> "Cuando crees que has descubierto las probabilidades, pero la siguiente observación hace que todo tu modelo se desmorone en el caos:
> '¿Qué... ¿Eh?! ¡No puede ser!'"
>
> — Miyuki Shirogane, *Kaguya-sama: Love is War*

Modelos Ocultos de Markov (HMMs) – Extendiendo las Cadenas de Markov con estados ocultos

1. Introducción a los Modelos Ocultos de Markov (HMMs)

Un **Modelo Oculto de Markov (HMM)** es un modelo estadístico en el que se asume que el sistema que se modela es un proceso de Markov con estados no observables (ocultos). Aunque no podemos observar directamente los estados ocultos, podemos observar salidas que dependen de esos estados.

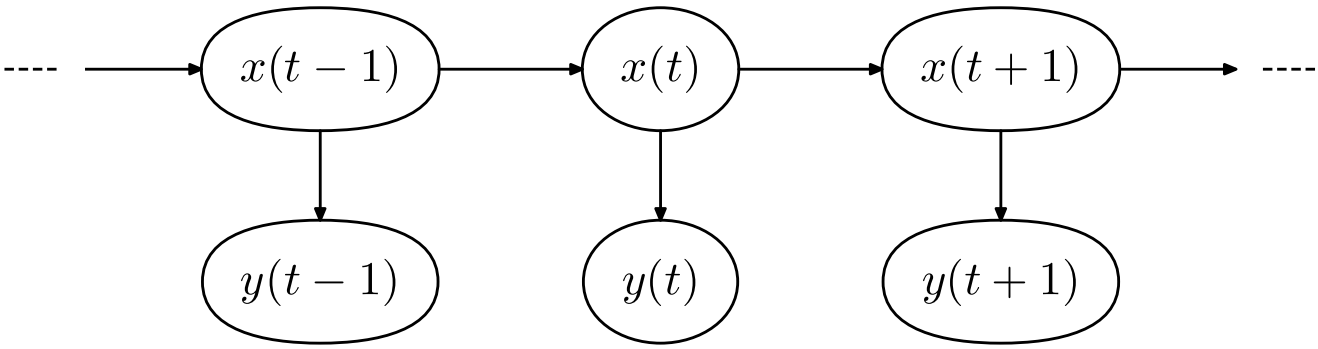


Figura: Modelo Oculto de Markov representado como una red bayesiana con cortes de tiempo. Los estados ocultos x(t) siguen la propiedad de Markov, mientras que las observaciones y(t) dependen solo del estado actual.

Fuente: Wikimedia Commons - Hidden Markov Model

Componentes clave

PROF

1. **Estados ocultos**: La secuencia de estados en la que puede estar el sistema (x_1, x_2, ..., x_n)
2. **Observaciones**: Las salidas visibles que podemos medir (y_1, y_2, ..., y_n)
3. **Probabilidades de transición**: La probabilidad de pasar de un estado oculto a otro
4. **Probabilidades de emisión**: La probabilidad de observar una determinada salida dado un estado oculto

Aplicación: Reconocimiento de voz

Una de las aplicaciones más exitosas de los HMMs es en el **reconocimiento de voz**. En este contexto:

- **Estados ocultos**: Representan los fonemas o unidades de habla subyacentes
- **Observaciones**: Las características del audio real (p.ej., propiedades espectrales)
- **Probabilidades de transición**: La probabilidad de que un fonema suceda a otro
- **Probabilidades de emisión**: La probabilidad de observar ciertas características de audio dado un fonema específico

Por ejemplo, cuando alguien dice la palabra "hello", observamos la forma de onda de audio, pero los fonemas reales (/h/, /ə/, /l/, /oʊ/) son estados ocultos que queremos inferir.

Los tres problemas básicos de los HMMs

1. **Problema de evaluación**: Dado un HMM y una secuencia de observaciones, ¿cuál es la probabilidad de que el modelo haya generado esas observaciones? (Se resuelve usando el **algoritmo Forward**)

2. **Problema de decodificación**: Dado un HMM y una secuencia de observaciones, ¿cuál es la secuencia de estados ocultos más probable que produjo esas observaciones? (Se resuelve usando el **algoritmo Viterbi**)

3. **Problema de aprendizaje**: Dada una secuencia de observaciones, ¿cuáles deben ser los parámetros del HMM para explicar mejor estas observaciones? (Se resuelve usando el **algoritmo Baum-Welch**)

Ejemplo: Predicción del clima

Imagina que queremos predecir patrones climáticos pero solo podemos observar si la gente lleva paraguas:

- **Estados ocultos**: El clima real (Soleado, Lluvioso)
- **Observaciones**: Si la gente lleva paraguas (Sí, No)
- **Probabilidades de transición**: ¿Qué probabilidad hay de que un día lluvioso sea seguido por un día soleado?
- **Probabilidades de emisión**: ¿Con qué probabilidad lleva la gente paraguas si está lluvioso vs si está soleado?

Este ejemplo sencillo ilustra cómo los HMMs pueden ayudarnos a inferir estados ocultos (el clima) a partir de datos observables (uso de paraguas).

Para reforzar la intuición, considera un escenario divertido de anime: *Kaguya-sama: Love is War*. En esta historia, los verdaderos sentimientos de cada personaje (por ejemplo, *"¿Está Kaguya secretamente enamorada de Miyuki?"*) son estados ocultos — guardan esos estados internos en secreto. Lo que observamos son sus acciones y palabras, que a menudo son "señales" engañosas.

