Ciencias Experimentales

Universidad Rey Juan Carlos | Servicio de Publicaciones

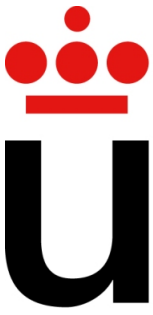
Joaquín Arias
Luciana Fidilio-Allende

LIBRO: Inteligencia Artificial, por y para la Ciberseguridad

Material docente en abierto de la Universidad Rey Juan Carlos

Agradecimientos

Este libro reúne, en un único volumen, el conjunto de materiales que articulan la asignatura de Inteligencia Artificial, por y para la Ciberseguridad. En él se integran la guía académica, los apuntes desarrollados a lo largo del curso, las transparencias utilizadas en clase y una selección de ejercicios diseñados para afianzar y aplicar los conocimientos adquiridos (y material adicional para seguir aprendiendo). Cómo citar esta obra: Arias, Joaquín y Fidilio-Allende, Luciana (2026). *LIBRO: Inteligencia Artificial, por y para la ciberseguridad*. BURJC, Madrid.



Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA INFORMÁTICA

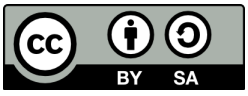
GRADO EN INGENIERÍA DE LA CIBERSEGURIDAD

Guia de la Asignatura

INTELIGENCIA ARTIFICIAL

2025-2026

Autor: Joaquín Arias



Copyright (c) 2026 Joaquín Arias. Esta obra está bajo la licencia CC BY-SA 4.0, Creative Commons Atribuciónn-CompartirIgual 4.0 Internacional.

1. Introducción de la Asignatura

El objetivo de la asignatura es introducir al alumno en las tecnologías basadas en Inteligencia Artificial enfocado en el concepto de agente inteligente.

Hoy en día las tecnologías asociadas a la Inteligencia Artificial, se utilizan en todos los campos de la ciencia. La asignatura está estructurada en tres grandes bloques:

- Módulo II. "Resolución de problemas búsqueda"
- Módulo III. "Representación del conocimiento"
- Módulo IV. "Aprendizaje automático"

Los conocimientos recibidos le permitirán al alumno:

- Disponer de una idea global del concepto e historia de la Inteligencia Artificial y sus aplicaciones.
- Buscar la solución a un problema de seguridad entre un conjunto de posibles opciones discretas que interaccionan.
- Modelar sistemas de conocimiento a cerca de sistemas de seguridad, sobre los que podra realizar consultas.
- Construir sistemas que solucionan los problemas que se le planteen de forma automática a partir de los datos que se les proporcione.

Como requisito indispensable para seguir la asignatura de Inteligencia Artificial es necesario haber cursado la asignatura de Lógica con especial atención a la lógica de primer orden y el método de resolución de Robinson con unificación. Para ello se recomienda a los alumnos repasar la asignatura en abierto de Lógica disponible en BURJC, handle http://hdl.handle.net/10115/20014: Arias, Joaquín (2022). Lógica: desde Aristóteles hasta Prolog. Madrid: Servicio de Publicaciones de la Universidad Rey Juan Carlos. ISBN:978-84-09-38265-1

2. Temario de la asignatura

Módulo I: Introducción a la Inteligencia Artificial.

Tema 1: Introducción a la Inteligencia Artificial.

Módulo II: Resolución de problemas: búsqueda.

Tema 2: Búsqueda no informada.

Tema 3: Búsqueda heurística.

Tema 4: Búsqueda multiagente.

Módulo III: Representación del conocimiento.

Tema 5: Representación probabilística del conocimiento.

Tema 6: Redes Bayesianas.

Tema 7: Inferencia en Redes Bayesianas.

Tema 8: Teoría de la decisión.

Módulo IV: Aprendizaje automático.

Tema 9: Aprendizaje probabilístico.

Tema 10: Aprendizaje inductivo.

Tema 11: Aprendizaje por refuerzo.

3. Desarrollo de la asignatura en 15 semanas

La asignatura la consideramos agrupada en cuatro módulos:

- El módulo I contiene el tema 1. Corresponde a introducción de la asignatura, Definición y objetivos de la IA, Historia y evolución de la Inteligencia Artificial, Áreas principales de la IA por y para la Ciberseguridad y Agentes inteligentes: tipos y entornos.
- El módulo II contiene los temas 2, 3 y 4. Corresponde a la parte de búsqueda no informada e informada (con heurísticas) con algunos ejemplos de búsqueda multiagente.
- El módulo III contiene los temas 5, 6, 7 y 8. Se presentan aspectos de la representación del conocimiento con presencia de incertidumbre, enfocado en el razonamiento probabilístico, las redes Bayesianas (modelado e inferencia) y una pequeña introducción a la teoría de la decisión.
- Finalmente, el módulo IV está formado por los temas 9, 10 y 11 y se centra en el aprendizaje automático partiendo de modelos básicos como Naive Bayes, los árboles de decisión e introduciendo los fundamentos del aprendizaje por refuerzo, en particular se explica el Q-Learning.

Considerando 15 semanas de clase, con una sesiones de dos horas cada una, proponemos la siguiente distribución:

Semana 1

Parte de la sesión, en la docencia presencial, se dedica a presentar la asignatura, leer la guía docente, explicar el sistema de evaluación y comentar como se desarrollarán las prácticas evaluables. El resto del primer día se dedica al Tema 1.

Para seguir el curso de manera autónoma recomendamos dedicar esta semana al tema 1 y a conocer la documentación disponible –dedicando especial atención a la lectura de

esta guía de la asignatura.

Semanas 2 a 4

La semana 2 se dedicará al Tema 2 (los alumnos que no conozcan los conceptos presentados en este tema, método de resolución de Robinson, tendrán que dedicar tiempo extra a revisar la asignatura en abierto de Lógica de la URJC (https://www.aulavirtual.urjc.es/moodle/course/view.php?id=221276).

Las semanas 3 a 4 se dedicará a completar el temario del módulo II.

Semana 5

Esta semana, en la modalidad presencial, se dedica a resolución de problemas en clase con especial dedicación a la preparación de la práctica evaluable que se califica mediante actividad autónoma en el aula virtual.

A los alumnos que cursan de manera autónoma se les recomienda realizar los ejercicios propuestos en los apuntes, transparencias y muy especialmente en el compendio de ejercicios (donde hay una selección de ejercicios más complejos con sus soluciones).

Estos ejercicios son similares a los ejercicios de exámenes y por lo tanto omitimos la publicación de exámenes en abierto (y recomendamos que adicionalmente los alumnos busquen ejercicios similares en internet, para afianzar los conocimientos adquiridos).

Semanas 6 a 9

Estas semanas se dedicarán al módulo III siguiendo la misma metodología que en los módulos anteriores.

Semanas 10

La sesión de esta semana se dedica a la práctica y resolución de problemas del módulo III, de manera similar a la sesión realizada en la semana 5 para el módulo II.

Semanas 11 a 13

Estas semanas se dedicarán al módulo IV siguiendo la misma metodología que en los módulos anteriores.

Semanas 14

La sesión de esta semana se dedica a la práctica y resolución de problemas del módulo IV, de manera similar a las sesiones realizadas en las semanas 5 y 10.

Semanas 15

El último día de clase, si se han podido impartir las 14 sesiones previas (teniendo en cuenta que en algunos años, por festividades o causas de fuerza mayor, no siempre es posible completarlas), y se ha completado el temario, se presenta un tutorial de Prolog y de s(CASP), un razonador no-monotono desarrollado por Joaquín Arias en colaboración con la University of Texas at Dallas e IMDEA Software Institute, para la resolución automáticas de problemas de razonamiento, en particular se resolverá el problema de las Garras estudiado en el módulo II.

4. Bibliografía

Como libros de referencia principales recomendamos:

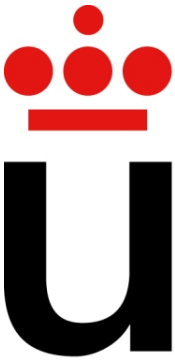
- S. J. Russell and P. Norvig (2021). Artificial Intelligence: A Modern Approach. 4th. Pearson. isbn: 978-1292401133.
- N. J. Nilsson (2001). Inteligencia Artificial: Nueva S´ıntesis. McGraw-Hill. isbn: 978-8448128241.

Otra bibliografía de consulta recomendada es la siguiente:

- The Art of Prolog, L. Sterling and E. Shapiro, The MIT Press, Cambridge, Mass., second edition, 1994. (disponible online gratis).
- A. Fernández Gil (2010). Sistemas expertos: Representación e inferencia. Problemas resueltos. Dykinson.
- H. Billhardt, A. Fernández and S. Ossowski (2015). Inteligencia Artificial. Ejercicios resueltos. Editorial Universitaria Ramón Areces.

- K. Breitman, M. A. Casanova and W. Truszkowski (2007). Semantic Web: Concepts, Technologies and Applications. Springer.
- P. McCorduck (1979). Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence. First edition. A. K. Peters. isbn: 978-0716710721.
- — (1991). MEaquinas que piensan: Una incursión personal en la historia y perspectivas de la inteligencia artificial. Editorial Tecnos. isbn: 978-8430920167.

Todos estos títulos están disponibles (existen varias copias) en la biblioteca de la URJC pero independientemente recomendamos la consulta de cualquier referencia relacionada con esta asignatura y los recursos disponibles en internet (por ejemplo ejercicios resueltos, aunque con visión crítica porque pueden contener errores).



Universidad Rey Juan Carlos

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE LA CIBERSEGURIDAD

Apuntes de Inteligencia Artificial

POR Y PARA LA CIBERSEGURIDAD

Autor: Joaquín Arias



Copyright ©2026 Joaquín Arias .Esta obra está bajo la licencia CC BY-SA 4.0, Creative Commons Atribuciónn-CompartirIgual 4.0 Internacional. Cómo citar esta obra: Arias, Joaquín y Fidilio-Allende, Luciana (2026). *LIBRO: Inteligencia Artificial, por y para la ciberseguridad*. BURJC, Madrid.

Índice general

Índice general i

Módulo I Introducción a la Inteligencia Artificial 1

1 Introducción a la Inteligencia Artificial 2

1.1 Definición y objetivos de la IA 2

1.2 Historia y evolución de la Inteligencia Artificial 3

1.3 Áreas principales de la IA por y para la Ciberseguridad 6

1.3.1 IA y ciberseguridad: oportunidades y amenazas 6

1.4 Agentes inteligentes: tipos y entornos 7

1.4.1 Aplicaciones (básicas) de IA 7

1.4.2 Ejercicios para casa I 10

1.4.3 Ejercicios para casa II 10

Módulo II Resolución de problemas y búsqueda 11

2 Búsqueda no informada 12

2.1 Representación de problemas como espacios de estados 12

2.2 Resolución del problema 14

2.2.1 Búsqueda en profundidad 15

2.2.2 Búsqueda en anchura 16

2.2.3 Búsqueda de coste uniforme 17

2.3 Comparación de algoritmos: complejidad y completitud 18

3 Búsqueda heurística 20

3.1 Heurísticas: definición y propiedades 20

3.2 Búsqueda voraz 21

3.3 Algoritmo A* 22

3.4 Heurísticas admisibles y consistentes 24

4 Búsqueda multiagente 26

4.1 Entornos multiagente 26
4.2 Juegos competitivos y cooperativos 27
4.3 Algoritmo Minimax 27
4.4 Poda alfa-beta 29
4.5 Introducción a juegos estocásticos 31

Módulo III Representación del conocimiento 33

5 Representación probabilística del conocimiento 34
5.1 Incertidumbre y razonamiento probabilístico 34
5.2 Probabilidad condicional 34
5.3 Regla de Bayes 37
5.4 Independencia y dependencia probabilística 38

6 Redes Bayesianas 40
6.1 Redes Bayesianas 40
6.1.1 Definición y componentes 40
6.1.2 Grafos dirigidos acíclicos 41
6.1.3 Tablas de probabilidad condicional 41
6.2 Construcción de modelos bayesianos 42

7 Inferencia en Redes Bayesianas 45
7.1 Inferencia exacta 45
7.1.1 Algoritmo de Enumeración 46
7.1.2 Algoritmo de Eliminación de variables 47
7.1.3 'Implementacón' en Python 48
7.2 Inferencia aproximada 49
7.2.1 Métodos principales 49
7.2.2 Algoritmos de propagación de creencias 50

8 Teoría de la decisión 52
8.1 Fundamentos de la teoría de la decisión 52
8.2 Utilidad 53
8.2.1 Funciones de utilidad 53
8.2.2 Utilidad Esperada 53
8.3 Modelos de decisión 53
8.3.1 Redes de decisión 53
8.3.2 Árboles de Decisión 55
8.3.3 Secuencias y procesos de Markov 55

Módulo IV Aprendizaje automático **58**

9 Aprendizaje probabilístico **59**

9.1 Aprendizaje automático 59
9.1.1 Introducción 59
9.1.2 Modelos generativos y discriminativos 60
9.1.3 Clasificación probabilística 60

9.2 Naive Bayes 61
9.2.1 Algoritmo 61
9.2.2 'Implementación' en Python 62

9.3 Evaluación de modelos 63

10 Aprendizaje inductivo **67**

10.1 Fundamentos del aprendizaje inductivo 67

10.2 Árboles de decisión 68
10.2.1 Fundamentos 68
10.2.2 Algoritmo ID3 68

10.3 Sobreajuste y generalización 72

10.4 Validación cruzada 72

11 Aprendizaje por refuerzo **73**

11.1 Fundamentos 73
11.1.1 Definición y formalización básica 73
11.1.2 Interacción agente-entorno 74
11.1.3 Motivación, aprendizaje sin modelo 74

11.2 Resolución basada en modelo 75
11.2.1 El entorno como MDP 75
11.2.2 Algoritmo de iteración de valores 75

11.3 Aprendizaje sin modelo 77
11.3.1 Fundamentos y Q-learning 77
11.3.2 Q-learning para entornos deterministas 78
11.3.3 Q-learning para entornos no deterministas 80

Bibliografía **83**

Agradecimientos

Esta obra, contiene en formato de apuntes las transparencias de Arias, Joaquín (2026). Inteligencia Artificial, por y para la Ciberseguridad. BURJC, Madrid, las cuales están basada en transparencias de Sascha, Holger y Alberto (URJC'23) & de Carmen y Paco (URJC'21).

Módulo I

Introducción a la Inteligencia Artificial

Universidad Rey Juan Carlos

Tema 1

Introducción a la Inteligencia Artificial

1.1 Definición y objetivos de la IA **2**

1.2 Historia y evolución de la Inteligencia Artificial **3**

1.3 Áreas principales de la IA por y para la Ciberseguridad **6**

1.3.1 IA y ciberseguridad: oportunidades y amenazas 6

1.4 Agentes inteligentes: tipos y entornos **7**

1.4.1 Aplicaciones (básicas) de IA 7

1.4.2 Ejercicios para casa I 10

1.4.3 Ejercicios para casa II 10

1.1. Definición y objetivos de la IA

Definición: La Inteligencia Artificial (IA) (McCarthy et al. 1955) es la disciplina que estudia y desarrolla sistemas capaces de realizar tareas que, tradicionalmente, requieren inteligencia humana, como razonar, aprender, tomar decisiones, percibir el entorno y adaptarse a él.

Desde la perspectiva de la ciberseguridad, la IA se concibe como:

- una herramienta para automatizar, mejorar y reforzar la detección, prevención y respuesta ante amenazas digitales.
- una amenaza capaz de adaptar ataques de forma autónoma, optimizar la evasión y escalar su impacto, aumentando la sofisticación y rapidez de las amenazas digitales.
- **Ejercicio:** Indica otras áreas o enfoques de la IA desde la perspectiva de la ciberseguridad.

