lo que indica que el alineamiento de la nueva secuencia con las secuencias de partida es:

$$\begin{array}{cccccccc} - & T & A & - & - & - & T & C \\ - & T & A & - & G & - & T & C \\ - & T & A & - & G & A & - & C \\ - & - & A & - & G & - & T & G \\ A & T & A & A & G & - & T & C \\ I & M & M & I & M & & M & M \end{array}$$

Podemos calcular también la probabilidad de esta secuencia respecto del modelo:

```
total_prob=model.modifiedFoward(x)
print(total_prob)
>> 1.393704882159438e-05
```

A continuación vamos a compara dos secuencias cuyas longitudes son menores que n:

```
x = "ATC"
result, prob = model.modifiedViterbi(x)
print(result)
```

```
>> ['D', 'M', 'D', 'M', 'M']
y = "AAC"
result, prob = model.modifiedViterbi(y)
print(result)
>> ['D', 'M', 'D', 'M', 'M']
```

Podemos ver que el algoritmo de Viterbi nos devuelve la misma secuencias de estados. El alineamiento de estas secuencias con las secuencias de partida es:

$$\begin{array}{cccccc} T & A & - & - & T & C \\ T & A & G & - & T & C \\ T & A & G & A & - & C \\ - & A & G & - & T & G \\ - & A & - & - & T & C \\ - & A & - & - & A & C \\ D & M & D & & M & M \end{array}$$

Observando este alineamiento parece lógico suponer que la secuencia x se ajusta más a la familia de partida que y. Comprobamos esta intuición aplicamos el algoritmo de avance:

```
total_prob=model.modifiedFoward(x)
print(total_prob)
>> 0.004601382312893029
total_prob=model.modifiedFoward(y)
print(total_prob)
>> 0.003040138977867695
```

Puesto que la probabilidad de x es superior a la de y, se cumple nuestra suposición.

CONCLUSIONES Y VÍAS FUTURAS

En este trabajo se ha realizado un estudio exhaustivo de los modelos de Markov ocultos y sus aplicaciones, centrándose especialmente en el ámbito de la biología. A partir de los conocimientos de probabilidad y de matrices positivas, se ha analizado con detalle el comportamiento de las cadenas de Markov, necesarias para el estudio de los modelos ocultos. Una vez introducida la estructura y los elementos que conforman los HMMs, se ha presentado los tres problemas asociados y los algoritmos que los resuelven. Se ha proporcionado un punto de vista matemático y probabilístico en la exposición de dichos algoritmos y se ha añadido mejoras para evitar problemas en la implementación en ordenador. Se ha explorado algunas de las aplicaciones de los HMMs en la biología y se ha desarrollado las variantes correspondientes a dichas aplicaciones. En definitiva, se ha logrado alcanzar los objetivos planteados desde el inicio del trabajo.

A lo largo del trabajo se ha podido observar la capacidad de los HMMs para adaptar a problemas concretos. Por ello, existe una gran cantidad de variantes como HMM generalizado (GHMM), HMM sensible de contexto (csHMM), etc. Como vías futuras se podría explorar más variantes de HMMs. También sería interesante estudiar modelos cuyas emisiones siguen una distribución de probabilidad concreta como por ejemplo Poisson HMM o modelos basados en una cadena de Markov con orden superior, esto es, una cadena de Markov que tiene en cuenta más estados pasados.

BIBLIOGRAFÍA

[1] Acedo, L. (2019). A hidden markov model for the linguistic analysis of the voynich manuscript. *Mathematical and Computational Applications*, *24*(1). Disponible en: https://www.mdpi.com/2297-8747/24/1/14.

[2] Axelson-Fisk, M. (2015). *Comparative Gene Finding: Models, Algorithms and Implementation*. Computational Biology. Springer London. Disponible en: https://link.springer.com/book/10.1007/978-1-4471-6693-1.