PROF

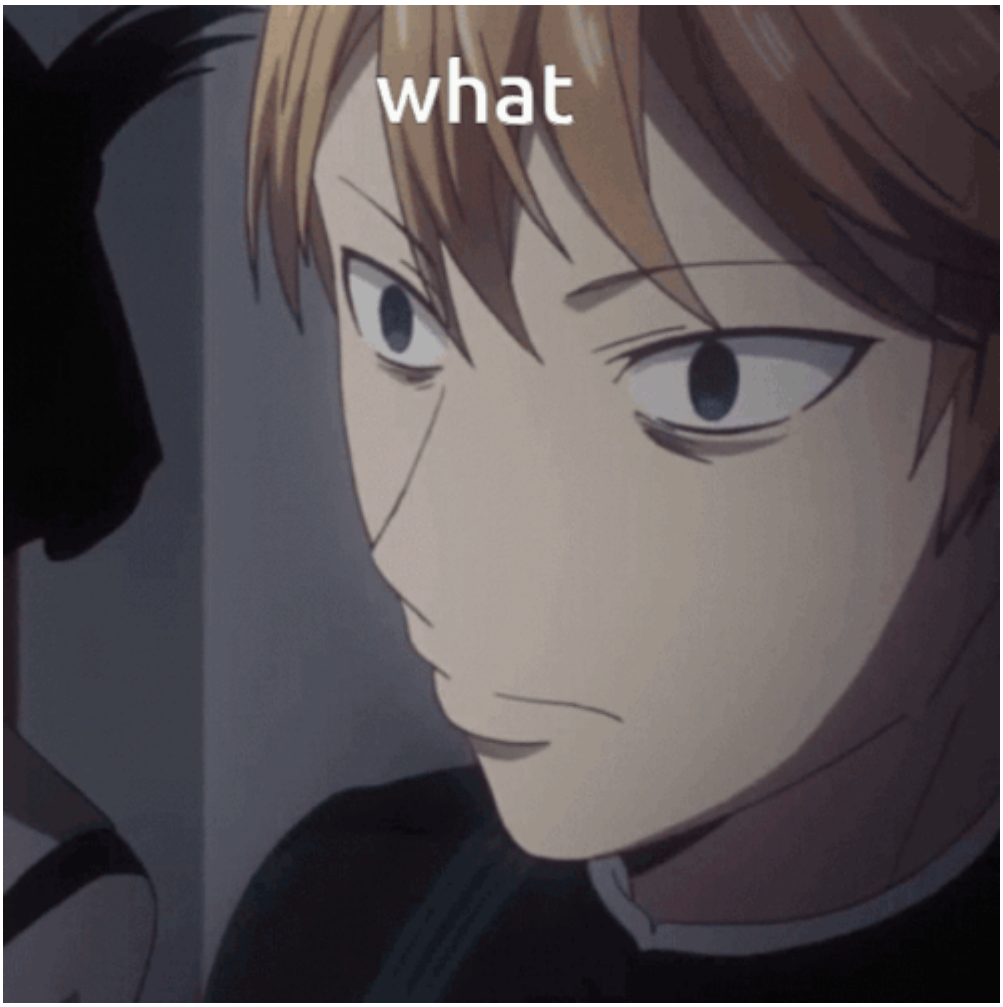


Figura: La reacción del Presidente Shirogane cuando su modelo probabilístico de los sentimientos de Kaguya se rompe por completo: ¡una representación perfecta de cuando las predicciones de nuestro HMM se enfrentan a observaciones inesperadas!

Si Kaguya molesta a Miyuki o se sonroja, esas son observaciones a partir de las cuales intentamos inferir su estado emocional oculto (amor o no amor). **Intuición de HMM**: tenemos una secuencia de acciones observables (diálogo, lenguaje corporal) generadas por una secuencia de estados emocionales no observados. Como espectadores (o como observadores de IA), mantenemos una **creencia** (b(s)) sobre el estado, que esencialmente es una probabilidad para cada estado posible dadas las observaciones hasta el momento. Esta creencia se actualiza conforme llegan nuevas observaciones. (En términos de HMM, (b(s_t) = P(s_t = \text{"enamorada"} \mid \text{todas las observaciones hasta el tiempo } t)), por ejemplo.)

PROF

Otro ejemplo relatable es interpretar el estado de ánimo de un amigo a partir de sus mensajes de WhatsApp. No puedes leer directamente su mente (estado oculto: feliz, triste, etc.), pero ves su estilo de escritura, el uso de emojis y el tiempo de respuesta (observaciones). A lo largo de una conversación (una secuencia de observaciones), infieres una secuencia probable de estados de ánimo. Tal vez respuestas cortas con "..." y sin emojis sugieren que tu amigo está en un estado malhumorado; un repentino "LOL" indica una transición a un estado más alegre. Aquí estás **filtrando** observaciones mediante un HMM en tu cabeza — usando las *señales observables* para adivinar el *contexto oculto*.

Formalmente, un HMM suele describirse como un proceso doblemente estocástico: un **proceso de estado** oculto que evoluciona con transiciones de Markov, y un **proceso observable** que produce emisiones desde cada estado (

Hidden Markov Models and their Applications in Biological Sequence Analysis - PMC
). Normalmente se denota (\lambda = (S, O, T, E)) donde:

- (S) es el conjunto de estados ocultos,
- (O) es el conjunto de observaciones,
- (T = {t(s,s')}) es la matriz de probabilidad de transición de estados (similar a la matriz de transición de una cadena de Markov),
- (E = {e(o|s)}) es el conjunto de probabilidades de emisión para las observaciones.

En cada paso de tiempo, el HMM transita del estado (s) a un nuevo estado (s') según (t(s,s')), y luego genera una observación (o) con probabilidad (e(o|s')). La **propiedad de Markov** sigue cumpliéndose para la secuencia de estados ocultos: el siguiente estado depende solo del estado actual, no de toda la historia (Hidden Markov model - Wikipedia). Sin embargo, a diferencia de una cadena de Markov visible, nosotros como observadores solo vemos las observaciones (o_t), no los estados (s_t).

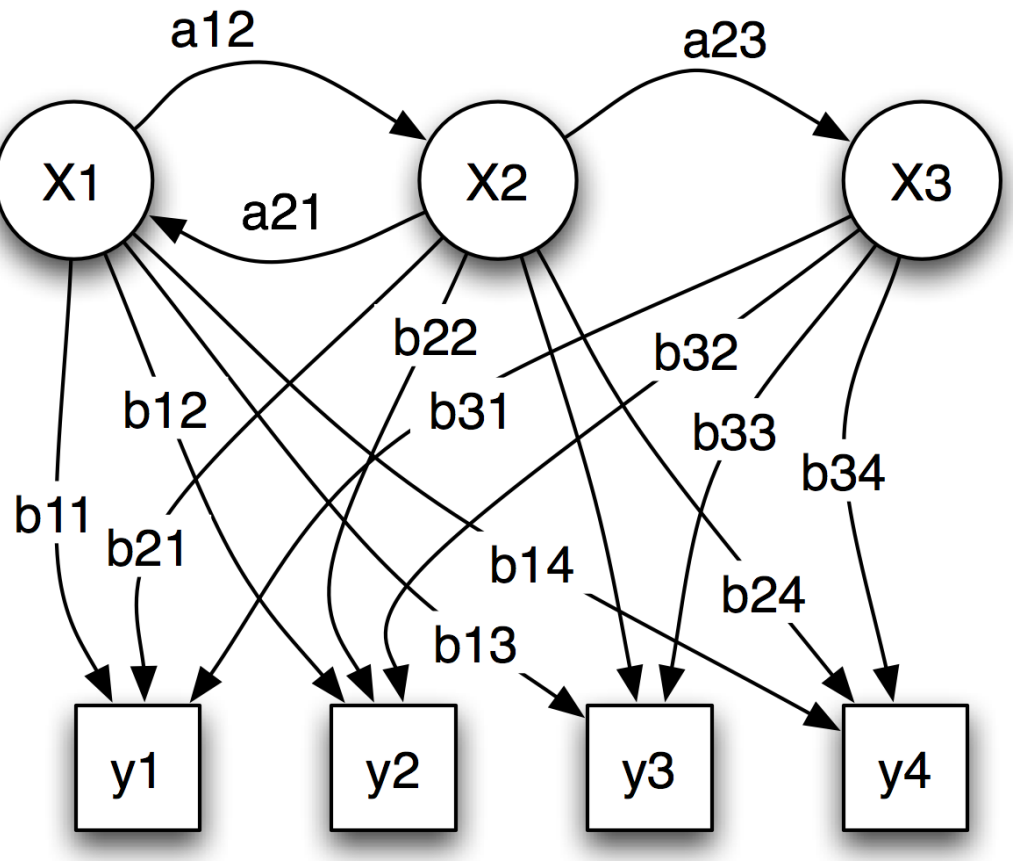


Figura 1: Ilustración de la estructura estado-observación de un HMM. Los estados ocultos (X_1, X_2, X_3) (círculos) tienen transiciones entre ellos (flechas curvas con etiquetas (a_{ij})), similar a una cadena de Markov. Cada estado puede emitir símbolos observables (y_1, y_2, y_3, y_4) (cuadrados) con ciertas probabilidades (b_{ij}). No podemos ver directamente los estados (X) — solo vemos la secuencia de salidas (y).