Clasificación sistemas autónomos (IA Fuerte vs. IA Débil):

- Sistemas que piensan como humanos

  "La interesante tarea de lograr que las computadoras piensen... Máquinas con mente, en su amplio sentido literal" (Haugeland 1985)
- Sistemas que actúan como humanos

  "El arte de crear máquinas con capacidad de realizar funciones que realizadas por personas requieren inteligencia" (Kurzweil 1990)
- Sistemas que actúan de forma racional

  "La rama de la Informática que se ocupa de la automatización del comportamiento inteligente" (Luger y Stubblefield 1993)

Objetivos principales de la IA:

- Actuar como humanos:
  - Prueba de Turing: (Turing 1950)
- Pensar como humanos:
  - Modelado cognitivo: crear teoría acerca de la mente humana.
  - General Problem Solver (GPS): (Newell y Simon 1961)
- Actuar de forma racional:
  - Razonamiento irrefutable, inferencia lógica, programación lógica... (Aristotle -350; Arias 2023; Arias 2024)
  - Agentes racionales: actuar de forma "correcta/óptima/eficiente" en un entorno (con respecto a unos valores). (Arias 2024)
- (Dejar de) alucinar: (Weise y Metz 2023)

1.2. Historia y evolución de la Inteligencia Artificial

La evolución de la IA está afectada por avances en distintas disciplinas:

- Filosofía (lógica, dualismo, materialismo, empirismo, positivismo, utilitarismo).
- Matemáticas (algoritmo, teorema de incompletitud, P vs. NP, teoría de la decisión).
- Psicología (comportamiento humano, conductistas, cognitiva).
- Lingüística, Sociología, Economía, Neurociencia, etc...

Orígenes (años 50–60)

- Término "Inteligencia Artificial" acuñado por McCarthy en 1956 (McCarthy et al. 1955).
- Primeros sistemas basados en lógica y reglas.
- Primeros modelos de neuronas artificiales (McCulloch/Pitts).
- Optimismo inicial sobre la rápida creación de máquinas inteligentes.

(McCorduck 1979; McCorduck 1991)

Sistemas expertos (años 60–75)

- Programas que replicaban el conocimiento de expertos humanos.
- Representación especializada del conocimiento:
  - Uso intensivo de reglas "si–entonces".
  - Reglas, marcos, guiones.
- Primeros sistemas expertos (Dendral, Prospector, Mycin).

Invierno de la IA (años 75-80)

- Motivada por el informe Lighthill en 1973 (Lighthill 1973).
- Declive de la computación neuronal (Minsky y Papert 1969).
- Limitaciones en escalabilidad y mantenimiento.

Proyecto FGCS (años 80-90)

- Proyecto de software de "quinta generación", Fifth Generation Computer Systems (FGCS) (Moto-Oka 2012).
  - Procesamiento en paralelo de gran cantidad de información.
  - Uso de lenguajes de programación lógica, (concurrent) Prolog, a niveles muy integrados con el hardware

...vs. enfoques basados en arquitectura Von Neumann.
- No logró el éxito comercial esperado y se "abandona" la IA simbólica.

Aprendizaje automático (años 90-2000)

- Enfoque en datos y estadística, regreso de las redes de neuronas:
  - Algoritmo de propagación hacía atrás. (Rumelhart, Hinton y Williams 1986)
- Modelos rigorosos y computacionalmente eficientes de manejo de incertidumbre (cadenas de Markov, redes Bayesianas).

IA moderna y aprendizaje profundo (2010–actualidad)

- Redes neuronales profundas (Deep Learning), múltiples capas y niveles de

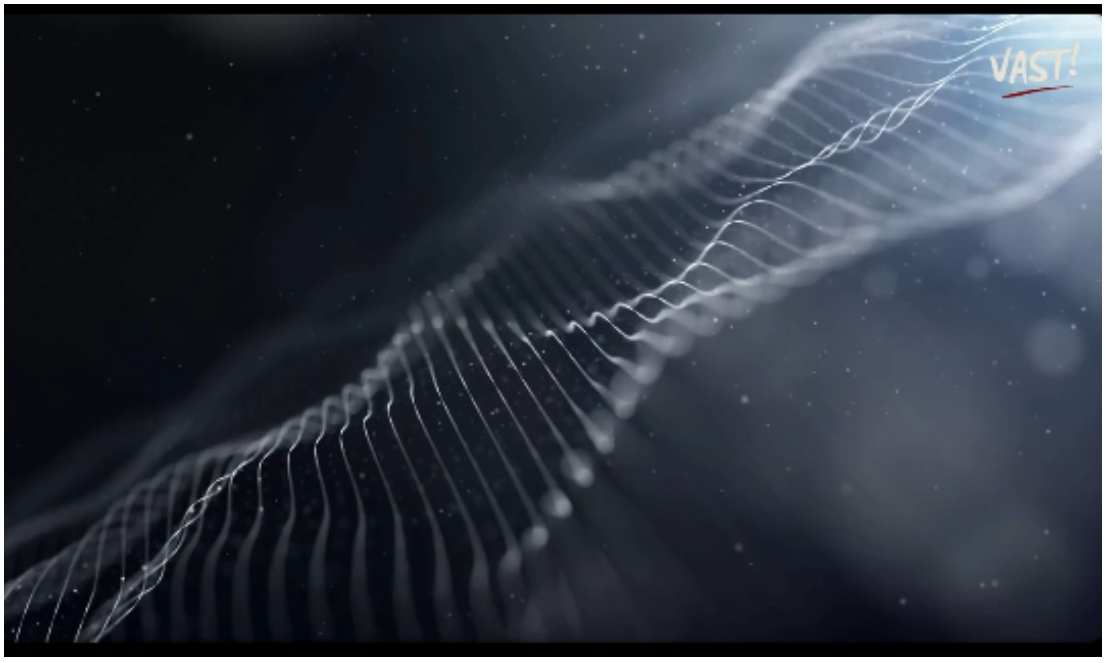
aprendizaje.

- Avances en visión artificial, procesamiento del lenguaje natural, ciberseguridad, sistemas autónomos / multiagente.
- Dependencia de grandes volúmenes de datos (Big Data) y potencia de cálculo para entrenamiento y/o inferencia (GPU's).
- Conceptos Trending Topic:
  - Modelos pre-entrenados (fine-tuning).
  - Aprendizaje reforzado (Reinforcement Learning, RL).
  - Computación distribuida (TensorFlow, PyTorch, JAX).
  - IA confiable (ética, seguridad, sesgos, explicabilidad)
  - Modelos multimodales (LLMs).
  - ...

Nombre propios de la IA (hoy):

- Judea Pearl (7/10).
- Yann Lecun (4/10).
- Geoffrey Hinton (3/10).
- Demis Hassabis (1/10).

Selección extraída de (VaST 2025)...



Link

1.3. Áreas principales de la IA por y para la Ciberseguridad

- Aprendizaje automático (Machine Learning)
  Clasificación de malware, detección de intrusiones, análisis de tráfico.
- Aprendizaje profundo (Deep Learning)
  Detección avanzada de amenazas, análisis de comportamiento.
- Procesamiento del Lenguaje Natural (PLN)
  Análisis de correos de phishing, ingeniería social automatizada.
- Visión artificial
  Seguridad física, reconocimiento facial (con implicaciones éticas).
- Sistemas multiagente
  Simulación de ataques, defensa distribuida.

1.3.1. IA y ciberseguridad: oportunidades y amenazas

Oportunidades:

- Detección temprana de intrusiones.
- Análisis automático de vulnerabilidades.
- Respuesta rápida a incidentes.
- Reducción de falsos positivos.
- Defensa adaptativa frente a ataques cambiantes.

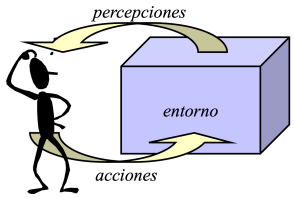
Amenazas:

- Uso de IA para generar malware avanzado.
- Ataques automatizados y personalizados.
- Deepfakes y desinformación.
- Ataques adversarios contra modelos de IA.
- Dependencia excesiva de sistemas opacos (black-box).

La IA es, por tanto, una herramienta dual:
puede fortalecer o debilitar la seguridad según su uso.

1.4. Agentes inteligentes: tipos y entornos

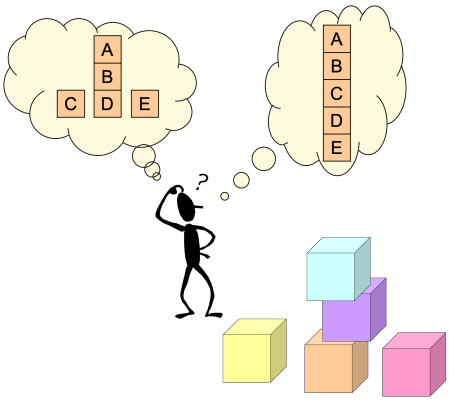
- Un agente inteligente es una entidad que:
  - Percibe su entorno mediante sensores.
  - Actúa sobre él mediante actuadores.
  - Toma decisiones para maximizar un objetivo.
- Entorno (en el que esta embebido):
  - Accesible / inaccesible.
  - Deterministas / estocástico.
  - Estáticos / dinámicos.
  - Discreto / contin

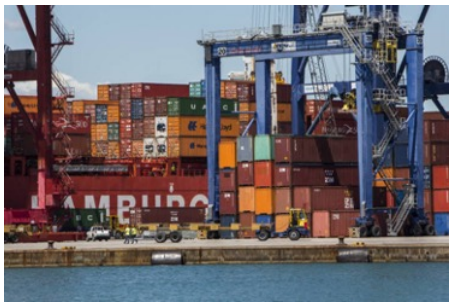


- Tipos de agentes:
  - Agentes reactivos simples: responden a estímulos inmediatos.
  - Agentes basados en modelos: mantienen información del estado interno.
  - Agentes basados en objetivos: toman decisiones para alcanzar metas.
  - Agentes basados en utilidad: optimizan resultados según criterios.
  - Agentes con aprendizaje: mejoran su comportamiento con la experiencia.

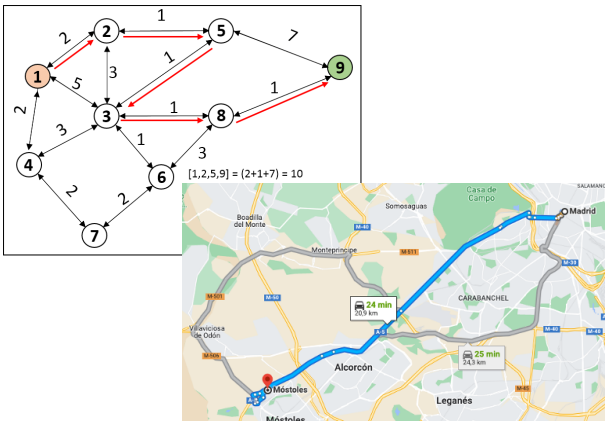
1.4.1. Aplicaciones (básicas) de IA

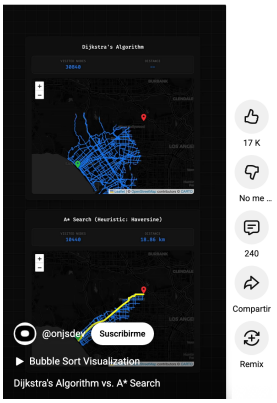
Problemas de planificación: Ordenar bloques





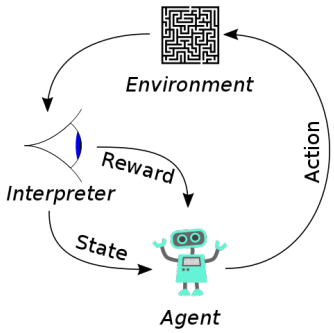
Problemas de optimización: Seleccionar camino más corto

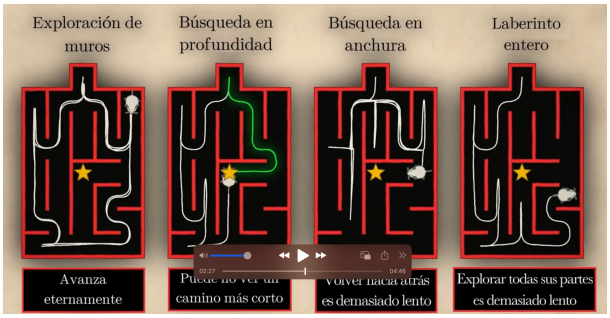




Link

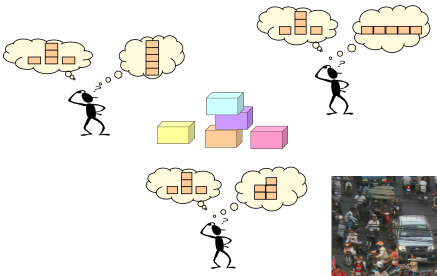
Técnicas para problemas complejos: Aprendizaje por refuerzo



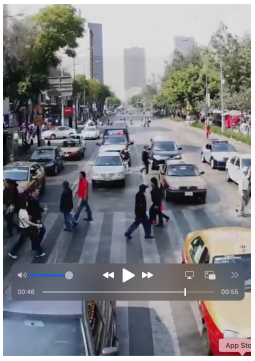


Link

Sistemas multiagente I: Distintos objetivos





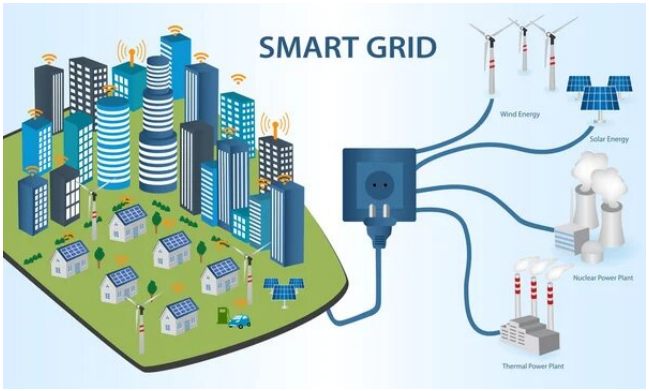


Link

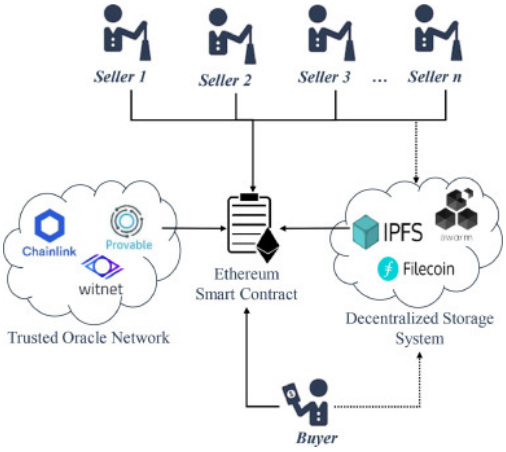
Sistemas multiagente II: Entorno competitivo (Óptimo local)



Sistemas multiagente III: Entorno cooperativo (Óptimo global)

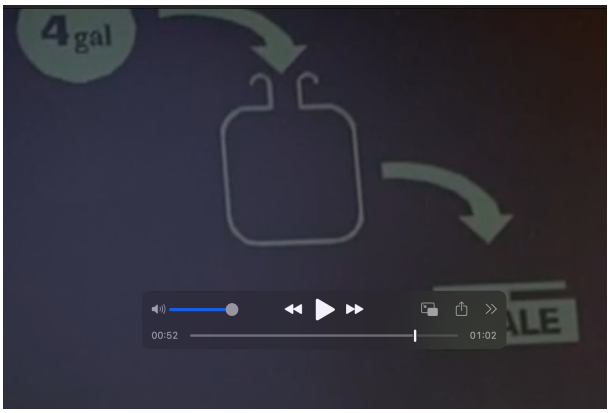


Sistemas multiagente IV: Estrategias mixtas (Subastas)



1.4.2. Ejercicios para casa I

- Escenario: Tenemos dos garrafas, una de 5 galones y otra de 3.
- Objetivo: Depositar 4 galones exactos para que la bomba no explote.



Link

1.4.3. Ejercicios para casa II

- Escenario: Una familia tiene que cruzar un puente frágil y sin iluminar. La hija tarda 1 min, el padre 2 min, la madre embarazada 5 min y el abuelo 8 min.
- Objetivo: Que todos crucen (max 2 por viaje y usando la linterna), antes de que se caiga (en 15 min).