[3] Barbosa Correa, R. (2016). *Procesos estocásticos con aplicaciones*. Universidad del Norte. Disponible en: https://elibro.net/es/lc/ugr/titulos/70066.

[4] Baum, L. E., & Petrie, T. (1966). Statistical Inference for Probabilistic Functions of Finite State Markov Chains. *The Annals of Mathematical Statistics*, *37*(6), 1554 – 1563.
URL https://doi.org/10.1214/aoms/1177699147

[5] Brassard, G., & Bratley, P. (1997). *Fundamentos de algoritmia*. Prentice-Hall.

[6] Churchill, G. A. (1989). Stochastic models for heterogeneous dna sequences. *Bulletin of mathematical biology*, *51*(1), 79–94.

[7] Devijver, P. A. (1985). Baum's forward-backward algorithm revisited. *Pattern Recognition Letters*, *3*(6), 369–373. Disponible en: https://www.sciencedirect.com/science/article/pii/0167865585900236.

[8] Domínguez García, J. L., & García Planas, M. I. (2015). *Introducción a la teoría de matrices positivas: aplicaciones*. Universitat Politècnica de Catalunya. Disponible en: https://elibro.net/es/lc/ugr/titulos/52187.

[9] Durbin, R., Eddy, S., Krogh, A., & Mitchison, G. (1998). *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge: Cambridge University Press.

[10] Ewens, W. J., & Grant, G. R. (2013). *Statistical Methods in Bioinformatics: An Introduction*. Springer New York, NY. Disponible en: https://link.springer.com/book/10.1007/978-1-4757-3247-4.

[11] Ferguson, J. D. (1980). *Symposium on the Application of Hidden Markov Models to Text and Speech*. Princeton, N.J.

[12] hmmlearn. hmmlearn documentation.
URL https://hmmlearn.readthedocs.io/en/latest/

[13] hmmlearn. hmmlearn github repository.
URL https://github.com/hmmlearn/hmmlearn

[14] Häggström, O. (2002). *Finite Markov Chains and Algorithmic Applications*. Cambridge: Cambridge University Press.

[15] Jurafsky, D., & Martin, J. H. (2009). *Speech and Language Processing (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

[16] Lou, X. Y. (2017). Hidden markov model approaches for biological studies. *Biometrics & Biostatistics International Journal*, *5*(4), 132–144.

[17] National Human Genome Research Institute. Doble hélice.
URL https://www.genome.gov/es/genetics-glossary/Doble-helice

[18] NumPy. Numpy documentation.
URL https://numpy.org/doc/stable/

[19] Rabiner, L. R. (1989). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, *77*(2), 257–286.

[20] Ruiz Reina, J. L., Martín Mateos, F. J., & Graciani Díaz, C. Ampliación de Inteligencia Artificial: Modelos ocultos de Markov. https://www.cs.us.es/cursos/aia-2019/temas/tema-Markov.pdf. Accedido el 10-01-2023.

[21] Russell, S., & Norvig, P. (2010). *Artificial Intelligence A Modern Approach*. Prentice Hall, 3 ed.

[22] Salinelli, E., & Tomarelli, F. (2014). *Discrete Dynamical Models*. Springer Cham.

[23] SciPy. Scipy documentation.
URL https://docs.scipy.org/doc/scipy/index.html

[24] Stamp, M. (2017). *Introduction to Machine Learning with Applications in Information Security*. Chapman & Hall/CRC Press.

[25] Vidyasagar, M. (2014). *Hidden Markov Processes: Theory and Applications to Biology*. Princeton University Press. Disponible en: https://ebookcentral.proquest.com/lib/ugr/detail.action?docID=1680802.

[26] Vélez Ibarrola, R. (1991). *Procesos estocásticos*. Universidad Nacional de Educación a Distancia, 2 ed.

[27] Yoon B, J. (2009). Hidden markov models and their applications in biological sequence analysis. *Current genomics*, *10*(6). Disponible en: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2766791/.