Fuente: Hidden Markov Model - Wikipedia

PROF

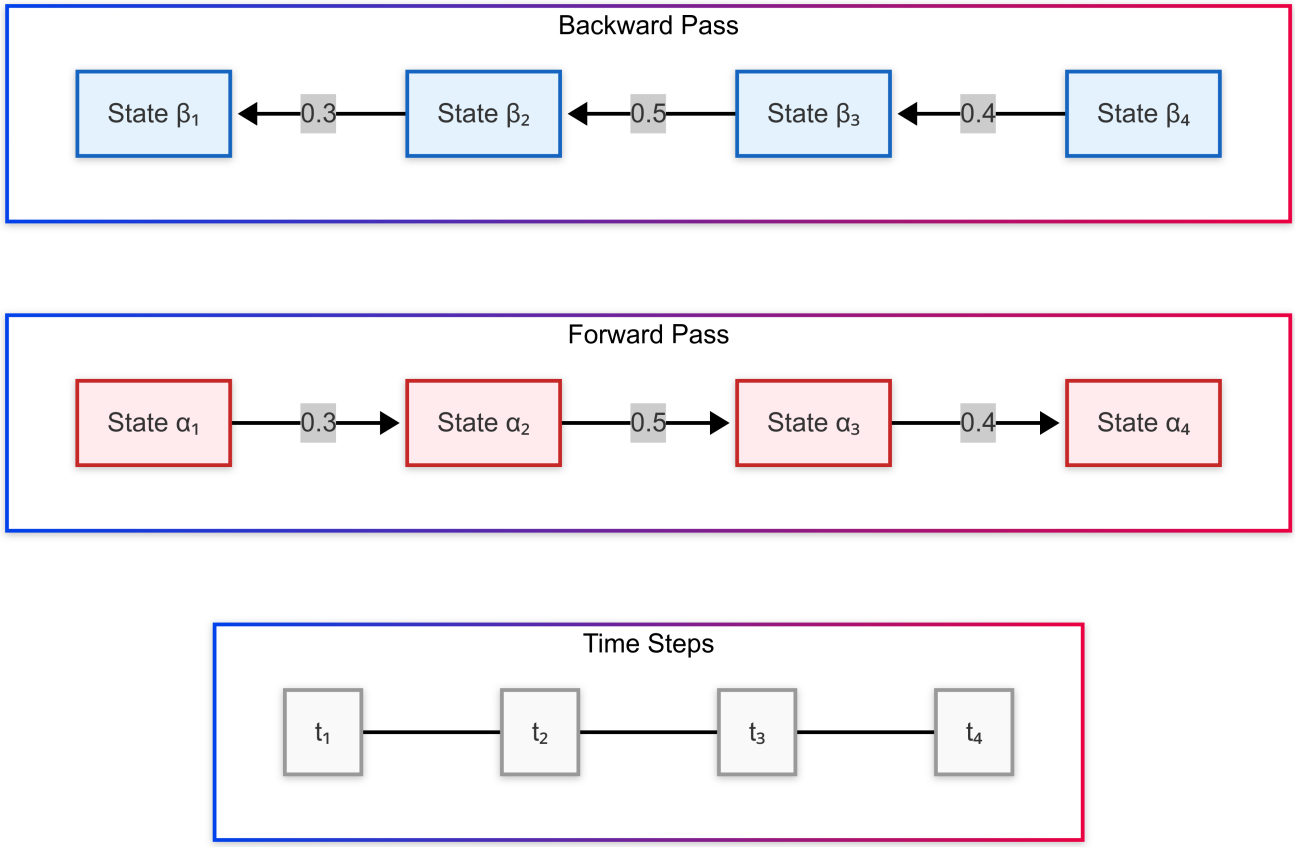


Figura 2: El algoritmo Forward-Backward en acción. La pasada hacia adelante (α) calcula probabilidades desde el inicio hasta el tiempo actual, mientras que la pasada hacia atrás (β) lo hace desde el final hasta el tiempo actual. Esto nos permite combinar información tanto de observaciones pasadas como futuras para inferir el estado en cualquier momento t.

Fuente: Forward-Backward Algorithm - Wikipedia

En resumen, un HMM es una **cadena de Markov con estados ocultos** (7.3: Markov Chains and HMMS - From Example to Formalizing - Biology LibreTexts). Modela situaciones donde el estado verdadero de un sistema evoluciona con dinámica de Markov, pero un agente solo recibe *emisiones* (lecturas de sensores, síntomas, palabras habladas, etc.) que dependen de esos estados. A continuación veremos cómo **diseñar** un HMM para un problema y luego cómo **calcular** cantidades útiles a partir de él.

PROF

2. Diseñando HMMs

Figura: Kaguya mostrando un comportamiento observable (beber té con nervios) mientras su verdadero estado emocional permanece oculto — una perfecta ilustración de los estados ocultos de un HMM y sus salidas observables.

PROF

Diseñar un HMM para un problema del mundo real implica decidir cuáles serán los estados ocultos y las observaciones, y estimar las probabilidades de transición y emisión a partir de datos. Comenzamos identificando el proceso oculto que nos interesa y las señales que podemos observar.

- **Estados ocultos ((S))**: Estos deben representar de manera significativa la condición subyacente que cambia con el tiempo. Por ejemplo, en un HMM de reconocimiento de voz, los estados ocultos podrían ser fonemas o palabras que se pronuncian en cada momento (Hidden Markov Models and their Applications in Biological Sequence Analysis - PMC). En nuestra analogía de Kaguya-sama, el estado oculto podría ser la emoción o intención verdadera de un personaje en una escena dada. En un modelo de clima, el estado oculto es el clima real (que no vemos directamente si estamos en interiores). Escoger la representación adecuada de estados es parte del arte: los estados deben ser lo bastante distintos como para producir observaciones diferentes.

- **Observaciones ((O))**: Estos son los datos que realmente vemos. En voz, las observaciones podrían ser características de la señal de audio en cada instante (las frecuencias del sonido). En Kaguya-sama, las observaciones son las acciones o diálogos de los personajes (que *insinúan* sus sentimientos). Para un chatbot que analiza mensajes de WhatsApp, las observaciones podrían ser el tono de texto o la cantidad de emojis en cada mensaje. El espacio de observación puede ser de símbolos discretos o valores continuos. A menudo discretizamos o categorizamos las observaciones para ajustarlas a un HMM (p.ej., "el tono del mensaje es positivo/negativo/neutral" como categorías discretas de observación).

- **Probabilidades de transición ((T))**: Debemos definir qué tan probable es que el estado oculto se mantenga o cambie. Este es el *modelo de transición de estado*. Por ejemplo, si modelamos el estado de ánimo diario como estados, (t(s,s')) podría reflejar "si alguien hoy está feliz, hay un 80% de probabilidad de que mañana siga feliz, 20% de que se vuelva neutral y muy poca probabilidad de saltar directamente a triste". Estas probabilidades pueden estimarse de datos etiquetados si están disponibles (con qué frecuencia vemos el estado *s* seguido por *s'* en secuencias de entrenamiento). Al diseñar, podrías partir de una suposición o conocimiento del dominio. Si los datos no están etiquetados (estados no identificados), aprenderemos estos parámetros con algoritmos como Baum-Welch (más adelante).

- **Probabilidades de emisión ((E))**: Este es el *modelo de observación*: para cada estado, qué tan probable es cada observación. Por ejemplo, si el estado oculto es "Kaguya está secretamente enamorada", ¿cuál es la probabilidad de que haga un comentario burlón (observación1) frente a darle un cumplido (observación2)? O, si el estado es "lluvioso", ¿qué probabilidad hay de observar "persona con paraguas" vs "sin paraguas"? Estas probabilidades (e(o|s)) se pueden estimar a partir de datos viendo con qué frecuencia se produce cada observación cuando el sistema está en el estado *s*. Si tenemos datos pareados (estado-observación), contamos frecuencias directamente. Si no, se tratan como incógnitas que aprendemos con algoritmos de entrenamiento.