Link

Módulo II

Resolución de problemas y búsqueda

Universidad Rey Juan Carlos

Tema 2

Búsqueda no informada

2.1 Representación de problemas como espacios de estados **12**

2.2 Resolución del problema **14**

2.2.1 Búsqueda en profundidad 15

2.2.2 Búsqueda en anchura 16

2.2.3 Búsqueda de coste uniforme 17

2.3 Comparación de algoritmos: complejidad y completitud **18**

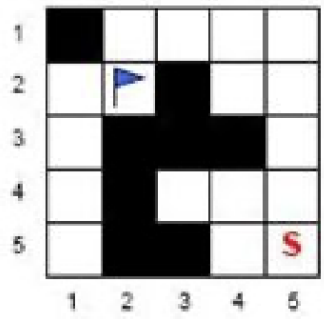
2.1. Representación de problemas como espacios de estados

- Un algoritmo de búsqueda es un conjunto de instrucciones.[1]
- La búsqueda en IA es un método universal de resolución de problemas:
  - Encontrar un elemento *e* en una lista.
  - Determinar el camino más corto en un grafo.
  - Planificar la secuencia de tareas a realizar en un proyecto.
- Dado un espacio de estados, busca las transiciones que llevan de un estado inicial a un estado objetivo, p.ej.:
  - Siguiente movimiento en el juego del tres en raya.
- Según la información del entorno disponible tenemos:
  - Algoritmos no informados o ciegos.
  - Algoritmos informados.
- EJEMPLO: búsqueda de un elemento en una lista tenemos:

[1] Consultar (Russell y Norvig 2021:sección 6) y (Nilsson 2001:capítulos 8 y 9).

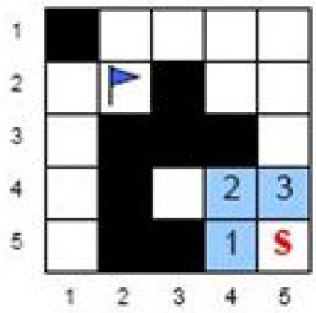
  - Búsqueda secuencial, no sabemos si la lista está ordenada.
  - Búsqueda binaria, sabemos que la lista está ordenada.
- En general los algoritmos informados son más eficientes:
  - Algoritmos heurísticos: (i) Voraz o Greedy y (ii) A*.
  - Algoritmos de mejora iterativa: (i) Escalada Simple y (ii) Escalada por Máxima Pendiente.
  - Algoritmos con adversario: (i) Minimax y (ii) Poda alfa-beta.

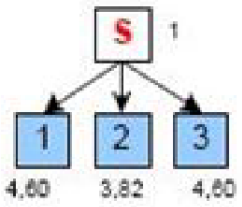
- Sparki quiere llegar a la bandera azul: las celdas de color negro son obstáculos y las celdas blancas son caminos posibles:



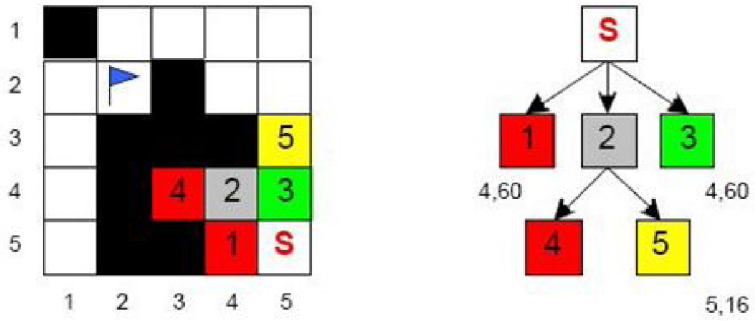


- La manera de representar el espacio de estados es mediante árboles.



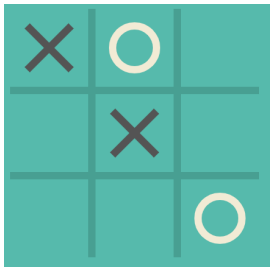


- Según exploramos el espacio algunos movimientos no permiten continuar (celdas marcadas en rojo).



Ejercicio 1 Diferencia entre un algoritmo ciego y un algoritmo informado.

Ejercicio 2 Dado el siguiente tablero del tres-en-raya:



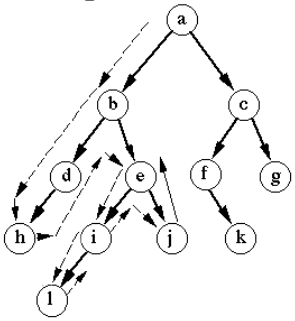
a) Dibujar el espacio de estados a partir de dicho tablero (juega la X).

b) Representar este tablero concreto usando dos tipos de dato algebraicos diferentes.

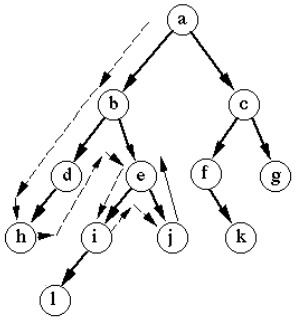
2.2. Resolución del problema

- Dada la representación del espacio de estados del problema:
  - Hay que encontrar un "camino" al estado objetivo.
  - Existen varias estrategias de búsqueda. P.ej.:

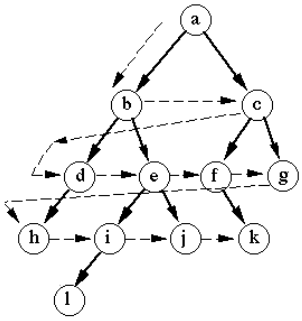
En profundidad



Profundización iterativa



En Anchura



- El espacio de estados modeliza el mundo mediante un grafo:
  - Cada estado posible es un nodo.
  - Los estados inicial, actual, objetivo son nodos específicos...
  - Una acción para pasar de un estado a otro es un arco entre nodos.
  - Una secuencia de acciones es un camino desde el nodo inicial al nodo objetivo.
- Hay problemas que se representan como grafos dirigidos (ver Fig. 2.1)

...otros como grafos no dirigidos (ver Fig. 2.2).

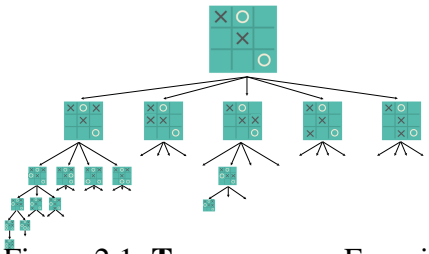


Figura 2.1: **Tres-en-raya:** Espacio de estados y posibles partidas desde `[X,O,_,_,X,_,_,_,O]`.

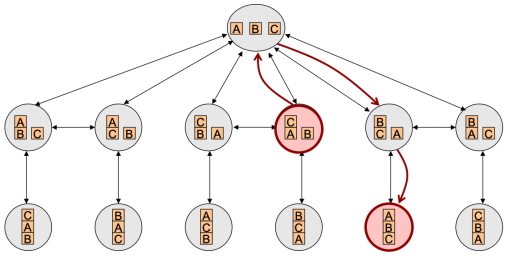


Figura 2.2: **3-bloques:** Espacio de estados y secuencia mínima para ir de `[[A,C],[B]]` a `[[C,B,A]]`.

2.2.1. Búsqueda en profundidad

```
ALGORITMO En_Profundidad_v0(E_Inicial, E_Objetivo)
   Q ← [E_Inicial]

   MIENTRAS Q ≠ [] HACER
      [E_Actual | Tail] ← Q
      Q ← Tail

      SI E_Actual == E_Objetivo ENTONCES
         DEVUELVE E_Actual

      PARA CADA Vecino EN vecinos(E_Actual) HACER
         Q ← [Vecino | Q]
   DEVUELVE Error
```

- Pseudocódigo para búsqueda en profundidad, versión 0:
  - Añade los nodos en la cabeza de la lista `Q`:

`[]`: lista vacía.

`[A|As]`: `A` es la cabeza (un nodo).
`As` es la cola (una lista).

- `vecinos(A)` son los nodos alcanzables desde `A`.

Ejercicio 3 Explica porque al aplicar `En_Profundidad_v0` podemos resolver el tres-en-raya de la Fig. 2.1

...pero entra en bucle con los 3-bloques de la Fig. 2.2.

```
ALGORITMO En_Profundidad_v1(E_Inicial, E_Objetivo)
   Q ← [E_Inicial]
   Visitados ← [E_Inicial]

   MIENTRAS Q ≠ [] HACER
      [E_Actual | Tail] ← Q
      Q ← Tail

      SI E_Actual == E_Objetivo ENTONCES
         DEVUELVE E_Actual

      PARA CADA Vecino EN vecinos(E_Actual) HACER
         SI Vecino ∉ Visitados ENTONCES
             Visitados ← [Vecino | Visitados]
             Q ← [Vecino | Q]
   DEVUELVE Error
```

- Pseudocódigo para búsqueda en profundidad, versión 1:
  - Evita expansión de nodos repetidos, recordando los nodos visitados.

`A`∉`As`: comprueba que el elemento `A` no está en la lista `As`.

Ejercicio 4 Adapta `En_Profundidad_v1` para que devuelva la secuencia de nodos para llegar al desde `E_Inicial` hasta `E_Objetivo`.

2.2.2. Búsqueda en anchura

```
ALGORITMO En_Anchura(E_Inicial, E_Objetivo)
   Q ← [E_Inicial]
   Visitados ← [E_Inicial]

   MIENTRAS Q ≠ [] HACER
      [E_Actual | Tail] ← Q
      Q ← Tail

      SI E_Actual == E_Objetivo ENTONCES
         DEVUELVE E_Actual
```

```
   PARA CADA Vecino EN vecinos(E_Actual) HACER
      SI Vecino ∉ Visitados ENTONCES
          Visitados ← [Vecino | Visitados]
          Q ← Q ++ [Vecino]
DEVUELVE Error
```

- Pseudocódigo para búsqueda en anchura:
  - Añade los nodos al final de la lista `Q`:

`A ++ B:` concatena las listas `A` y `B`.

  - No expande nodos visitados. No es necesario pero evita búsquedas redundantes.

Ejercicio 5 Aplica `En_Anchura` al 3-bloques (Fig. 2.2) y explica, con tus propias palabras, diferencias al aplicar `En_Profundidad_v1`.

2.2.3. Búsqueda de coste uniforme

```
ALGORITMO Coste_Uniforme(E_Inicial, E_Objetivo)
   Q ← [(E_Inicial,0)]

   MIENTRAS Q ≠ [] HACER
      [(E_Actual,C) | Tail] ← Q
      Q ← Tail

      SI E_Actual == E_Objetivo ENTONCES
         DEVUELVE E_Actual

      PARA CADA Vecino EN vecinos(E_Actual) HACER
         Q ← Q ++ [(Vecino,C+1)]
         Q ← sort(Q)
   DEVUELVE Error
```

- Pseudocódigo para búsqueda de coste uniforme (constante):
  - Considera coste `1` para cada acción:

`(A,C):` el nodo `A` tiene un coste asociado `C`.

  - Añade el nodo y el coste a `Q` y los ordena de manera ascendente según el coste.

Ejercicio 6 Aplica `Coste_Uniforme` al 3-bloques (Fig. 2.2) y explica si hace falta (o no) recordar algún(os) nodo(s) visitado(s).

```
ALGORITMO Coste_Uniforme_G(E_Inicial, E_Objetivo)
   Q ← [(E_Inicial,0)]

   MIENTRAS Q ≠ [] HACER
      [(E_Actual,C) | Tail] ← Q
      Q ← Tail

      SI E_Actual == E_Objetivo ENTONCES
         DEVUELVE E_Actual

      PARA CADA <Vecino,G> EN vecinos(E_Actual) HACER
         Q ← Q ++ [(Vecino,C+G)]
         Q ← sort(Q)
   DEVUELVE Error
```

- Pseudocódigo para búsqueda de coste uniforme:
  - Considera un coste `G` diferente para cada transición entre nodos:

`<A,G>:` el nodo `A` esta conectado con arista de peso `G`.

Ejercicio 7 Adapta los algoritmos `En_Profundidad_v1` y `En_Anchura` para que acumulen el coste de transiciones (peso de las aristas).

2.3. Comparación de algoritmos: complejidad y completitud

Ejercicio 8 Encontrar un camino desde Arad hasta Bucarest. Representar los distintos valores de `Q` (en al menos 10 iteraciones).

Usando los algoritmos:[2]

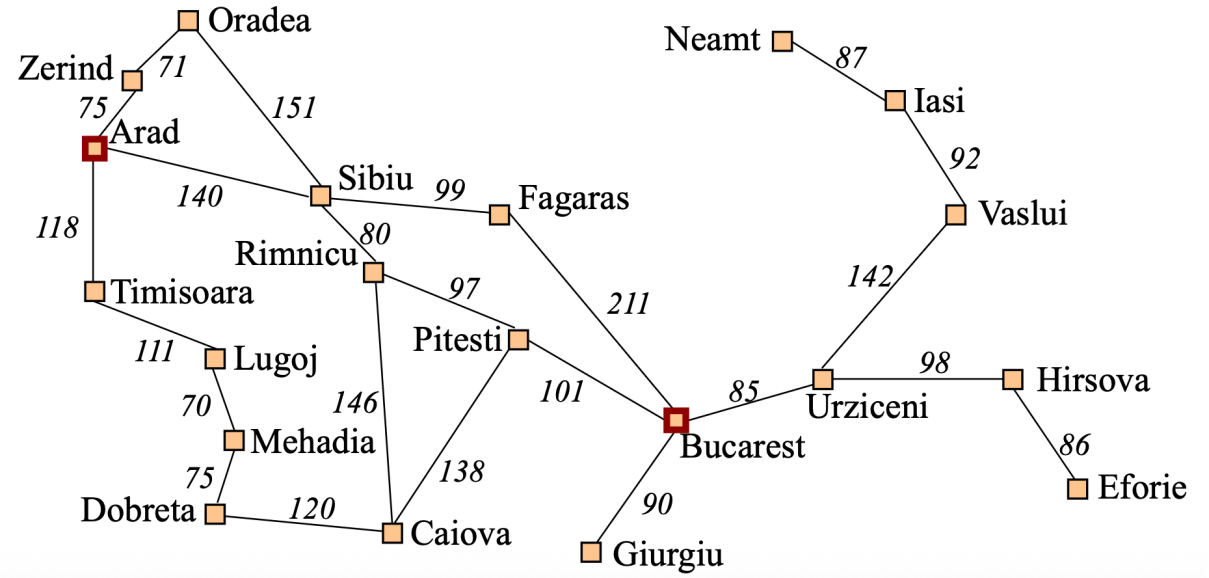
a) `En_Profundidad_v1`.

b) `En_Anchura`.

c) `Coste_Uniforme`.

d) `Coste_Uniforme_G`.

[2] Adaptados para acumular costes (**Ejercicio 7**).



Algoritmo	**Tiempo**	**Espacio**	**Completo**	**Óptimo**	**Comentarios**
En anchura	$O(b^d)$	$O(b^d)$	Sí	Sí (costes uniformes)	Explora por niveles; alto consumo de memoria.
En profundidad	$O(b^m)$	$O(bm)$	No (ciclos)	No	Bajo consumo de memoria; puede quedar atrapada.
Profundización iterativa	$O(b^d)$	$O(bd)$	Sí	Sí (costes uniformes)	Combina completitud de anchura con bajo consumo de memoria de profundidad.
Coste uniforme	$O(b^{\lceil C^*/\varepsilon \rceil})$	$O(b^{\lceil C^*/\varepsilon \rceil})$	Sí ($\varepsilon > 0$)	Sí	Generaliza anchura a costes no uniformes.

Notación: b: Factor de ramificación. d: Profundidad de la solución más superficial. m: Profundidad máxima del árbol de búsqueda. C^*: Coste de la solución óptima. ε: Coste mínimo de una acción.

Universidad Rey Juan Carlos

Tema 3

Búsqueda heurística

3.1 Heurísticas: definición y propiedades 20

3.2 Búsqueda voraz 21

3.3 Algoritmo A* 22

3.4 Heurísticas admisibles y consistentes 24

3.1. Heurísticas: definición y propiedades

Definición:

Una **heurística** es una función $h(n)$ que estima el coste mínimo necesario para alcanzar el estado objetivo desde un nodo n.