Construir un HMM significa **modelar un entorno parcialmente observable y estocástico**. "Parcialmente observable" porque el agente no ve directamente el estado (); "estocástico" porque las transiciones de estado y las observaciones son probabilísticas, no deterministas. Aun así, esto es muy útil en IA porque muchos procesos del mundo real tienen factores ocultos y aleatoriedad.

PROF

Algunos escenarios concretos:

- **Inferencia de estrategia en juegos**: Considera un videojuego multijugador o un juego de mesa. La estrategia real de tu oponente (agresiva, defensiva, farol, etc.) está oculta. Solo ves sus movimientos (observaciones). Un agente de IA podría usar un HMM donde los estados representan la estrategia o “tipo” de oponente (), y las observaciones son sus acciones cada turno. Modelando probabilidades de transición, la IA infiere si el oponente es propenso a cambiar de estrategia a mitad de partida. Por ejemplo, en **póker**, las cartas del oponente y si está faroleando son estados ocultos; sus patrones de apuestas son observaciones. Con el tiempo, un HMM puede ayudar a detectar patrones (quizá cierto patrón de apuestas indica un farol) y actualizar la creencia sobre la fuerza de la mano del oponente. Esto es esencialmente una aplicación de HMM en *teoría de juegos* simplificada.

- **Análisis de tendencias en redes sociales**: Imagina una IA que monitorea Twitter (X) para detectar temas en tendencia. El *estado de tendencia* (qué tema está "caliente") es oculto en medio de millones de publicaciones. Las observaciones podrían ser las frecuencias de ciertas palabras clave o hashtags. Un HMM podría tener estados como "Tema A en tendencia", "Tema B en tendencia", etc., o estados más abstractos como "auge de la tendencia vs. en declive". Las probabilidades de transición modelan cómo las tendencias suben o bajan (p.ej., una tendencia a menudo se mantiene un tiempo o pasa de creciente a decreciente). Las probabilidades de emisión modelan la probabilidad de ver ciertos hashtags o palabras clave dado un estado de tendencia. Al aprender de datos, la IA puede detectar cuándo ocurre un cambio de estado (por ejemplo, cuando los tuits sobre #GameStop de repente se disparan, pasando a un estado de "GameStop-hype"). Esto demuestra cómo los HMMs pueden detectar *cambios en contextos ocultos* como el interés público, aunque solo veamos observaciones ruidosas (tuits).

- **Seguimiento del estado emocional**: (Regresando a nuestro ejemplo de WhatsApp) Supongamos que diseñamos un HMM para interpretar la trayectoria emocional de un amigo a lo largo de una semana de chats. Definimos estados ocultos como {Feliz, Neutral, Estresado, Triste}, y observaciones derivadas de los mensajes: quizá {emoji positivo, texto neutral, palabras negativas, respuesta lenta, etc.}. Con datos de chats pasados (si tuviéramos etiquetas del estado de ánimo real o autorreportes), podríamos estimar las probabilidades de transición (quizá "Estresado" tiende a persistir 2-3 días, luego pasa a "Triste" con prob. 0.5 o regresa a "Neutral" con 0.5) y las de emisión (cuando "Feliz", 70% de emojis positivos; cuando "Triste", 50% de respuestas lentas, etc.). El resultado es un HMM que un chatbot de IA podría usar para **inferir el estado de ánimo actual** de tu amigo a medida que la conversación avanza, a pesar de no observarlo directamente. Así funciona un *filtro* mental basado en HMM: el sistema usa las observaciones para adivinar el estado oculto.

Al diseñar HMMs, **el conocimiento del dominio** es muy valioso para elegir los estados y dar forma al modelo. A veces los estados ocultos corresponden a realidades físicas (como "llueve/no llueve" en el ejemplo del paraguas), pero otras veces son abstractos (como "régimen 1" vs. "régimen 2" en un modelo financiero, ver Hidden Markov Models - An Introduction | QuantStart). Puede que no sepamos exactamente qué significa cada estado oculto, pero si el HMM con 2 o 3 estados ajusta bien los datos, igual puede ser útil para predecir.

PROF

También conviene tener en cuenta que los HMMs suponen que el proceso es *sin memoria* (Markov) y *estacionario* (las probabilidades de transición no cambian con el tiempo). En la realidad, estas suposiciones pueden ser aproximadas. Por ejemplo, la transición del estado de ánimo de una persona podría no ser estrictamente de primer orden (el estado de anteayer puede influir en el de mañana), o las probabilidades pueden cambiar según la estación del año. Aun así, los HMMs suelen servir como un modelo razonable y más simple que capta la dinámica esencial. Han sido **extremadamente populares en IA** en áreas como reconocimiento de voz, bioinformática y procesamiento del lenguaje porque equilibran el rigor matemático con la capacidad de manejar información oculta (Hidden Markov Models and their Applications in Biological Sequence Analysis - PMC).

3. Cálculo con HMMs

Figura: Hayasaka analizando situaciones con destreza — representa cómo desciframos estados ocultos a partir de una secuencia de observaciones.

PROF

Una vez que tenemos un HMM, hay tres problemas computacionales fundamentales que solemos querer resolver:

1. **Verosimilitud (Likelihood)**: Calcular la probabilidad de una secuencia de observaciones, (P(o_1, o_2, \dots, o_T)), dado el modelo. En otras palabras, cuán *probable* es que nuestro HMM produzca cierta secuencia de observaciones. A esto a menudo se le llama el **problema de evaluación**. Por ejemplo, dado un HMM que modela oraciones en inglés, ¿cuál es la probabilidad de que genere la secuencia observada "I lost my phone"? Este cálculo suma sobre todas las posibles secuencias de estados ocultos que podrían producir esas observaciones. Hacerlo por fuerza bruta es exponencial ((N^T) secuencias si hay N estados y T pasos de tiempo), pero hay un método de programación dinámica llamado **algoritmo Forward** (Forward algorithm - Wikipedia). El algoritmo Forward calcula de forma eficiente la verosimilitud (P(o_{1:T})) acumulando iterativamente las probabilidades de estar en cada estado en el tiempo *t* y emitir (o_t). Esencialmente propaga un **estado de creencia** hacia adelante (Forward algorithm - Wikipedia). Al

final obtenemos (P(o_{1:T})). Esto es útil, por ejemplo, para comparar modelos o detectar anomalías (una secuencia de observaciones con probabilidad extremadamente baja podría indicar algo inesperado, como en reconocimiento de gestos un movimiento imposible).

2. **Inferencia de estados**: Determinar el estado oculto más probable (o distribución de probabilidad sobre estados) dadas las observaciones. Esto puede significar calcular la **distribución de creencia** (b(s_t) = P(s_t \mid o_1,\dots,o_t)) en el tiempo *t* (filtrado), o la probabilidad de cada estado en *t* dados *todas* las observaciones (suavizado), o hallar la secuencia de estados ocultos más probable (\arg\max_{s_{1:T}} P(s_{1:T} \mid o_{1:T})) (conocido como **decodificación**). Por ejemplo, un reconocedor de voz de IA puede actualizar continuamente su creencia sobre qué palabra se está pronunciando al llegar nueva señal de audio (filtrado). Más tarde, tras escuchar toda la frase, puede revisar su conjetura de palabras anteriores (suavizado). El **algoritmo Forward-Backward** proporciona un método eficiente para hacer suavizado combinando probabilidades forward (de principio a t) y backward (de final a t) para cada instante (Forward algorithm - Wikipedia). Por otro lado, para obtener la secuencia de estados ocultos más probable, usamos el **algoritmo Viterbi** (Viterbi algorithm - Wikipedia). Viterbi es un algoritmo de programación dinámica que rastrea el camino de mayor probabilidad de estados que podrían producir las observaciones, guardando en cada paso la trayectoria más probable y haciendo *backtracking* (Viterbi algorithm - Wikipedia). Si el algoritmo Forward es como *sumar todos los ritmos que podrían producir una canción medio apagada*, Viterbi es como *encontrar la única melodía más probable que encaja con esas notas apagadas*. En términos prácticos, si nuestro HMM rastrea la intención del usuario a partir de movimientos del ratón, Viterbi podría darnos la secuencia de intenciones más probable ("seleccionar -> arrastrar -> soltar") que explica la trayectoria observada del cursor.

3. **Aprendizaje (Estimación de parámetros)**: Descubrir los mejores parámetros del modelo (las (t(s,s')) y (e(o|s)), e incluso el conjunto de estados) a partir de datos. Este es el **problema de entrenamiento** de los HMMs. Si tenemos un conjunto de secuencias de observaciones (y quizá algo de conocimiento parcial de los estados), ¿cómo ajustamos el HMM para que represente esos datos? La solución clásica es el **algoritmo Baum–Welch**, que es un caso particular del algoritmo de **Expectation-Maximization (EM)** para HMMs (Baum–Welch algorithm - Wikipedia). Baum–Welch comienza con una suposición inicial de los parámetros y luego los mejora iterativamente: en la fase E usa el modelo actual para calcular el conteo esperado de transiciones y emisiones (a menudo con el procedimiento forward-backward para manejar estados ocultos), y en la fase M actualiza las probabilidades para maximizar la verosimilitud de las observaciones dadas esas cuentas esperadas (Baum–Welch algorithm - Wikipedia). Con iteraciones, el modelo normalmente converge a un conjunto de parámetros que explican bien los datos de entrenamiento. Por ejemplo, si estamos entrenando un HMM para reconocimiento de gestos, Baum–Welch ajustará las probabilidades de transición entre "gestos" y las probabilidades de emisión de varios eventos táctiles para cada gesto, hasta que el modelo se ajuste a todas las secuencias de entrenamiento. Se puede pensar en Baum–Welch como *una IA que intenta aprender las emociones ocultas de la gente a lo largo del tiempo observando sus mensajes*: comienza con suposiciones aleatorias y, con cada nueva conversación, refina su idea de cómo los estados llevan a ciertas palabras (y cómo se suceden) para converger en un modelo cada vez más preciso.

PROF

Una analogía rápida por algoritmo:

- **Algoritmo Forward (Filtrado)**: Imagina que escuchas una canción en la habitación de al lado. Oyes parte de la melodía (observaciones) pero algunas notas están apagadas. Mantienes una creencia de qué acorde o nota es probable ahora, según lo que has oído. A medida que entra cada nuevo sonido, actualizas esa creencia. Esto es como el filtrado: actualizar progresivamente (b(s)) conforme se acumula evidencia (Forward algorithm - Wikipedia).

- **Algoritmo Viterbi (Decodificación)**: Ahora piensa que tratas de identificar la canción exacta. Consideras mentalmente todas las canciones posibles y encuentras la que mejor encaja con los sonidos apagados que escuchas. Devuelves la canción más probable (secuencia de estados). Eso equivale a Viterbi: encontrar la secuencia oculta más probable que explique las observaciones (Viterbi algorithm - Wikipedia). En un contexto de IA, Viterbi puede usarse para decodificar la secuencia de palabras más probable a partir del audio — básicamente lo que hacían muchos sistemas de reconocimiento de voz basados en HMM (Viterbi algorithm - Wikipedia).

- **Algoritmo Baum–Welch (Aprendizaje)**: Supón que tratas de aprender los patrones de humor de un amigo sin que él te lo diga directamente. Al principio, adivinas al azar cómo se transita de un estado de ánimo a otro y qué palabras usa en cada estado. Cada día observas sus mensajes y actualizas tu conjetura ("Mmm, dijo 'estoy cansado', eso suele esperarse más en el estado Triste, quizás deba aumentar la probabilidad de ese estado hoy"). Con el paso de semanas, refinas estas probabilidades para predecir mejor su estado de ánimo. Baum–Welch hace esto de forma sistemática en un HMM, usando todos los datos de observación para ajustar los parámetros del modelo hacia una mayor verosimilitud (Baum–Welch algorithm - Wikipedia).

En lo matemático, gran parte de estos algoritmos implica sumar y multiplicar probabilidades, pero la programación dinámica los hace eficientes (normalmente $O(N^2 \cdot T)$, donde N es el número de estados y T la longitud de la secuencia, para algoritmos como Forward o Viterbi). Las implementaciones de software para HMM suelen manejar detalles como reescalado (para evitar subdesbordamientos de probabilidades muy pequeñas) (Baum–Welch algorithm - Wikipedia). El resultado es que, incluso con 5 estados y 1000 observaciones, podemos computar las cosas en milisegundos en lugar de revisar millones de trayectorias.

PROF

Ventajas y desventajas: Los HMMs son relativamente eficientes y bien entendidos. Pueden modelar secuencias de forma interpretable (a menudo puedes asignar un significado claro a los estados). Los métodos de programación dinámica (Forward, Viterbi, etc.) garantizan resultados óptimos para sus problemas respectivos (verosimilitud, mejor trayectoria) dado el modelo. Los HMMs manejan el ruido y la incertidumbre de forma elegante usando probabilidades. Sin embargo, tienen limitaciones: la suposición de Markov (el estado depende solo del anterior) puede ser demasiado simple para tareas con dependencias de largo alcance. Además, los HMMs estándar asumen que las probabilidades no cambian con el tiempo (estacionariedad), lo cual podría no cumplirse si el comportamiento del proceso varía (p.ej., un HMM entrenado de lunes a viernes quizá no se ajuste directamente al comportamiento de fin de semana). Otra consideración es que si el espacio de observaciones es muy grande o continuo, puede requerirse un modelo de emisión más complejo (como mixtures gaussianas), complicando el entrenamiento. Además, entrenar con Baum–Welch (EM) puede atascarse en óptimos locales, así que quizás no halle el mejor modelo global, sino uno "bueno" para los datos. Pese a ello, los HMMs siguen siendo una herramienta fundamental para datos secuenciales en IA, y entenderlos sienta las bases para modelos más complejos.

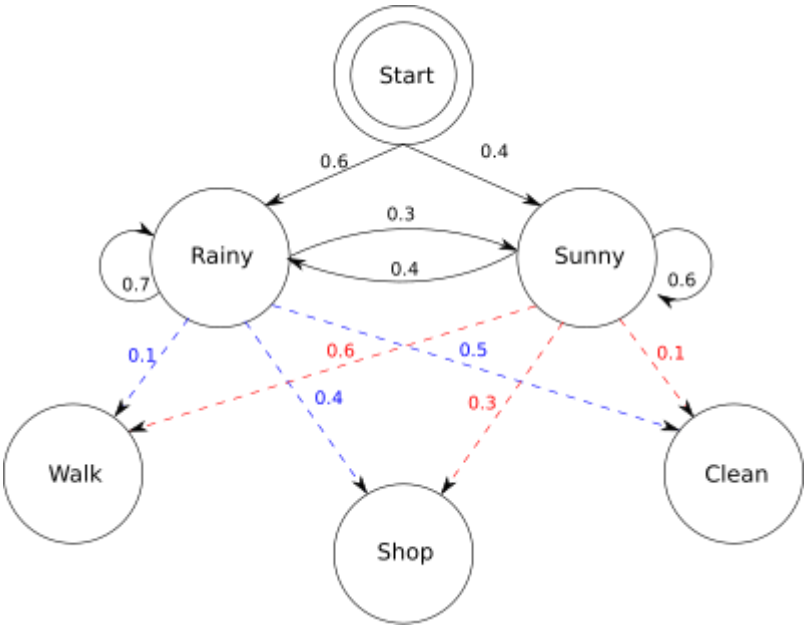


Figura 3: El algoritmo Viterbi halla la secuencia de estados más probable a través de un enrejado de posibilidades. Cada nodo representa un estado en un tiempo particular, y las aristas muestran transiciones. El algoritmo calcula de forma eficiente la mejor trayectoria (resaltada) al llevar registro de la máxima probabilidad de llegar a cada estado en cada tiempo, y luego retrocede para reconstruir el camino.