- Su objetivo principal es guiar el proceso de búsqueda hacia las regiones más prometedoras del espacio de estados, reduciendo el número de nodos explorados.
- Las heurísticas no garantizan por sí mismas una solución óptima, pero permiten mejorar significativamente la eficiencia de los algoritmos de búsqueda informada.

Propiedades de las heurísticas

- **Informatividad**: cuanto más cercana sea la estimación al coste real, más eficiente será la búsqueda.
- **Bajo coste computacional**: la heurística debe ser rápida de calcular.
- **No negatividad**: generalmente se cumple que $h(n) \geq 0$.
- **Calidad**: una heurística domina a otra si proporciona estimaciones más precisas sin sobreestimar el coste real.

3.2. Búsqueda voraz

Definición:

La **búsqueda voraz** (*Greedy Best-First Search*) es un algoritmo de búsqueda informada que selecciona en cada iteración el nodo que parece estar más cercano al objetivo según la heurística.

- La función de evaluación se define como:

$$f(n) = h^*(n)$$

- Este enfoque ignora el coste acumulado desde el nodo inicial y se basa únicamente en la estimación heurística.

Características

- No garantiza encontrar la solución óptima.
- No siempre es completa en espacios de búsqueda infinitos.
- Es rápida y sencilla de implementar.

```
ALGORITMO Voraz(E_Inicial, E_Objetivo, h*)
   Q ← [(E_Inicial, h*(E_Inicial) )]

   MIENTRAS Q ≠ [] HACER
      [(E_Actual, _ ) | Tail] ← Q
      Q ← Tail

      SI E_Actual == E_Objetivo ENTONCES
         DEVUELVE E_Actual

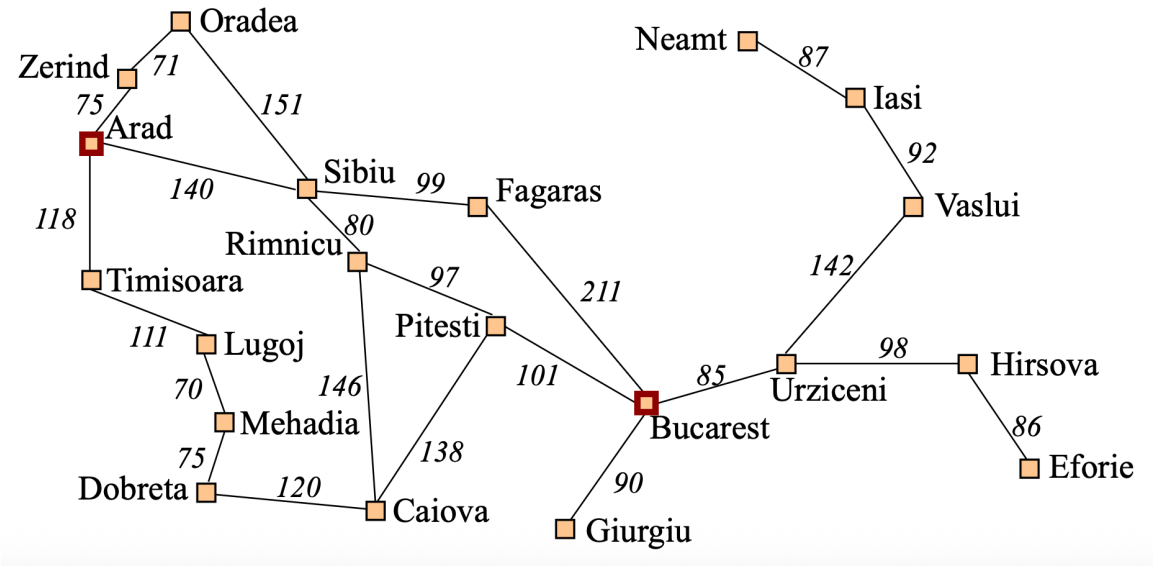
      PARA CADA Vecino EN vecinos(E_Actual) HACER
         Q ← Q ++ [(Vecino, h*(Vecino) )]
         Q ← sort(Q)
   DEVUELVE Error
```

- Pseudocódigo para búsqueda voraz:
  - Solo considera la estimación `h*(n)` en cada nodo:

`(A,h*(A))`: el nodo `A` tiene un coste estimado a destino de `h*(A)`.

  - Añade el nodo y el coste a `Q` y los ordena de manera ascendente según el coste.

Ejercicio 9 Encontrar un camino desde Arad hasta Bucarest usando el algoritmo `Voraz` y la función heurística $h^*(n)$ adjunta.



Ciudad	$h^*(n)$		
Arad	366	Mehadia	241
Bucarest	0	Neamt	234
Caiova	160	Oradea	380
Dobreta	242	Pitesti	98
Eforie	161	Rimnicu	193
Fagaras	178	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Tabla 3.1: $h^*(n)$: distancia desde n hasta Bucarest.

3.3. Algoritmo A*

Definición:

El algoritmo **A*** es un método de búsqueda informada que combina el coste real desde el inicio con una estimación heurística del coste restante.

- La función de evaluación se define como:

$$f(n) = g(n) + h^*(n)$$

donde:

- $g(n)$ representa el coste real desde el nodo inicial hasta el nodo n.
- $h^*(n)$ es la estimación heurística del coste desde n hasta el objetivo.

- El algoritmo selecciona el nodo con menor valor de $f(n)$ para su expansión, garantizando soluciones óptimas (si la heurística utilizada es admisible [pág. 3.4]).

Ventajas y desventajas

- **Ventajas**: completo y óptimo con heurísticas adecuadas.
- **Desventajas**: elevado consumo de memoria y dependencia de la calidad de la heurística.

```
ALGORITMO A*(E_Inicial, E_Objetivo, h*)
   Q ← [(E_Inicial, 0, h*(E_Inicial) )]

   MIENTRAS Q ≠ [] HACER
      [(E_Actual, G_Actual, _ ) | Tail] ← Q
      Q ← Tail

      SI E_Actual == E_Objetivo ENTONCES
         DEVUELVE E_Actual

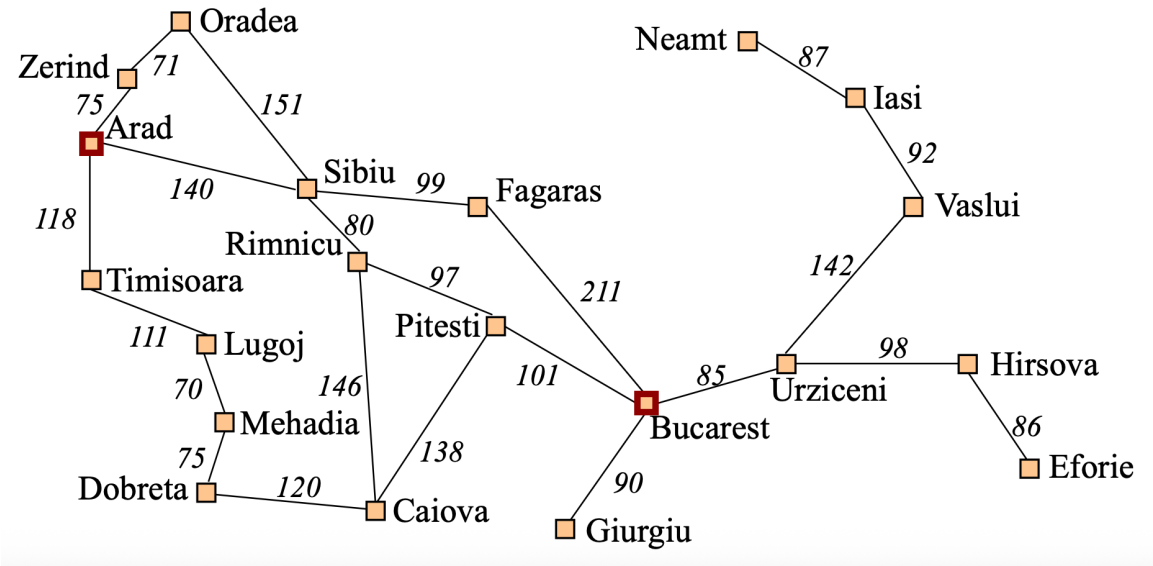
      PARA CADA <Vecino,G> EN vecinos(E_Actual) HACER
         G_Vecino ← G_Actual + G
         F_Vecino ← G_Vecino + h*(Vecino)
         Q ← Q ++ [(Vecino, G_Vecino, F_Vecino)]
         Q ← sort(Q)
   DEVUELVE Error
```

- Pseudocódigo para algoritmo A*:
  - Considera el coste real `G` de ir de `E_Actual` a `Vecino`:

`(A,g(A),f(A))`: el nodo `A` tiene una coste real desde inicio de `g(A)` y un coste estimado total de inicio a destino de `f(A)`.

  - Añade el nodo y los costes a `Q` y los ordena de manera ascendente según `f(A)`.

Ejercicio 10 Encontrar un camino desde Arad hasta Bucarest usando el algoritmo A^* con la función heurística h^* adjunta.



Ciudad	$h^*(n)$		
Arad	366	Mehadia	241
Bucarest	0	Neamt	234
Caiova	160	Oradea	380
Dobreta	242	Pitesti	98
Eforie	161	Rimnicu	193
Fagaras	178	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Tabla 3.2: $h^*(n)$: distancia desde n hasta Bucarest.

3.4. Heurísticas admisibles y consistentes

Heurísticas admisibles

Una heurística es **admisible** (u optimista) si nunca sobreestima el coste real mínimo para alcanzar el objetivo:

$$h^*(n) \leq h(n)$$

donde $h(n)$ representa el coste real óptimo desde el nodo n hasta el objetivo.

- El uso de heurísticas admisibles garantiza que el algoritmo A* encuentre una solución óptima.

Heurísticas consistentes

Una heurística es **consistente** (o monótona) si cumple la siguiente condición para cualquier nodo n y su sucesor n':

$$h^*(n) \leq g(n,n') + h^*(n')$$

donde $g(n,n')$ es el coste de la acción que lleva de n a n'.

- Las heurísticas consistentes aseguran que los valores de $f(n)$ no disminuyan a lo largo de un camino y simplifican la implementación del algoritmo A*.
- Cabe destacar que toda heurística consistente es admisible, pero no toda heurística admisible es consistente.

Universidad Rey Juan Carlos

Tema 4

Búsqueda multiagente

4.1 Entornos multiagente 26

4.2 Juegos competitivos y cooperativos 27

4.3 Algoritmo Minimax 27

4.4 Poda alfa-beta 29

4.5 Introducción a juegos estocásticos 31

4.1. Entornos multiagente

- La búsqueda multiagente estudia problemas en los que intervienen varios agentes que toman decisiones de forma simultánea o secuencial.
- A diferencia de la búsqueda clásica, el resultado de una acción depende no solo del entorno, sino también de las acciones realizadas por otros agentes.
- En este tipo de problemas:
  - Existen múltiples agentes con objetivos propios.
  - El entorno es dinámico y estratégico.
  - Cada agente debe razonar teniendo en cuenta el comportamiento de los demás.
- Ejemplos típicos de búsqueda multiagente incluyen juegos de mesa, robots colaborativos y sistemas de negociación automática.

- Un entorno multiagente se caracteriza por la interacción entre dos o más agentes. Estos entornos pueden clasificarse según distintas dimensiones:
  - **Cooperativos**: los agentes comparten objetivos comunes.
  - **Competitivos**: los objetivos de los agentes están en conflicto.
  - **Mixtos**: algunos objetivos compartidos, otros opuestos.

- Otras propiedades relevantes son:
  - Deterministas o estocásticos.
  - De información completa o incompleta.
  - Por turnos o con acciones simultáneas.
- La complejidad de estos entornos surge de la necesidad de anticipar las decisiones de otros agentes.

4.2. Juegos competitivos y cooperativos

Juegos competitivos

La teoría de juegos estudia situaciones en las que varios agentes toman decisiones estratégicas y el resultado para cada uno depende de las acciones de los demás.

En los **escenarios competitivos**:

- Los agentes tienen objetivos opuestos o en conflicto.
- La ganancia de un agente implica la pérdida de otro (juegos de suma cero).
- Cada agente asume que sus oponentes actúan de forma racional.

- Tipos de juegos dependen:
  - Jugadores: dos (tres-en-raya) | ...
  - Azar: con (parchís) | sin (ajedrez).
  - Información: completa (damas) | incompleta (poker).
- Objetivo: anticipar la acción del adversario y encontrar la estrategia óptima.
- Consideraciones / notación:
  - Hay un estado inicial s_0.
  - Hay estados terminales (objetivo): $terminal(s) \rightarrow true|false$
  - Los estados terminales tienen valor de utilidad: $utilidad(s) \rightarrow k,\ k \in \mathbb{R}$

4.3. Algoritmo Minimax

Definición: El algoritmo Minimax es un método de búsqueda utilizado en juegos competitivos por turnos, deterministas y de suma cero, con información completa. [John von Neumann, 1928]

- Se basa en dos tipos de jugadores:
  - **MAX**: intenta maximizar la utilidad.
  - **MIN**: intenta minimizar la utilidad.
- El algoritmo construye un árbol de juego donde los nodos representan estados y las hojas (estados terminales) contienen valores de utilidad. La decisión óptima se obtiene propagando los valores a la raíz del árbol.
- Aunque Minimax garantiza la mejor decisión posible, su principal inconveniente es el alto coste computacional.

```
ALGORITMO Minimax(E_Inicial)
   DEVUELVE valor_MAX(E_Inicial)

FUNCION valor_MAX(E_Actual)
   SI terminal(E_Actual) ENTONCES
       DEVUELVE utilidad(E_Actual)
   Valor ← −∞
   PARA CADA Vecino EN vecinos(E_Actual) HACER
       Valor ← max(Valor, valor_MIN(Vecino) )
   DEVUELVE Valor

FUNCION valor_MIN(E_Actual)
   SI terminal(E_Actual) ENTONCES
       DEVUELVE utilidad(E_Actual)
   Valor ← ∞
   PARA CADA Vecino EN vecinos(E_Actual) HACER
       Valor ← min(Valor, valor_MAX(Vecino) )
   DEVUELVE Valor
```

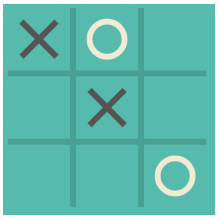
- Pseudocódigo para Minimax:
  - Empieza el agente MAX
  - Dos funciones mutuamente recursivas.

Heurísticas

- Introducir prueba de suspensión en cierta profundidad, en estados reposo...
- Estimar valor de utilidad en estados no terminales: $utilidad^*(s) \to k,\ k \in \mathbb{R}$
- Decisión no óptima, depende de la calidad de la heurística.

Ejercicio 11 Adapta el algoritmo `Minimax` (llámalo `Minimax`*) para suspender a un profundidad `p` y estimar el valor de utilidad con `utilidad`*.

Ejercicio 12 Dada la siguiente posición del tres-en-raya (MAX juega X):

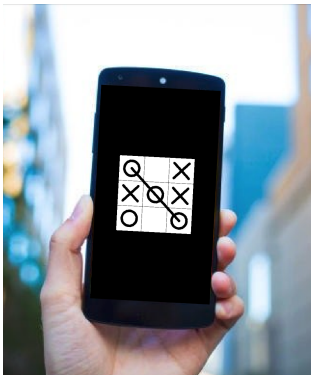


a) Calcula el valor de Minimax para los siguientes 5 estados posibles.

b) ¿Cuál es el siguiente movimiento óptimo de MAX según Minimax?

c) Estima el valor de Minimax* (**Ejercicio 11**) para profundidad 3 y $utilidad^*(s)$ = líneas abiertas para MAX − líneas abiertas para MIN.

- **Una cosa más**: Minimax se ha aplicado tradicionalmente en el tres-en-raya, inocente juego que, en Juegos de guerra (1983), se transforma en una metáfora del conflicto nuclear:
  - En `https://github.com/Xuaco/Tic-Tac-Toe` tenéis código, y manualc para implementarlo en Java usando Processing.
  - En (Arias y Carro 2019) se emplea como benchmark (también en su variante 4x4) para evaluar mejoras avanzadas en la estrategia de búsqueda de Prolog (ver (Arias 2024:Tema 3)).