Fuente: Viterbi Algorithm - Wikimedia Commons

4. Inferencia y agentes de IA

PROF

Figura: Chika en modo detective — representa cómo evaluamos e investigamos patrones en las observaciones para ver si encajan en las expectativas de nuestro modelo.

Ahora que entendemos los HMMs y cómo calcular con ellos, veamos cómo los **agentes de IA** pueden aprovechar la inferencia en HMM para la toma de decisiones. En IA (como describen muchos libros de texto), hay diferentes tipos de agentes: Reflex simples, Basados en modelo, Basados en objetivos, Basados en utilidad y agentes de Aprendizaje. Los HMMs pueden desempeñar un papel en los más sofisticados, que necesitan manejar información de estado oculto. Los repasamos:

- **Agentes de Reflex simples**: Escogen acciones basadas solo en la *observación* actual, sin estado interno. No mantienen un HMM ni ningún modelo del mundo — simplemente aplican reglas “si

veo X, hago Y". Dado que los HMMs se centran en estimar el estado oculto, un agente reflex simple no los usa explícitamente. Por ejemplo, un filtro de spam que mira un email (observación) y decide "spam" o "no spam" es reflexivo. Sin embargo, si forzamos un poco la definición, podría mejorarse usando un HMM en segundo plano que provea una observación refinada: p.ej., un sistema de notificaciones de WhatsApp que se active solo si la creencia de un HMM indica que tu amigo está molesto. Aun así, estrictamente, el agente reflex simple no mantiene memoria de secuencias, así que no suele necesitar un HMM (que se alimenta de historial).

- **Agentes de Reflex basados en modelo**: Estos mantienen un estado interno para manejar la observabilidad parcial. ¡Aquí brillan los HMMs! Un agente basado en modelo tiene un modelo de cómo funciona el mundo (transiciones) y de cómo las observaciones se relacionan con el estado (emisiones) (). Eso es esencialmente un HMM. El agente usa el modelo para **actualizar su estado de creencia** cada vez que recibe una observación. Por ejemplo, considera una **IA de reconocimiento de voz** en un dispositivo de hogar inteligente. El dispositivo no conoce directamente qué palabra se dice (estado oculto), solo recibe las señales de audio (observaciones). Mantiene una creencia sobre qué palabra/fonema se está pronunciando usando un modelo (HMM). Esto es un agente reflex basado en modelo: su decisión (p.ej., ejecutar un comando de voz) se basa en el estado inferido (la palabra detectada) y no solo en la forma de onda bruta (
  Hidden Markov Models and their Applications in Biological Sequence Analysis - PMC
). Otro ejemplo: un robot que navega por un edificio con visión borrosa. Lleva en su interior un HMM sobre su ubicación: estados = posiciones posibles, observaciones = lo que ve con su cámara borrosa. El robot actualiza su creencia de dónde está a medida que se mueve y obtiene observaciones nuevas (algoritmo Forward). Sus acciones (como "girar a la izquierda") se basan en esa creencia. En definitiva, los agentes basados en modelo **usan la inferencia de HMM para gestionar el estado oculto**, haciéndolos más efectivos que los reflex simples.

- **Agentes basados en objetivos (Goal-Based)**: Estos agentes no solo rastrean el estado sino que eligen acciones para lograr objetivos. En un entorno parcialmente observable, se basan en la creencia (calculada por el HMM) para llevar a cabo *planificación orientada a objetivos*. Por ejemplo, imagina un agente cuyo objetivo es encontrar un tesoro en un laberinto, pero no puede ver todo el laberinto (solo observaciones locales). Su HMM podría rastrear una probabilidad sobre dónde está el tesoro (estado oculto) según pistas (observaciones). Con un objetivo ("encontrar tesoro"), el agente planifica acciones que maximicen la probabilidad de llegar a la ubicación del tesoro según su creencia. Otro caso: un dron autónomo que busca entregar un paquete a una persona en una multitud. La ubicación de la persona es un estado oculto; el dron recibe observaciones parciales (reconocimiento facial confuso, color de ropa). El HMM mantiene la creencia de dónde podría estar la persona. Un enfoque *goal-based* usa esa creencia para planear movimientos de búsqueda (si se cree que la persona está al este, vuela hacia allá). En esencia, el agente considera *estados de creencia* al buscar un plan que cumpla el objetivo.

- **Agentes basados en utilidad (Utility-Based)**: Estos extienden los basados en objetivos teniendo una función de utilidad que valora qué tan buenas son distintas situaciones. En un contexto parcialmente observable, el agente usará la creencia (del HMM) para calcular utilidades esperadas de las acciones. Por ejemplo, en póker, un agente IA puede tener una utilidad que representa su ganancia monetaria. Desconoce las cartas del oponente (estado oculto) pero mantiene una creencia sobre la mano del oponente basada en las apuestas observadas. Al decidir

apostar o retirarse, el agente sopesa la utilidad esperada: si cree con un 70% que el oponente está faroleando (mano débil) y 30% que es fuerte, usará esas probabilidades para elegir la acción de mayor rendimiento esperado. Aquí, **el HMM aporta (b(s))** (p.ej., "70% farol, 30% mano fuerte"), y el agente basado en utilidad combina eso con los posibles resultados de sus acciones para maximizar la utilidad. Otro ejemplo: un agente de diagnóstico médico donde el estado del paciente es oculto; el HMM lo estima a partir de síntomas. Un enfoque de utilidad consideraría tratamientos (acciones) y, usando la creencia, calcula qué tan efectivos o dañinos podrían ser, eligiendo el de mejor utilidad esperada.

- **Agentes de Aprendizaje**: Estos agentes pueden mejorar su desempeño aprendiendo con la experiencia. Con respecto a HMMs, un agente de aprendizaje podría **ajustar los parámetros (o la estructura) del HMM** sobre la marcha al recopilar más datos. Imagina un asistente personal de IA que aprende la rutina del usuario (estados ocultos = “en el trabajo”, “en casa”, “yendo al gimnasio”; observaciones = sensores del teléfono, uso de apps). Al principio tiene un HMM genérico, pero a medida que observa la conducta del usuario, aprende que va al gimnasio a las 6 pm (ajusta la probabilidad de transición para reflejar ese hábito). Esto podría hacerse con una versión en línea de Baum-Welch o conteo simple si se conocen los estados. Otro ejemplo es si el agente descubre un nuevo estado u observación que no estaba contemplado — un agente de aprendizaje podría ampliar su HMM para incluirlo. Por ejemplo, un sistema de reconocimiento de voz que se adapta al acento del usuario con el tiempo ajustando las probabilidades de emisión para ciertos fonemas. Incluso se podría redefinir la cantidad de estados si los datos sugieren la necesidad de dividir un estado en dos. En conclusión, un agente de aprendizaje usa retroalimentación del entorno (o señales de recompensa) para refinar las matrices (T) y (E) del HMM, mejorando con el tiempo su precisión de inferencia.

Volviendo a la analogía con la teoría de juegos, considera **detectar faroles en póker**. Un agente de aprendizaje basado en utilidad podría usar un HMM para modelar el estilo de juego de un oponente (estado oculto = tendencia a farolear vs. jugar recto). Inicialmente, no conoce al oponente, así que parte de un modelo genérico. A medida que juega varias rondas (observa las apuestas y a veces las cartas en el showdown), *aprende* las transiciones y emisiones ocultas: por ejemplo, el oponente farolea más tras perder una mano grande (cambio de estado). El agente actualiza su HMM y lo usa para tomar mejores decisiones la próxima vez (inferir probabilidad de farol). Con muchas partidas, entrena un HMM adaptado específicamente a ese oponente, anticipando sus faroles mejor que un agente sin aprendizaje.

PROF

Más allá de los HMM: MDP/POMDP: Hasta ahora asumimos que las acciones del agente no afectan las transiciones de estado (hemos visto observación pasiva o decisiones unilaterales). En un **MDP (Markov Decision Process)**, las acciones del agente influyen en las transiciones de estado (es como una cadena de Markov controlada). Si el agente puede observar completamente el estado, tenemos un MDP. Si no, surge un **POMDP (Partially Observable Markov Decision Process)**, que es básicamente un MDP + un HMM para la percepción () (). En un POMDP, el agente mantiene una creencia (como (b(s)) de filtrado en HMM) y elige acciones basadas en esa creencia. Nuestra discusión sobre agentes basados en objetivos o utilidad con incertidumbre es esencialmente un marco de POMDP: el *estado de creencia* se actualiza vía inferencia estilo HMM, y se toman decisiones para maximizar utilidad o lograr objetivos. Resolver POMDPs de forma exacta es más complejo que la inferencia en HMM, pero conceptualmente muestran cómo el HMM encaja en la panorámica más amplia de la toma de decisiones en IA. Muchos sistemas

prácticos aproximan las soluciones de POMDP haciendo estimación de estado con HMM + decisiones heurísticas, porque resolver POMDP de manera exacta puede ser costoso.

En resumen, los HMMs funcionan como el **"cerebro" de la percepción** en agentes de IA que manejan incertidumbre. Ya sea que el sistema sea reflejo y solo use la salida del HMM como entrada, o un agente de planificación completo que elige acciones, la capacidad de inferir el estado oculto a partir de observaciones es esencial. Los HMMs aportan una forma probabilística y eficiente de realizar esta inferencia. Transforman secuencias de datos de sensores en estimaciones de estado que la lógica de decisión del agente puede usar.

(Para visualizar, imagina un diagrama donde un agente recibe observaciones del entorno, las pasa a un módulo de actualización de creencia HMM (inferencia bayesiana), y obtiene un estado de creencia actualizado que se usa en el módulo de decisión. El agente actúa, afecta al entorno y el ciclo se repite — básicamente la arquitectura de un POMDP).

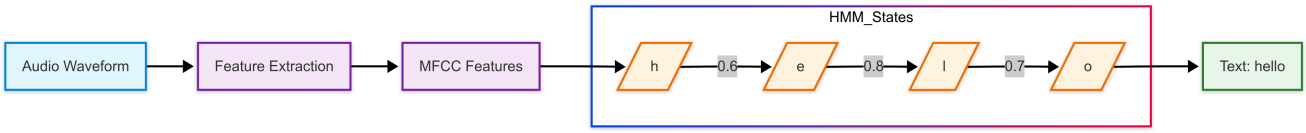


Figura 4: Aplicación de HMMs en reconocimiento de voz. El diagrama muestra cómo las formas de onda de audio se convierten en vectores de características (observaciones), que se usan en el HMM para inferir la secuencia más probable de fonemas o palabras (estados ocultos).

Fuente: Speech Recognition - Wikipedia

5. Conclusión y pasos siguientes

Hemos presentado los Modelos Ocultos de Markov como una poderosa extensión de las cadenas de Markov para modelar dinámica oculta en datos secuenciales. Un HMM nos permite **mantener una creencia sobre estados ocultos** y actualizarla conforme llegan observaciones, algo invaluable para sistemas de IA que trabajan con información parcial. Vimos cómo se definen los HMMs por probabilidades de transición y emisión, y cómo generan observaciones a partir de secuencias de estados ocultos. Mediante algoritmos como Forward, Viterbi y Baum-Welch, la IA puede hacer inferencia probabilística — calcular la probabilidad de una secuencia de observaciones, inferir el estado más probable, e incluso *aprender* el modelo a partir de la experiencia.

PROF

En términos prácticos, los HMMs conectan las *cadenas de Markov puras* (con estados completamente visibles) con escenarios de la vida real impulsados por sensores, donde no vemos directamente el estado del sistema. Permiten la **inferencia probabilística** de forma rigurosa, un tema central en IA. Muchas aplicaciones mencionadas — reconocimiento de voz, predicción de intenciones del usuario, modelado de oponentes, detección de tendencias — ilustran cómo los HMMs llevan las cadenas de Markov al mundo donde la observabilidad es incompleta. Al convertir los datos de observación en estimaciones probabilísticas de estado, los HMMs permiten a los agentes tomar decisiones mejor informadas en lugar de adivinar a ciegas.

Si seguimos avanzando, comprender los HMMs nos prepara para temas más complejos. Uno de ellos es el **MDP (Markov Decision Process)**, donde introducimos *acciones* y *recompensas* en el marco de Markov. En un MDP, un agente elige acciones que causan transiciones de estado (con ciertas probabilidades) y buscamos políticas óptimas (acciones en cada estado) para maximizar la recompensa. Si combinamos la

idea de MDP con la incertidumbre del estado — un agente que no observa el estado con certeza — obtenemos los **POMDP (Partially Observable MDP)** (). En un POMDP, el agente no conoce el estado con certeza, así que lleva consigo una creencia (similar al filtrado de un HMM) y elige acciones basadas en esa creencia. Resolver POMDPs de forma óptima es complejo, pero muchos problemas de robótica e IA se formulan así. En esencia, los HMMs (que manejan la incertidumbre sobre el estado) son el componente de percepción de un POMDP, mientras que los algoritmos de MDP manejan la parte de decisión.