4.4. Poda alfa-beta

Definición: La poda alfa–beta es una optimización del algoritmo Minimax que reduce el número de nodos evaluados sin alterar el resultado final.

- Se utilizan dos parámetros:
  - α: el mejor valor encontrado hasta el momento para MAX.
  - β: el mejor valor encontrado hasta el momento para MIN.
- Cuando se detecta que una rama no puede mejorar la solución actual, se deja de explorar. En el mejor de los casos, la complejidad se reduce de $O(b^d)$ a $O(b^{d/2})$, en el caso medio a $O(b^{3d/4})$.

```
ALGORITMO Alfa-Beta(E_Inicial)
   DEVUELVE valor_MAX(E_Inicial, −∞, ∞)


FUNCION valor_MAX(E_Actual, α, β)
```

```
    SI terminal(E_Actual) ENTONCES
        DEVUELVE utilidad(E_Actual)
    PARA CADA Vecino EN vecinos(E_Actual) HACER
        α ← max(α, valor_MIN(Vecino,α,β) )
        SI α ≥ β ENTONCES
            SALIR DEL BUCLE  % poda beta
    DEVUELVE α

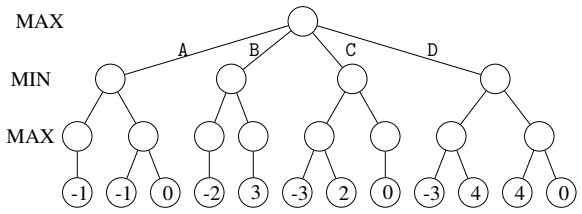

FUNCION valor_MIN(E_Actual, α, β)
    SI terminal(E_Actual) ENTONCES
        DEVUELVE utilidad(E_Actual)
    PARA CADA Vecino EN vecinos(E_Actual) HACER
        β ← min(β, valor_MAX(Vecino,α,β) )
        SI β ≤ α ENTONCES
            SALIR DEL BUCLE  % poda alfa
    DEVUELVE β
```

- Pseudocódigo para Minimax con poda alfa-beta:
  - α: es el mejor valor de evaluación para MAX en el camino hasta estado actual.
  - β: es el mejor para MIN.
- Poda alfa:
  - $\beta \leq \alpha$
  - `valor_MIN = min(`β`, ...)` $\leq$ β
  - `valor_MAX = max(`α`,valor_MIN,...)`
  - Como $\beta \leq \alpha$, entonces `valor_MAX` no depende de `valor_MIN` y no hay que explorar otros vecinos de MIN.

Ejercicio 13 Explica con tus palabras la poda beta en un nodo MAX.

Ejercicio 14 Dibuja un árbol de juego con poda alfa profunda, i.e., poda de los hijos de un nodo MIN al comparar con el α de un ancestro lejano.

Ejercicio 15 Dado el árbol de juego de la figura:



a) ¿Cuántos nodos evitamos generar respecto al algoritmo `Minimax` si aplicamos el algoritmo `Alfa-Beta`?

b) ¿Cuál es el siguiente movimiento óptimo de MAX `(A,B,C o D)`?

4.5. Introducción a juegos estocásticos

Los juegos estocásticos

Incorporan incertidumbre / aleatoriedad en los resultados de las acciones.

Características:

- El estado siguiente depende de una probabilidad.
- Puede haber eventos aleatorios (dados, cartas, ruido del entorno).

Ejemplos:

- Póker.
- Backgammon.
- Juegos con fallos de sensores o actuadores.

ExpectiMinimax: Extensión del Minimax:

- Se Añade un nuevo jugador, AZAR, cuya decisión es aleatoria.
- Los vecinos de un nodo AZAR:
  - Son los posibles resultados del evento, p.ej., `cara` o `cruz`.
  - Cada resultado tiene asociado la probabilidad de que ocurra: para la moneda es `cara;1/2` y `cruz;1/2`.

```
ALGORITMO ExpectiMinimax(E_Inicial)
   DEVUELVE valor(E_Inicial, MAX)

FUNCION valor(E_Actual, Juega)
   SI terminal(E_Actual) ENTONCES
       Valor ← utilidad(E_Actual)
   SINO SI Juega == MAX ENTONCES
       Valor ← −∞
       PARA CADA Vecino EN vecinos(E_Actual) HACER
            Valor ← max(Valor, valor(AZAR_MAX,Vecino) )
   SINO SI Juega == MIN ENTONCES
       Valor ← ∞
       PARA CADA Vecino EN vecinos(E_Actual) HACER
            Valor ← min(Valor, valor(AZAR_MIN,Vecino) )
   SINO
       Valor ← 0
       PARA CADA <Vecino,P> EN vecinos(E_Actual) HACER
            Valor ← Valor + P * ( Juega == AZAR_MAX ? valor(Vecino, MIN) : valor(Vecino,MAX) )
   DEVUELVE Valor
```

- Pseudocódigo para ExpectiMinimax:

`<Vecino,P>` el nodo `Vecino` tiene una probabilidad `P` asociada.

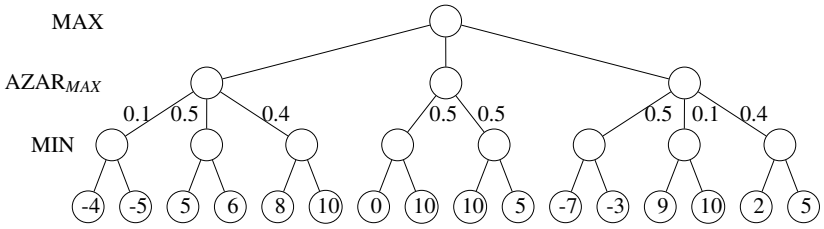
`C ? S : N:` abreviación de `SI C ENTONCES S SINO N`.

- La escala de los valores de `utilidad` afectan al resultado:
  - No devolver ∞ o $-\infty$.
  - Escalar los valores en intervalo finito, p.ej. $[1, 10]$.

Ejercicio 16 * Implementa ExpectiMinimax con poda alfa-beta.

(Russell y Norvig 2021:sección 6.5.1)

Ejercicio 17 Dado el árbol de juego de la figura:



a) Calcula el valor de cada nodo aplicando `ExpectiMinimax`.

b) * ¿Cuántos nodos evitamos generar respecto `ExpectiMinimax` si realizamos una pofa alfa-beta, sabiendo que $utilidad(n) \in [-10, 10]$?

[0]* Ejercicio avanzado; no se espera su resolución dentro del marco de la asignatura.

Módulo III

Representación del conocimiento

Universidad Rey Juan Carlos

Tema 5

Representación probabilística del conocimiento

5.1 Incertidumbre y razonamiento probabilístico 34

5.2 Probabilidad condicional 34

5.3 Regla de Bayes 37

5.4 Independencia y dependencia probabilística 38

5.1. Incertidumbre y razonamiento probabilístico

- En la vida real, en la IA (y en ciberseguridad), muchas decisiones deben tomarse con información incompleta o incierta (p.ej., alertas ruidosas, datos faltantes, ataques desconocidos). La representación probabilística permite modelar esta incertidumbre de forma rigurosa.
- **Incertidumbre**: falta de certeza sobre el estado real del sistema (por ejemplo, si una alerta corresponde a un ataque real).
- **Razonamiento probabilístico**, usa la probabilidad como medida cuantitativa de creencia.
  - Permite combinar evidencias, actualizar creencias y tomar decisiones informadas.
- **Ejemplo**: Probabilidad de que un tráfico sea malicioso dado un conjunto de indicadores (IP sospechosa, puerto inusual, patrón de comportamiento).

5.2. Probabilidad condicional

C	P(C)
sol	0,2
lluvia	0,5
niebla	?

Tabla 5.1: Distribución de probabilidad.

- Variable aleatoria: parte del mundo con incertidumbre, con un dominio de valores.
- Distribución de probabilidad: asigna una probabilidad P a cada valor x del dominio:

$$0 \leq P(V = x) \leq 1 \qquad \sum_{x \in Dom(V)} P(V = x) = 1$$

- **Definición**: La probabilidad condicional expresa la probabilidad de un evento A dado que otro ya ocurrió B:

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

A: evento de interés.

B: evidencia observada.

- **Ejemplo**: Distribución de probabilidad de:
  - T: Temperatura.
    Dominio: Dom(T) = {frio, calor}.
    Probabilidad: P(frio)=P(calor)=0,5.
  - C: Clima.

Ejercicio 18 Dada la distribución de probabilidad de la variable C de la tabla 5.1, ¿Cuál es la probabilidad P(C=niebla)?

- Distribución conjunta de $P(V_1 = x_1,\ V_2 = x_2,\ \ldots,\ V_n = x_n)$, se denota como, $P(x_1, x_2, \ldots, x_n)$, y cumple:

$$P(x_1, x_2, \ldots, x_n) \geq 0$$

$$\sum_{x_1 \in Dom(V_1)} \cdots \sum_{x_n \in Dom(V_n)} P(x_1, x_2, \ldots, x_n) = 1$$

- **Ejemplo**: Temperatura y clima:

- Un evento E es un conjunto de posibles valores de la distribución. Su probabilidad se calcula como:

$$P(E) = \sum_{(x_1,\ldots,x_n) \in E} P(x_1, \ldots, x_n)$$

T	C	P(T,C)
frio	sol	0,2
frio	lluvia	0,3
calor	sol	0,4
calor	lluvia	0,1

Tabla 5.2: Distribución conjunta.

Ejercicio 19 Dada la tabla 5.2, calcula la probabilidad P(E) para los eventos:

a) E = calor y sol.

b) E = calor.

c) E = calor o sol.

- La Distribución marginal de un subconjunto de variables $\vec{S}$ se obtiene a partir de la distribución conjunta de todas las variables $\vec{X}$.

$$P(\vec{S}) = \sum_{\vec{X}\setminus\vec{S}} P(\vec{X})$$

Ejercicio 20 Dada una distribución conjunta con tres variables (V_1, V_2, V_3), indica las distribuciones marginales y la fórmula correspondiente.

- **Ejemplo**: Dada la distribución conjunta para temperatura T y clima C de la Tabla 5.2 calcula las distribución marginal correspondientes.

  P(T):

  P(T=frio) = $\sum_{C}$ P(T=frio,C) =

  = P(T=frio, C=sol) + P(T=frio, C=lluvia) =

  $= 0,2 + 0,3 = 0,5$

  P(T=calor) = $0,4 + 0,1 = 0,5$

  P(C):

  P(C=sol) = $0,2 + 0,4 = 0,6$

  P(C=lluvia) = $0,3 + 0,1 = 0,4$

- **Ejemplo**: Dada la distribución conjunta de T y C de la tabla 5.2, calcularemos la probabilidad de C cuando hace calor, es decir, la probabilidad condicional $P(C|T = calor)$.

a) Primero aplicando la fórmula:

P(C=sol | T=calor) =

$$= \frac{\text{P(C=sol,T=calor)}}{\text{P(calor)}} = \frac{0,4}{0,5} = 0,8$$

P(C=lluvia | T=calor) =

$$= \frac{\text{P(C=lluvia, T=calor)}}{\text{P(calor)}} = \frac{0,1}{0,5} = 0,2$$

b) Ahora, seleccionando las probabilidades conjuntas que concuerdan con la evidencia:

T	C	P(T,C)
calor	sol	0,4
calor	lluvia	0,1

Y normalizamos para que sume 1:

C	P(C | T=calor)
sol	0,8
lluvia	0,2

Ejercicio 21 Calcula la probabilidad de C cuando hace frio, i.e., P(C | T=frio).

5.3. Regla de Bayes

- **Definición**: La regla de Bayes permite invertir probabilidades condicionales:

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

  Donde:
  - $P(A)$: probabilidad a priori.
  - $P(B|A)$: verosimilitud.
  - $P(A|B)$: probabilidad a posteriori.
- **Ejemplo**: Actualizar la probabilidad de intrusión tras observar un comportamiento anómalo.
  - $P(\text{Intrusión}|\text{Alerta IDS}) = 0,90$
  - $P(\text{Intrusión}|\text{Alerta IDS}, \text{Comportamento anómalo}) = 0,91$
- **Regla del producto**: Obtenemos la probabilidad conjunta $P(A,B)$ a partir de la probabilidad condicional $P(A|B)$:

$$P(A|B) = \frac{P(A,B)}{P(B)} \Leftrightarrow P(A,B) = P(A|B) \cdot P(B)$$

- **Regla de la cadena**: Nos permite escribir una distribución conjunta como producto incremental de distribuciones condicionales:

$$P(x_1,x_2,x_3) = P(x_1)\cdot P(x_2|x_1)\cdot P(x_3|x_1,x_2)$$

En general $P(x_1,x_2,\ldots,x_n) = \prod_i P(x_i|x_1,\ldots,x_{i-1})$

Ejercicio 22 Demuestra la Regla de Bayes $P(A|B) = \frac{P(B|A)\cdot P(A)}{P(B)}$ a partir de la regla del producto y la regla de la cadena.

5.4. Independencia y dependencia probabilística

- **Definición** En IA (aplicada o no a la ciberseguridad), no todos los eventos son iguales: algunos ocurren de forma aislada y otros están relacionados. Por lo tanto disntinguimos entre varios escenarios:

1. **Independencia probabilística**

   Dos eventos aleatorios A y B son independientes si la ocurrencia de uno no afecta a la probabilidad de que ocurra el otro.

   $$P(A,B) = P(A)\cdot P(B)$$

   Esto implica también que:

   $$P(A|B) = P(A) \text{ y } P(B|A) = P(B)$$

   Es decir, conocer que B ha ocurrido no aporta información sobre A, y viceversa. Interpretación intuitiva:

   - No existe relación causal ni informativa entre los eventos.
   - Observar uno no cambia nuestras creencias sobre el otro.

2. **Dependencia probabilística**

   Dos eventos A y B son dependientes si la ocurrencia de uno sí modifica la probabilidad del otro:

   $$P(A|B) \neq P(A)$$

   La observación de un evento aporta información relevante sobre el otro.

   - Existe relación causal, correlación o una causa común.
   - Ver un evento cambia nuestro grado de creencia sobre el otro.

3. **Independencia condicional**

   Dos eventos pueden ser dependientes en general, pero independientes dado un tercer evento C.

   $$P(A,B|C) = P(A|C)\cdot P(B|C)$$

- Ejemplo: Cuando aumenta el consumo de helado (A), también aumenta el número de asesinatos (B). [Reichenbach, H. (1956)]

Ejercicio 23 ¿Quién es el evento C en el ejemplo?

Detectar la independencia condicional es fundamental en redes bayesianas, porque permite simplificar el modelo sin perder capacidad explicativa.

Universidad Rey Juan Carlos

Tema 6

Redes Bayesianas

6.1 Redes Bayesianas **40**

6.1.1 Definición y componentes 40

6.1.2 Grafos dirigidos acíclicos 41

6.1.3 Tablas de probabilidad condicional 41

6.2 Construcción de modelos bayesianos **42**

6.1. Redes Bayesianas

6.1.1. Definición y componentes

- **Definición**: Una red bayesiana es un modelo probabilístico gráfico que representa un conjunto de variables aleatorias y sus relaciones de dependencia condicional mediante un grafo dirigido acíclico (DAG).
- Formalmente, una red bayesiana define una distribución de probabilidad conjunta sobre un conjunto de variables $V = \{V_1, V_2, \ldots, V_n\}$, que se factoriza de acuerdo con la estructura del grafo como:

$$P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P(V_i \mid \text{Pa}(V_i))$$

  donde $\text{Pa}(V_i)$ denota el conjunto de nodos padres de la variable V_i.

- **Modelo probabilístico**: Las redes bayesianas permiten representar y razonar bajo incertidumbre, combinando conocimiento previo, evidencia observada y relaciones de dependencia entre variables. Son especialmente útiles para modelar sistemas complejos, inciertos o parcialmente observables.
- ¡Los modelos son siempre simplificaciones! ¡Son aproximaciones!

- Pueden no representar a todas las variables o todas las interacciones entre variables.
- "All models are wrong; but some are useful."

[George E. P. Box (1976)]

6.1.2. Grafos dirigidos acíclicos

- **Definición**: En un grafo dirigido acíclico que define la estructura de una red bayesiana:
  - Los nodos representan variables aleatorias.
  - Las aristas dirigidas representan dependencias probabilísticas directas.
- La ausencia de una arista entre dos nodos indica una relación de independencia condicional bajo ciertas condiciones.
- Cada variable V_i es condicionalmente independiente de sus no-descendientes dado el valor de sus padres:

  $$V_i \perp\!\!\!\perp \text{NoDesc}(V_i) \mid \text{Pa}(V_i)$$

  esta propiedad permite representar de forma compacta de la distribución conjunta sobre todas las variables.

6.1.3. Tablas de probabilidad condicional

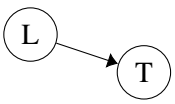
- **Definición**: En una red bayesiana, cada nodo V tiene asociada una tabla de probabilidad condicional (CPT) que define la probabilidad del nodo dado sus padres:

  $$P(V \mid \text{Pa}(V))$$

  - Si no tiene padres, la probabilidad es a priori P(V).
- Para cada valor de Pa(V) se debe cumplir:

  $$\sum_{x \in V} \text{P(V=x} \mid \text{Pa(V)=y)} = 1$$

- **Ejemplo**: Dada la red bayesiana:



L	P(L)
sí	0.3
no	0.7

L	T	P(T | L)
sí	alto	0.8
sí	bajo	0.2
no	alto	0.3
no	bajo	0.7

Observamos que la lluvia $L \in \{si, no\}$ influye en el trafico $T \in \{alto, bajo\}$.
...y a partir de la CPT sabemos que:

- Si llueve, la probabilidad de tráfico alto es del 80 %.
- Si no llueve, esta baja a 30 %.
- etc...

6.2. Construcción de modelos bayesianos

- Pasos para **construir un modelo bayesiano**:
  1. Identificar variables relevantes
  2. Definir relaciones de dependencia
  3. Diseñar el DAG
  4. Especificar las CPT (expertos o datos)
  5. Validar el modelo

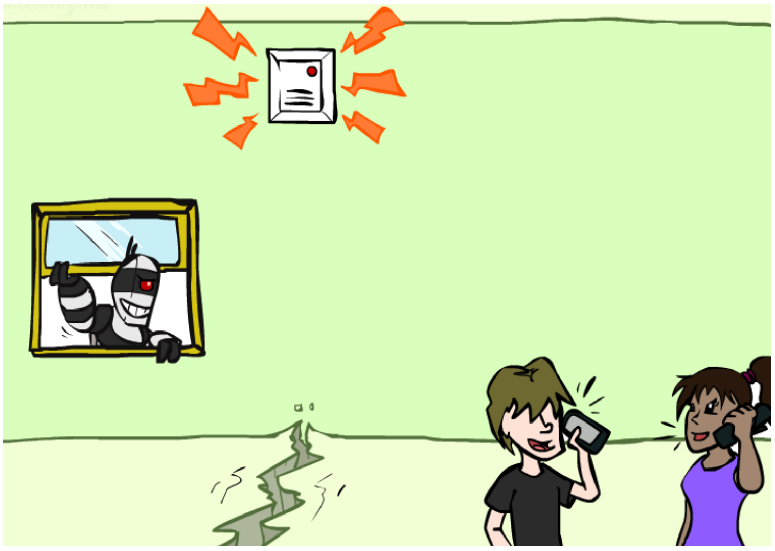


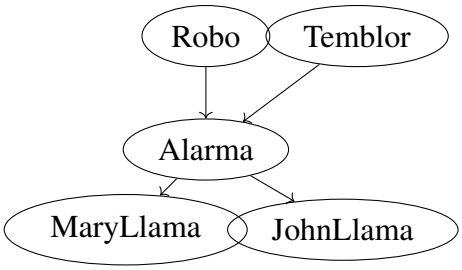
Figura 6.1: Suena la alarma por robo o terremoto...

- Ejemplo: Alarma (Fig. 6.1):

  1. Variables relevantes:
  - R: Robo
  - A: Alarma
  - M: Mary llama
  - J: John llama
  - T: Temblor

2. Dependencias
- Robo y Temblor son independientes.
- Dada la variable Alarma, JohnLlama y MaryLlama son condicionalmente independientes del resto de variables.
- En particular, no dependen de si hubo robo, temblor ni de si la otra persona llamó.

3. DAG:



4. CPT:

R	P(R)
si	0.001
no	0.999

T	P(T)
si	0.002
no	0.998

R	T	A	P(A|R,T)
si	si	si	0.95
si	si	no	0.05
si	no	si	0.94
si	no	no	0.06
no	si	si	0.29
no	si	no	0.71
no	no	si	0.001
no	no	no	0.999

A	M	P(M|A)
si	si	0.7
si	no	0.3
no	si	0.01
no	no	0.99

A	J	P(J|A)
si	si	0.9
si	no	0.1
no	si	0.05
no	no	0.95

5. Para validar el modelo no necesitamos calcular el CPT conjunta ($2^5 = 32$) es suficiente con estas 20:

$$P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P(V_i \mid \mathrm{Pa}(V_i))$$

Ejercicio 24 Dado el ejemplo Alarma de la Figura 6.1 calcula las siguientes probabilidades:

a) Se ha producido un robo. No se han percibido temblores, pero la alarma ha sonado. Mary ha llamado para avisar, mientras que John no ha dado señales.

b) Dado $\vec{V} = (R, T, A, M, J)$ calcula $P(\vec{V}) = P(si, no, si, si, no)$.

Ejercicio 25 Verifica que todas las tablas CPTs del ejemplo Alarma de la Figura 6.1 están normalizadas (instancia las fórmulas requeridas para ello).

Universidad Rey Juan Carlos

Tema 7

Inferencia en Redes Bayesianas

7.1 Inferencia exacta **45**
7.1.1 Algoritmo de Enumeración 46
7.1.2 Algoritmo de Eliminación de variables 47
7.1.3 'Implementacón' en Python 48
7.2 Inferencia aproximada **49**
7.2.1 Métodos principales 49
7.2.2 Algoritmos de propagación de creencias 50

7.1. Inferencia exacta

- La inferencia en redes bayesianas permite determinar la probabilidad de ciertos eventos o estados dado un conjunto de evidencias.
  - Se trata de calcular la probabilidad de nodos desconocidos (ocultos) dado la información observada.
- Ejemplo: ¿Cuál es la probabilidad de que exista malware si la alarma del IDS ha sonado y el tráfico de red es anómalo?

- **Definición**: La inferencia exacta busca calcular probabilidades precisas usando todas las dependencias y la estructura de la red. Métodos comunes:
  - Enumeración: Se calcula toda la distribución y luego se eliminan las variables ocultas.
  - Eliminación de variables: se suman probabilidades marginalizando nodos intermedios.
- Ventaja: resultados exactos.
- Desventaja: computacionalmente costoso en redes grandes.

7.1.1. Algoritmo de Enumeración

- **Algoritmo de Enumeración**: Consiste en calcular todas las combinaciones posibles de las variables no observadas.
  - Muy simple conceptualmente.
  - Muy costoso para redes medianas o grandes (explosión combinatoria).
- **Ejemplo:** Dado el ejemplo de la Alarma de la Figura 6.1, calcular la probabilidad de que haya habido un robo dado que la alarma ha sonado, Mary ha llamado y John no:

$$\text{P(R=si} \mid \text{A=si, M=si, J=no)}$$

  - Variables ocultas: Robo (R) y Temblor (T)
  - Variables observadas: Alarma (A=si), Mary (M=si), John (J=no)
- **Ejemplo (cont.):**

Paso 1: Enumerar todas las combinaciones de variables ocultas R y T

R	T	$P(R)$	$P(T)$	$P(A\|R,T)$	$P(M\|A)$	$P(J\|A)$	Numerador
si	*si*	0,001	0,002	0,95	0,7	0,1	$1{,}33 \cdot 10^{-7}$
si	*no*	0,001	0,998	0,94	0,7	0,1	$6{,}55 \cdot 10^{-5}$
no	*si*	0,999	0,002	0,29	0,7	0,1	$4{,}06 \cdot 10^{-5}$
no	*no*	0,999	0,998	0,001	0,7	0,1	$6{,}99 \cdot 10^{-5}$

Numerador (producto de probabilidades para cada fila)

Paso 2: Calcular numerador de Bayes para R=si:

solo sumas las filas donde R=si

Numerador = $\sum_T$ P(R=si, T, A=si, M=si, J=no) =

$$= 1{,}33 \cdot 10^{-7} + 6{,}55 \cdot 10^{-5} \approx 6{,}56 \cdot 10^{-5}$$

- **Ejemplo (cont.):**

Paso 3: Calcular denominador de Bayes:

sumar sobre todas las combinaciones de las variables ocultas R y T.

$\sum_{R,T}$ P(R, T, A=si, M=si, J=no) =

$$= \left(6{,}56 \cdot 10^{-5}\right) + 4{,}06 \cdot 10^{-5} + 6{,}99 \cdot 10^{-5} \approx 0{,}000161$$

Paso 4: Inferencia por enumeración: resultado del paso 2 entre el del paso 3.

$$\text{P(R=si} \mid \text{A=si, M=si, J=no)} = \frac{\text{P(R=si,A=si,M=si,J=no)}}{\text{P(A=si,M=si,J=no)}} = \frac{6{,}56 \cdot 10^{-5}}{0{,}000161} \approx 0{,}408$$

Resultado: dadas las evidencias hay un **41 %** de probabilidad de robo.

7.1.2. Algoritmo de Eliminación de variables

- **Algoritmo de Eliminación de Variables**: Consiste en sumar (marginalizar) sistemáticamente las variables no observadas para simplificar el cálculo de probabilidades.
- **Ejemplo:** Dado el ejemplo de la Alarma de la Figura 6.1, calcular la probabilidad de que haya habido un robo dado que la alarma ha sonado, Mary ha llamado y John no:

$$\text{P(R=si} \mid \text{A=si, M=si, J=no)}$$

  - Variables ocultas: Robo (R) y Temblor (T)
  - Variables observadas: Alarma (A=si), Mary (M=si), John (J=no)

- **Ejemplo (cont.):**

Paso 1: Factorización de la probabilidad conjunta:

$$\text{P(R,T,A,M,J)} = \text{P(R)} \cdot \text{P(T)} \cdot \text{P(A} \mid \text{R,T)} \cdot \text{P(M} \mid \text{A)} \cdot \text{P(J} \mid \text{A)}$$

Paso 2: Elegir orden de eliminación:

Eliminamos T primero, por lo tanto calculamos f(R,A,M,J).

Paso 3: Marginalización sobre T.

$$\text{f(R,A,M,J)} = \sum_T \text{P(R)} \cdot \Big(\text{P(T)} \cdot \text{P(A} \mid \text{R,T)} \Big) \cdot \text{P(M} \mid \text{A)} \cdot \text{P(J} \mid \text{A)} =$$

$$= \text{P(R)} \cdot \Big(\text{P(T=si)} \cdot \text{P(A} \mid \text{R,T=si)} + \text{P(T=no)} \cdot \text{P(A} \mid \text{R,T=no)} \Big) \cdot \text{P(M} \mid \text{A)} \cdot \text{P(J} \mid \text{A)}$$

- **Ejemplo (cont.):**

Paso 4: Sustituir probabilidades numéricas para $R = si$ y $R = no$.

$$f(R = \text{si}, A = \text{si}, M = \text{si}, J = \text{no}) =$$

$$= 0{,}001 \cdot \big(0{,}002 \cdot 0{,}95 + 0{,}998 \cdot 0{,}94\big) \cdot 0{,}7 \cdot 0{,}1 \approx 6{,}56 \cdot 10^{-5}$$

$$f(R = \text{no}, A = \text{si}, M = \text{si}, J = \text{no}) =$$

$$= 0{,}999 \cdot \big(0{,}002 \cdot 0{,}29 + 0{,}998 \cdot 0{,}001\big) \cdot 0{,}7 \cdot 0{,}1 \approx 1{,}105 \cdot 10^{-4}$$

Paso 5: Calcular la probabilidad condicionada

$$\text{P(R=si} \mid \text{A=si,M=si,J=no)} = \frac{\text{f(R=si, A=si,M=si,J=no)}}{\text{f(A=si,M=si,J=no)}} = \frac{6{,}56 \cdot 10^{-5}}{6{,}56 \cdot 10^{-5} + 1{,}105 \cdot 10^{-4}} \approx 0{,}373$$

Resultado: dadas las evidencias hay un **37 %** de probabilidad de robo.[1]

7.1.3. 'Implementacón' en Python

```
# Probabilidad conjunta P(R,T,A,M,J)
def P(r, t, a, m, j):
  return ( P_R[r] *
           P_T[t] *
           P_cond(a, P_A[(r,t)]) *
           P_cond(m, P_M[a]) *
           P_cond(j, P_J[a]) )
# Factor f(R,A,M,J) eliminando T
def f(r, a, m, j):
  suma_T = (
    P_T['si'] * P_cond(a, P_A[(r,'si')]) +
    P_T['no'] * P_cond(a, P_A[(r,'no')])
  )
  return ( P_R[r] *
           suma_T *
           P_cond(m, P_M[a]) *
           P_cond(j, P_J[a]) )
# Probabilidades condicionales binarias
def P_cond(v, p_si):
    return p_si if v == 'si' else 1 - p_si


# Probabilidades marginales / CPT
P_R = {'si': 0.001, 'no': 0.999}
P_T = {'si': 0.002, 'no': 0.998}
P_A = {('si','si'): 0.95,
       ('si','no'): 0.94,
       ('no','si'): 0.29,
       ('no','no'): 0.001}   # P(A=si | R,T)
P_M = {'si': 0.7, 'no': 0.01}  # P(M=si | A)
P_J = {'si': 0.9, 'no': 0.05}  # P(J=si | A)
# Calculo por Enumeración
n1 = sum(P('si',t,'si','si','no') for t in ['si','no'])
x1 = sum(P('no',t,'si','si','no') for t in ['si','no'])
P_enum = n1 / (n1+x1)
# Calculo por Eliminación de Variables
n2,x2 = f('si', 'si','si','no'), f('no', 'si','si','no')
```

[1] La diferencia con respecto lo obtenido con Enumeración se debe a redondeo interno.

```
P_elim = n2 / (n2+x2)

desc = "P(R=si | A=si, M=si, J=no)"
print(f"Enumeración:        {desc} = {P_enum}")
print(f"Eliminación de Var: {desc} = {P_elim}")
```

- Implementación del ejemplo de la Figura 6.1. Try it in `https://paiza.io`.

7.2. Inferencia aproximada

- **Definición**: La inferencia aproximada es un conjunto de métodos que estiman las probabilidades en redes bayesianas cuando la inferencia exacta es muy costosa o inviable.
- Se utiliza en redes grandes o con muchas dependencias, donde calcular probabilidades exactas es computacionalmente pesado.
  - Genera una estimación de la probabilidad posterior basada en muestras o simplificaciones del modelo.
- Objetivo: obtener resultados útiles con menor coste computacional que la inferencia exacta.

7.2.1. Métodos principales

- **Muestreo probabilístico** (Sampling).
  - Se generan múltiples escenarios (muestras) de la red según la distribución conjunta.
  - Se cuentan cuántas veces ocurren los eventos de interés para estimar probabilidades.

  Técnicas comunes:
  - Muestreo directo.
  - Muestreo por rechazo.
  - Muestreo de Gibbs.
  - Monte Carlo (MCMC).

- **Variational Inference**.
  - Se aproxima la distribución posterior compleja mediante una más simple.
  - Optimiza los parámetros de la distribución simplificada para que se acerque a la verdadera posterior.
  - Muy usado en redes grandes o con variables continuas.

- **Loopy Belief Propagation**.
  - Extiende la propagación de creencias a redes con ciclos.
  - No garantiza resultados exactos, pero suele converger a buenas aproximaciones.