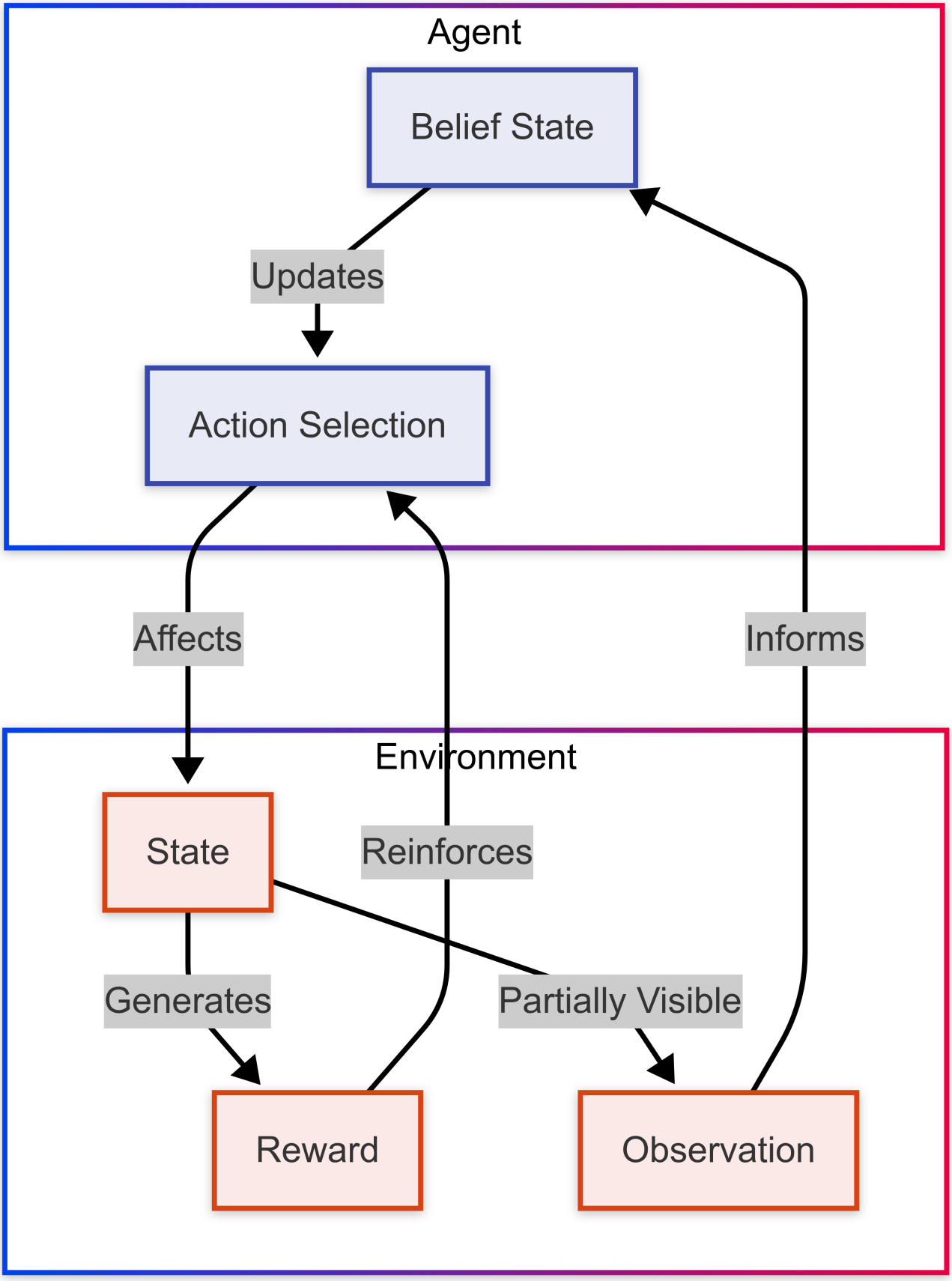
Antes de sumergirnos en MDPs y POMDPs, vale la pena subrayar la relevancia de los HMMs: son **una herramienta fundamental para el razonamiento probabilístico temporal**. Incluso en la era del deep learning, los conceptos de los HMMs (estado, observación, transición, emisión, actualización de creencia) aparecen en modelos más complejos como las redes bayesianas dinámicas.

Para cerrar, consideremos un ejemplo creativo que lo vincule todo: **reconocimiento de gestos** en un dispositivo inteligente. Supongamos que queremos que una IA entienda gestos del usuario (como deslizar, pellizcar, ampliar) en una pantalla táctil. Podemos diseñar un HMM donde los estados ocultos correspondan a las fases del gesto (“inicio”, “mitad”, “fin” del desplazamiento, o distintos comandos), y las observaciones sean los eventos de toque (posiciones de los dedos, movimientos). A medida que el usuario mueve los dedos, el HMM actualiza su creencia de qué gesto está ocurriendo. Al final del movimiento, el dispositivo puede decodificar el gesto más probable (usando Viterbi). Si lo desplegamos en una app real, con el tiempo puede *aprender* ajustes (si el usuario tiene tendencia a un “long-press” antes de deslizar — el HMM puede añadir un estado extra). Así se ve cómo un HMM puede integrarse en un sistema de IA interactivo, interpretando datos ruidosos para producir un entendimiento útil que guíe la respuesta del sistema.

Te toca a ti – ¡Piensa en ejemplos! Imagina otro uso de un HMM en un escenario real de IA. Podría ser cualquier situación con secuencias de observaciones y un contexto oculto. Por ejemplo, ¿cómo se usaría un HMM en **monitorización de la salud** (estado oculto = condición del paciente, observaciones = lecturas de un wearable)? ¿O en **finanzas** (estado oculto = régimen del mercado, observaciones = movimientos de precio, ver Hidden Markov Models - An Introduction | QuantStart)? Proponer ejemplos es una buena práctica para entrenar la identificación de la parte oculta y la parte observable que forman la esencia de un HMM.

PROF

En conclusión, los Modelos Ocultos de Markov nos ofrecen un método fundamentado para **"ver lo invisible"** — reducir la incertidumbre acerca de estados que no podemos observar directamente, usando probabilidades para cuantificar nuestras creencias y actualizarlas conforme recibimos nueva evidencia. Al pasar a la toma de decisiones y planificación bajo incertidumbre, mantén la intuición de que un agente no necesita certeza absoluta para actuar; puede llevar en la mente una distribución de probabilidad y, aun así, tomar decisiones inteligentes. Los HMMs son el primer paso en ese camino de la *IA probabilística*, que nos conduce a MDPs, POMDPs y más allá.



PROF

Figura 5: Diagrama de un Proceso de Decisión de Markov (MDP) que muestra la interacción entre un agente de IA y su entorno. El agente toma acciones basadas en el estado actual, y el entorno responde con un nuevo estado y recompensa. En un POMDP, el agente no ve el estado verdadero directamente, sino que solo recibe observaciones — combinando así un MDP con un HMM.

Fuente: Partially Observable Markov Decision Process - Wikipedia

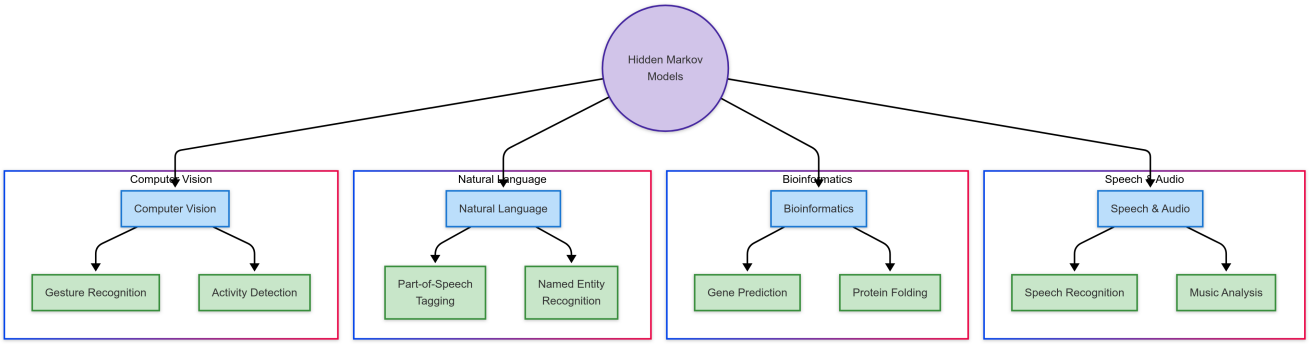


Figura 6: Visión general de aplicaciones de HMM en diversos campos. Desde el reconocimiento de voz hasta la bioinformática, los HMMs ayudan a descubrir patrones ocultos en datos secuenciales al modelar la relación entre estados ocultos y salidas observables.

Fuente: Hidden Markov Model - Wikipedia

Figura: Las sesiones de planificación estratégica del consejo estudiantil — ilustran cómo los agentes deben tomar decisiones basadas en observaciones parciales e inferencias sobre estados ocultos.

Fuente: Kaguya-sama: Love is War - Tenor