7.2.2. Algoritmos de propagación de creencias

- **Definición**: Los algoritmos de propagación de creencias permiten realizar inferencia en redes bayesianas de manera eficiente, especialmente en redes tipo árbol o de cliques.
- Concepto:
  - Cada nodo envía y recibe mensajes probabilísticos a sus vecinos.
  - Estos mensajes representan la creencia de un nodo sobre otro nodo dado la evidencia observada.
  - Permite calcular probabilidades marginales de manera distribuida.

- Ventajas:
  - Reduce el costo computacional.
  - Distribuido: los cálculos se pueden realizar localmente en cada nodo.
  - Muy eficiente para redes de gran tamaño o de estructura jerárquica.
- Ejemplo: detección de intrusión con sensores distribuidos.
  - Cada sensor envía mensajes sobre la probabilidad de intrusión.
  - Después de varias iteraciones, cada nodo tiene una creencia actualizada de la probabilidad local y global.

a) **Propagación en árboles**

Tree-structured Belief Propagation

1. Seleccionar un nodo raíz y recorrer el árbol *desde las hojas* (mensaje ascendente).
2. Cada nodo calcula un mensaje basado en sus propias CPT y los mensajes recibidos de sus hijos.
3. Recorrer el árbol desde la raiz hacia los nodos hojas (mensaje descendente) para actualizar creencias en todos los nodos.
4. Una vez finalizada la propagación, cada nodo puede calcular su probabilidad marginal exacta dado la evidencia.

b) **Propagación en grafos con ciclos**

Loopy Belief Propagation

- Extiende el algoritmo a redes que contienen ciclos.

- Los nodos envían mensajes de manera iterativa hasta que los valores convergen o se alcanza un número máximo de iteraciones.
- No garantiza exactitud, pero suele ofrecer aproximaciones útiles.



Universidad
Rey Juan Carlos

Tema 8

Teoría de la decisión

8.1 Fundamentos de la teoría de la decisión **52**

8.2 Utilidad **53**

8.2.1 Funciones de utilidad 53

8.2.2 Utilidad Esperada 53

8.3 Modelos de decisión **53**

8.3.1 Redes de decisión 53

8.3.2 Árboles de Decisión 55

8.3.3 Secuencias y procesos de Markov 55

8.1. Fundamentos de la teoría de la decisión

- **Definición**; La teoría de la decisión en Inteligencia Artificial se centra en cómo elegir la mejor acción cuando existe incertidumbre. La idea fundamental es maximizar la utilidad esperada dadas las evidencias.
- Herramientas principales:
  - Redes de decisión.
  - Árboles de decisión.
  - Secuencias de decisiones.
  - Procesos de Decisión de Markov (MDP).

8.2. Utilidad

8.2.1. Funciones de utilidad

- **Definición**: La Utilidad es una medida (numérica) que representa el grado de beneficio o preferencia de un estado respecto a otro.
  - Permite comparar decisiones (que conducen a distintos estado) bajo incertidumbre.
- Una Función de utilidad asigna un valor numérico a cada estado posible:

$$U : \text{Estados} \to \mathbb{R}$$

- Tipos de funciones de utilidad:
  - Lineal: neutral al riesgo, $U(x) = kx$
  - Cóncava: adverso al riesgo, $U(x) = \sqrt{x}$
  - Convexa: amante del riesgo, $U(x) = x^2$

8.2.2. Utilidad Esperada

- **Definición**: Utilidad Esperada (EU, Expected Utility) se utiliza para decidir la acción óptima bajo incertidumbre:

$$EU(a) = \sum_i P(s_i \mid a) \cdot U(a, s_i)$$

- **Ejemplo**: Decide si conviene actualizar los servidores o no:

Acción a	Estado s	Probabilidad $P(s \mid a)$	Utilidad $U(a,s)$
Actualizar	Seguro	0.8	50
Actualizar	Fallo	0.2	-80
No actualizar	Seguro	0.5	100
No actualizar	Vulnerable	0.5	0

La acción con mayor EU es la decisión óptima.

...en este ejemplo es no actualizar ($24 < 50$).

8.3. Modelos de decisión

8.3.1. Redes de decisión

- **Definición**: Las Redes de decisión son una extensión de las Redes Bayesianas que permiten modelar acciones, utilidades y probabilidades.

  [Judea Pearl, 1988]

- Tipos de nodos:
  1. Azar o Probabilidad: representan la incertidumbre, la suma de la probabilidad de los eventos posibles es 1.
  2. Decisión: Quien toma la decisión puede elegir entre varias acciones posibles.
  3. Valor o Utilidad: resultado final de la ruta de decisión (objetivo a maximizar).
- **Objetivo**: Determinar la acción que maximiza la utilidad esperada dada la evidencia disponible.

- **Ejemplo**: Dada estas CPT's, conviene llevar o dejar el paraguas:

Clima	Probabilidad
c	$P(c)$
Sol	0.7
Lluvia	0.3

Acción	Clima	Utilidad
a	c	$U(a,c)$
Dejar	Sol	100
Dejar	Lluvia	0
Llevar	Sol	20
Llevar	Lluvia	70

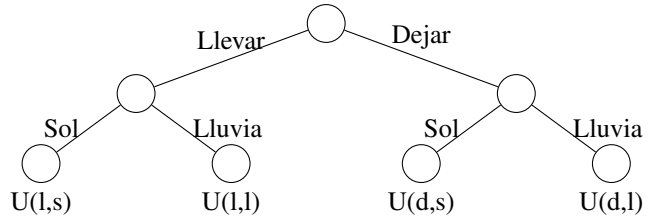
- **Procedimiento** para elegir la decisión óptima:
  1. Instanciamos las evidencias conocidas.
  2. Probamos cada acción como opción.
  3. Calculamos la utilidad esperada de cada acción (con la probabilidad posterior).
  4. Elegimos la acción con mayor utilidad.

- **Ejemplo (cont.)**: La decisión óptima es dejar el paraguas, EU(Dejar) $>$ EU(Llevar):

$$EU(\text{Dejar}) = 0{,}7 \cdot 100 + 0{,}3 \cdot 0 = 70$$
$$EU(\text{Llevar}) = 0{,}7 \cdot 20 + 0{,}3 \cdot 70 = 35$$

8.3.2. Árboles de Decisión

- **Definición** En los árboles de decisión, cada nodo representa un estado del mundo, las ramas representan acciones posibles, y las hojas representan la utilidad.
- Objetivo: Visualizar la toma de decisiones secuenciales.
- **Ejemplo (cont.)**: crea el árbol de decisión del problema del paraguas:



Se calcula la utilidad esperada y se elige la **rama óptima**.[1]

8.3.3. Secuencias y procesos de Markov

- **Definición**: Un Procesos de Decisión de Markov (MDP) es un modelo para tomar decisiones secuenciales cuando los resultados dependen de varios pasos en el tiempo:
  - Estados $S = \{s_0, s_1, ..., s_n\}$
  - Acciones $A = \{a_0, a_1, ..., a_m\}$
  - Transición de estados $P(s' \mid s, a)$
  - Recompensas/utilidad $R(s, a, s')$
- **Ejemplo:** Decidir como acelerar depende de la temperatura del motor.

E. actual s	Acción a	E. siguiente s'	Recompensa $R(s,a,s')$
Frío	Fuerte	Caliente	2
Caliente	Fuerte	Sobrecalentado	-10
Frío	Suave	Frío	1
...			

- Conceptos básicos: Para elegir la mejor acción en cada estado, usamos la política, y para evaluar sus efectos los valores de estado y de estado-acción.
  - Política π: asigna una acción a cada estado. $\pi : S \to A$
    - La política óptima maximiza la utilidad esperada a largo plazo.
  - Valor de un estado: utilidad esperada empezando en s

    $V^*(s) = \max_a Q^*(s,a)$

[1] Calculo parecido al ExpectiMinimax (ver Tema 3), hay otras estrategias como MDPs, RL, etc.

- Valor de un estado-acción: utilidad esperada con acción a desde s

$$Q^*(s,a) = \sum_{s'} P(s' \mid s,a)\Big(R(s,a,s') + \gamma \cdot V_k(s')\Big) \qquad \gamma \leq 1$$

- **Procedimiento**: La resolución básica consiste en:
  1. Inicializar $V_0(s) = 0$
  2. Repetir hasta convergencia: $V_{k+1}(s) = \max_a Q^*(s,a)$

- **Ejemplo (cont.)**: Dado el grafo de la Figura 8.1 que modela el problema de como acelerar con un MDP (con $\gamma = 1$):

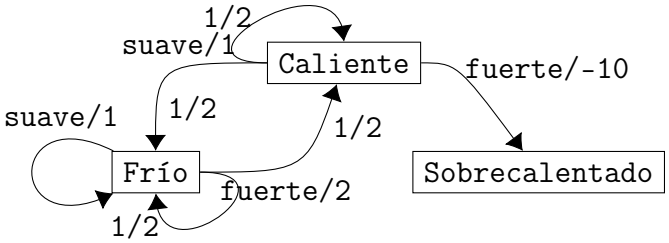


Figura 8.1: Conviene acelerar fuerte o suave.

a) Calcular V* (y Q*) de forma iterativa.[2]

$V_0(F) = 0 \quad V_0(C) = 0 \quad V_0(S) = 0$

$Q_1(C,f) = -10$

$Q_1(C,s) = \frac{1}{2}(1+0) + \frac{1}{2}(1+0) = 1$

$Q_1(F,f) = \frac{1}{2}(2+0) + \frac{1}{2}(2+0) = 2$

$Q_1(F,s) = 1+0 = 1$

$Q_1(S,\_) = 0$ (nodo terminal $V_x(S) = 0$)

$V_1(F) = 2 \quad V_1(C) = 1$

$Q_2(C,f) = -10$

$Q_2(C,s) = \frac{1}{2}(1+2) + \frac{1}{2}(1+1) = 2{,}5$

$Q_2(F,f) = \frac{1}{2}(2+1) + \frac{1}{2}(2+2) = 3{,}5$

$Q_2(F,s) = 1+2 = 3$

$V_2(F) = 3{,}5 \quad V_2(C) = 2{,}5$

$V_3(F) = 5 \quad V_3(C) = 4{,}25$ ...

- **Ejemplo (cont.)**

b) La iteración de valores no converge porque, con $\gamma = 1$, los ciclos con recompensas positivas hacen que $V(s)$ y $Q(s,a)$ crezcan indefinidamente.

c) Aún así tenemos política óptima:

$$\pi^*(C) = s \quad \pi^*(F) = f$$

[2]Notación: $s \in$ {F=Frio, C=Caliente, S=Sobrecalentado}; $a \in$ {f=fuerte, s=suave}

NOTA: MDP calcula políticas completas,
ExpectiMinimax solo calcula la acción para un único estado...

...pero ¿si no conocemos $P(s' \mid s,a)$ ni $R(s,a,s')$?[3]

Ejercicio 26 Dado el grafo de la Figura 8.1, que modela el problema de como acelerar con un MDP, calcular V* (y Q*) de forma iterativa con $\gamma = 0{,}5$.

[3]En el Tema 11 veremos Aprendizaje por Refuerzo para aprender políticas óptimas interando con el entorno.

Módulo IV

Aprendizaje automático

Universidad Rey Juan Carlos

Tema 9

Aprendizaje probabilístico

9.1 Aprendizaje automático **59**
- 9.1.1 Introducción 59
- 9.1.2 Modelos generativos y discriminativos 60
- 9.1.3 Clasificación probabilística 60

9.2 Naive Bayes **61**
- 9.2.1 Algoritmo 61
- 9.2.2 'Implementación' en Python 62

9.3 Evaluación de modelos **63**

9.1. Aprendizaje automático

9.1.1. Introducción

- **Definición**: Rama de la inteligencia artificial que permite a las máquinas aprender de los datos sin ser explícitamente programadas.
  - Se basa en modelos matemáticos y estadísticos para capturar relaciones en los datos.
- Objetivo principal: hacer predicciones o tomar decisiones basadas en patrones aprendidos.
- Categorías de aprendizaje automático:
  - Aprendizaje Supervisado: Datos con etiquetas conocidas.

    Ejemplo: Clasificación de correos como spam/no spam.
  - Aprendizaje No Supervisado: Datos sin etiquetas, se buscan patrones o agrupamientos. Ejemplo: Segmentación de malware.

  - Aprendizaje por Refuerzo: Agente aprende mediante prueba y error con recompensas. Ejemplo: Malware que encuentra vulnerabilidades.

- Componentes Clave:
  - **Datos:** Información histórica o de entrenamiento.
  - **Modelo:** Representación matemática de patrones en los datos.
  - **Algoritmo de Aprendizaje:** Procedimiento para ajustar el modelo a los datos.
  - **Evaluación:** Medición de desempeño del modelo (precisión, error, F1-score, etc.).
- Ventajas del Aprendizaje Automático:
  - Capacidad de automatizar tareas complejas.
  - Adaptación a cambios en los datos.
  - Identificación de patrones no evidentes para humanos.

9.1.2. Modelos generativos y discriminativos

- **Modelos Generativos**: Intentan modelar la distribución conjunta $P(X,Y)$, donde X son características y Y etiquetas.
  - Ejemplo: Naive Bayes, Modelos de mezcla de Gaussianas (GMM).
  - Ventaja: Se pueden generar nuevos datos y manejar datos faltantes.
- **Modelos Discriminativos**: Modelan directamente $P(Y|X)$, la probabilidad de la etiqueta dada la entrada.
  - Ejemplo: Regresión logística, Redes Bayesianas discriminativas.
  - Ventaja: Normalmente mejor rendimiento predictivo cuando no interesa modelar los datos de forma completa.

9.1.3. Clasificación probabilística

- **Definición:** Asignar a cada instancia x la clase y con la mayor probabilidad condicional dado x:

$$\hat{y} = \arg\max_{y \in Y} P(Y = y \mid X = x)$$

- Ejemplo: Naive Bayes.[1]

$$P(Y \mid X_1, X_2, \ldots, X_n) \propto P(Y) \prod_{i=1}^{n} P(X_i \mid Y)$$

[1] Usamos $\propto$ porque conserva el orden entre clases de la probabilidad posterior, aunque no esté normalizada.

- Supone independencia condicional de las características X_i.
- La clase con mayor $P(Y|X)$ se asigna como predicción.

Relación con temas anteriores:

- **Representación probabilística:** Se basa en probabilidades condicionales y la regla de Bayes.
- **Redes Bayesianas:** Pueden usarse como modelo para representar dependencias entre variables.
- **Inferencia:** Se realiza inferencia para calcular $P(Y|X)$ a partir del modelo.
- **Teoría de la decisión:** La clase con mayor probabilidad puede considerarse la acción óptima bajo pérdida cero-uno.

9.2. Naive Bayes

9.2.1. Algoritmo

- **Algoritmo Naive Bayes**:
  1. Convierte los datos en tabla de frecuencias.
  2. Crea tabla de probabilidades.
  3. Calcula probabilidad posterior de cada clase (sin normalizar).
  4. Predicción: La clase con la probabilidad posterior más alta.
- **Ejemplo**: Predecir si se juega al tenis cuando hace sol a partir de las observaciones de la Figura 9.1:

Clima	s	n	l	l	s	s	n	l	n	n
Juegan	–	+	+	–	+	+	+	–	+	+

Figura 9.1: Clima y partidos disputados (o no) en US Open.

1. Tabla de frecuencias

Clima	Juegan	
	si (+)	no (–)
Nublado (n)	4	0
Sol (s)	2	1
Lluvia (l)	1	2

2. Tabla de Probabilidades

Clima	Juegan		P(C)
	si	no	
Nublado	4	0	0.4
Sol	2	1	0.3
Lluvia	1	2	0.3
P(J)	0.7	0.3	

- **Ejemplo (cont.)**

3. Calcular P(si | Sol) y P(no | Sol), usando:

$$P(Y \mid X_1, X_2, \ldots, X_n) \propto P(Y) \prod_{i=1}^{n} P(X_i \mid Y)$$

Nota que con solo un X_i no es muy útil...

- $P(si \mid Sol) \propto P(si) \cdot P(Sol \mid si) =$

$$= \frac{\#si}{\#total} \cdot \frac{\#\{si \wedge sol\}}{\#si} = \frac{7}{10} \cdot \frac{2}{7} = 0{,}2$$

- $P(no \mid Sol) \propto P(no) \cdot P(Sol \mid no) =$

$$= \frac{\#no}{\#total} \cdot \frac{\#\{no \wedge sol\}}{\#no} = \frac{3}{10} \cdot \frac{1}{3} = 0{,}1$$

4. Predicción: SI, cuando hace sol se juega.

$$\text{P(si} \mid \text{Sol)} > \text{P(no} \mid \text{Sol)}$$

Ejercicio 27 Usando los datos de la Figura 9.1, estimar con Naive Bayes la probabilidad de jugar al tenis cuando llueve.

Ejercicio 28 Explica por qué Naive Bayes no aporta ventajas con un solo atributo.

Ejercicio 29 Razona por qué no es necesario normalizar la probabilidad posterior.

Ejercicio 30 Justifica por qué Naive Bayes es más útil con varios atributos.

9.2.2. 'Implementación' en Python

```
from sklearn.naive_bayes import CategoricalNB
# Datos del problema (mapeado a números
Atributos = [[2],[1],[0],[0],[2],[2],[1],[0],[1],[1]]
Clases    = [ 0 , 1 , 1 , 0 , 1 , 1 , 1 , 0 , 1 , 1 ]
# Aprendizaje usando Naive Bayes
modelo_nb = CategoricalNB()
modelo_nb.fit(Atributos, Clases)
# Inferencia ¿Se juegan si hace sol? ¿Y si llueve?
```

```
Preguntas    = [[2],[0]]
Predicciones = modelo_nb.predict(Preguntas)
# Imprime resultados
for pregunta, prediccion in zip(Preguntas, Predicciones):
    print(f"Clima: {pregunta[0]} -> ¿Se juega?: {prediccion}")
```

Ejercicio 31 Predecir usando el código si se juega con clima nublado.

Ejercicio 32 Calcula e imprime la probabilidad de la predicción.[2]

- Código en Python del problema del tenis de la Figura 9.1:
  - Los datos deben mapearse a valores numéricos positivos.
  - `Atributos` y `Preguntas` son listas de observaciones, donde cada es una lista de atributo(s).
  - `pregunta[0]` es el primer atributo de la observación.
  - Try it in `https://paiza.io`.

9.3. Evaluación de modelos

- Tras entrenar un modelo probabilístico, es crucial medir su desempeño.
- La evaluación permite:
  - Comparar modelos.
  - Detectar sobreajuste (*overfitting*).
  - Ajustar parámetros y seleccionar el modelo. óptimo
- Para problemas de clasificación probabilística se usan métricas clásicas:
  - **Exactitud (Accuracy)**: proporción de predicciones correctas.
  - **Precisión (Precision)**: proporción de positivos predichos correctos.
  - **Sensibilidad (Recall0**: proporción de positivos reales correctos.
  - **F1-Score**: media armónica de precisión y recall.
- Lo aplicaremos, p.ej., para evaluar Naive Bayes.

Matriz de Confusión:

- Herramienta visual:
  - Filas: clases reales.
  - Columnas: clases predichas
- Ejemplo: 2 clases (Positivo / Negativo)

[2] Consulta documentación en `https://scikit-learn.org`

$$\begin{bmatrix} \text{TP} & \text{FP} \\ \text{FN} & \text{TN} \end{bmatrix}$$

- TP: Verdaderos positivos
- TN: Verdaderos negativos
- FP: Falsos positivos
- FN: Falsos negativos

Ejercicio 33 Calcula las métricas de evaluación:

a) Accuracy.

b) Precision.

c) Recall.

d) F1.

Para el problema del tenis con los observaciones y predicciones de la Figura 9.2.

Clima	s	n	l	l	s	s	n	l	n	n
Juegan	–	+	+	–	+	+	+	–	+	+
Predicción	+	+	–	–	+	+	+	–	+	+

Figura 9.2: Clima, partidos disputados (o no) y predicción.

```
# (cont.) implementación Naive Bayes
from sklearn import metrics
import matplotlib.pyplot as plt
# Evaluación usando los mismos datos, NO HACER!
P_todo = modelo_nb.predict(Atributos)
# Calculamos distintas métricas
accuracy = metrics.accuracy_score(Clases, P_todo)
precision = metrics.precision_score(Clases, P_todo)
recall = metrics.recall_score(Clases, P_todo)
f1 = metrics.f1_score(Clases, P_todo)
matrix = metrics.confusion_matrix(Clases, P_todo)
# Imprimimos los resultados
print(f"\nExactitud (Accuracy): {accuracy:.2f}")
print(f"Precisión (Precision): {precision:.2f}")
print(f"Sensibilidad (Recall): {recall:.2f}")
print(f"F1-Score: {f1:.2f}")
print(f"\nMatriz de Confusión:\n {matrix}")
```

- Añadir al código en Python (pág. 9.2.2) del problema del tenis de la Figura 9.1.
  - Try it in `https://paiza.io`.

- NOTA: Los datos de Entrenamiento y Prueba no deben coincidir... (pág. 10.4)

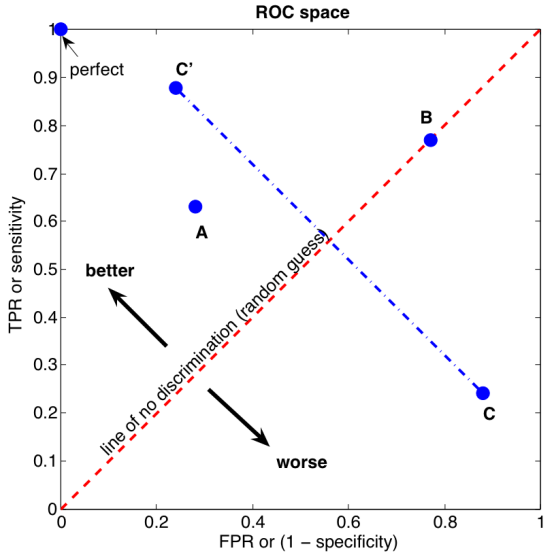
Ejercicio 34 Comprobar que los resultados coinciden con los calculados en el **Ejercicio 33**
.

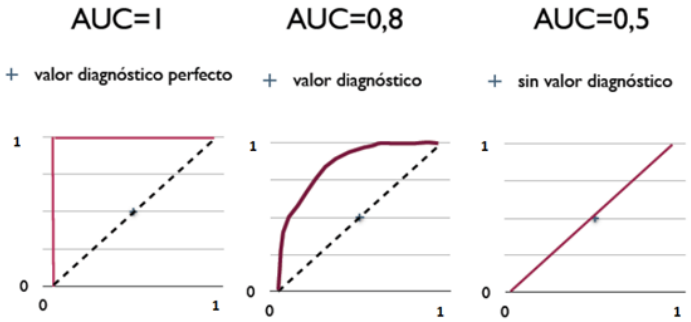
Curvas ROC y AUC

- Para clasificadores probabilísticos, la probabilidad predicha permite variar el umbral.
- **Curva ROC**:

  (Receiver Operating Characteristic)

  - Eje X: Tasa de falsos positivos (FPR)
  - Eje Y: Tasa de verdaderos positivos (TPR)
- **AUC**: (Area Under the Curve)
  - Medida global de desempeño
  - AUC = 1: perfecto.
  - AUC = 0.5: aleatorio.



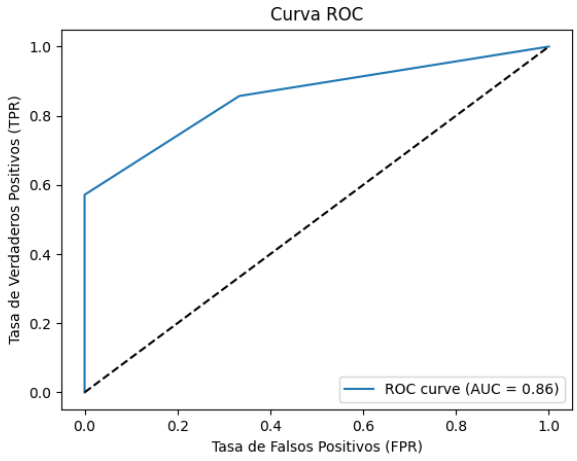


```
# (cont.) implementación Naive Bayes + metrics
import matplotlib.pyplot as plt
# Necesitamos probabilidades predichas
probabilidades = modelo_nb.predict_proba(Atributos)[:,1]
fpr, tpr, thresholds = roc_curve(Clases, probabilidades)
auc = roc_auc_score(Clases, probabilidades)
# Imprime AUC
print(f"\nAUC: {auc:.2f}")
# Dibujar curva ROC
plt.figure()
plt.plot(fpr, tpr, label=f'ROC curve (AUC = {auc:.2f})')
plt.plot([0,1], [0,1], 'k--')  # diagonal de referencia
plt.xlabel('Tasa de Falsos Positivos (FPR)')
plt.ylabel('Tasa de Verdaderos Positivos (TPR)')
plt.title('Curva ROC')
plt.legend(loc='lower right')
plt.show()
```

- Añadir al código en Python (págs. 9.2.2 y 9.3) del problema del tenis de la Figura 9.1.

`[:,1]` Índice de la probabilidad de la clase positiva.

- Try it in https://pythononline.net/



Universidad Rey Juan Carlos

Tema 10

Aprendizaje inductivo

10.1 Fundamentos del aprendizaje inductivo **67**

10.2 Árboles de decisión **68**

10.2.1 Fundamentos 68

10.2.2 Algoritmo ID3 68

10.3 Sobreajuste y generalización **72**

10.4 Validación cruzada **72**

10.1. Fundamentos del aprendizaje inductivo

- **Definición**: El aprendizaje inductivo es el proceso mediante el cual un sistema aprende una hipótesis general a partir de ejemplos particulares.
  - Se parte de un conjunto de datos etiquetados.
  - Se infiere una función h que aproxima una función objetivo f:
    $$f : \vec{X} \to Y.$$
  - El objetivo es generalizar correctamente a nuevos casos no observados.
- **Esquema formal**:
  $$D = \{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)\}$$
  Aprender una hipótesis $h \in H$ tal que:
  $$h(\vec{x}) \approx y$$

10.2. Árboles de decisión

10.2.1. Fundamentos

- **Definición**: Modelo predictivo que utiliza una estructura jerárquica de decisiones para clasificar o predecir valores:
  - Cada nodo interno representa una prueba sobre un atributo.
  - Cada rama representa el resultado de la prueba.
  - Cada hoja representa una clase o valor predicho.
- **Criterios de división**:
  - Entropía e Información Ganada. (ID3, C4.5)
  - Índice Gini. (CART)
- **Ventajas:**
  - Interpretables.
  - Manejan datos categóricos y numéricos.
  - No requieren supuestos fuertes sobre la distribución.

10.2.2. Algoritmo ID3

```
ALGORITMO ID3(Nodo, Ejemplos, Atributos)
   SI termina(Nodo, Ejemplos, Atributos) ENTONCES
       crea_Hoja(Nodo)
   SINO
       Aj ← mejorAtributo(Ejemplos, Atributos)
       Etiqueta(Nodo) ← Aj

       PARA CADA v ∈ Dominio(Aj) HACER
           Hijo ← crea_Hijo(Nodo)
           crea_Arista(<Nodo,v,Hijo>)
           Es_v ← { e ∈ Ejemplos | Aj(e) = v }
           ID3(Hijo, Es_v, Atributos \ {Aj} )
```

- Algoritmo depende de dos funciones/preguntas clave:
  `mejorAtributo/2`: ¿Cómo seleccionar el mejor atributo?
  `termina/3`: ¿Cuándo determinar si la rama debe terminar?

- Construir recursivamente el árbol desde el nodo raíz.

a) Si el nodo es terminal:
   - Etiquetarlo como hoja.

b) Si no:

1. Seleccionar el atributo más relevante `Aj` y etiquetar el nodo.
2. Crear un hijo para cada valor `v` en el dominio de `Aj`.
3. Para cada hijo, construir `Es_v` con los ejemplos del nodo con `v` en `Aj`.
4. Repite el proceso para todos los hijos.

- **Definición**: El mejor atributo es aquel que:
  - Agrupa mejor los ejemplos según la clase que queremos aprender.
  - Reduce el "desorden" de los posibles valores de clase.

- **Ejemplo**: Decidir si el banco acepta la solicitud de tarjeta de crédito.

	edad	saldo	Tarjeta
E1:	< 18	bajo	No
E2:	< 18	alto	No
E3:	18-25	medio	Si
E4:	18-25	alto	Si
E5:	18-25	medio	No

a) Para `saldo` tenemos:
   - alto $\rightarrow$ {E2, E4} {No, Si}
   - medio $\rightarrow$ {E3, E5} {Si, No}
   - bajo $\rightarrow$ {E1} {No}

b) Para `edad` tenemos:
   - $< 18 \rightarrow$ {E1, E2} {No, No}
   - medio $\rightarrow$ {E3, E4, E5} {Si, Si, No}

A ojo parece que `edad` clasifica mejor.

```
FUNCION mejorAtributo(Ejemplos, Atributos):
   # Calcular entropía global:
   E_global ← −∑_c p_c(Ejemplos) log_2 p_c(Ejemplos)
   # Calcular ganancia de cada atributo Aj:
   PARA CADA Aj ∈ Atributos HACER
       PARA CADA v ∈ Dominio(Aj) HACER
           S_v ← { e ∈ Ejemplos | Aj(e) = v }
           E_Aj_v ← −∑_c p_c(S_v) log_2 p_c(S_v)
       E_Aj ← ∑_v p_{S_v}(Ejemplos) E_Aj_v
       Ganancia(Aj) ← E_global - E_Aj
   # Devuelve el Atributo con mayor ganancia
```

DEVUELVE $\underset{Aj \in Atributos}{\arg\max}$ Ganancia(Aj)

- Elegimos el atributo con mayor Ganancia, basada en la Entropía (desorden) global/local. [Shannon, 1948]
  - Sea $p_v(C)$ la probabilidad de tener v en C:

$$E(C) = -\sum_v p_v(C) \log_2 p_v(C)$$

  - **Ejemplo (cont.)**

    E_global $= -\sum_{v \in \{Si,No\}} p_v(C) \cdot log_2\ p_v(C) =$

    $$= -\left(\tfrac{2}{5} \cdot log_2\ \tfrac{2}{5} + \tfrac{3}{5} \cdot log_2\ \tfrac{3}{5}\right) = 0,97$$

    Ganancia de saldo y edad:

    E_saldo_alto = ... = 1

    E_saldo_medio = ... = 1

    E_saldo_bajo = ... = 0

    E_saldo $= \frac{2}{5} \cdot 1 + \frac{2}{5} \cdot 1 + \frac{1}{5} \cdot 0 = 0,8$

    Ganancia(saldo) = 0,97 - 0,8 = 0,17

    E_edad_< 18 = ... = 0

    E_edad_18-25 = ... = 0,91

    E_edad $= \frac{2}{5} \cdot 0 + \frac{3}{5} \cdot 0,91 = 0,55$

    Ganancia(edad) = 0,97 - 0,55 = **0,42**

    Efectivamente edad clasifica mejor.

```
FUNCION termina(Nodo, Ejemplos, Atributos):
   # No hay ejemplos
   SI Ejemplos = ∅ ENTONCES
       Etiqueta(Nodo) ← mejorClase(Padre(Nodo))
       DEVUELVE True
   # Todos los ejemplos son de la misma clase
   PARA CADA c ∈ Clases HACER
       SI p_c(Ejemplos) = 1 ENTONCES
           Etiqueta(Nodo) ← c
           DEVUELVE True
   # No quedan atributos
   SI Atributos = ∅ ENTONCES
       Etiqueta(Nodo) ← mejorClase(Ejemplos)
       DEVUELVE True
   # En cualquier otro caso no es terminal
   DEVUELVE False
```

  - Una rama debe terminar (devuelve True) si:
    - No hay ejemplos para clasificar.

- Todos los ejemplos son de la misma clase.
- o No hay atributos con los que clasificar.

`mejorClase(S)` devuelve:

- La clase mayoritaria de `S`.
- o Una clase aleatoria (o aleatoria ponderada).

`Padre(N)` devuelve el padre del nodo `N`.

Heurísticas Detener expansión cuando:

- Hay una clase claramente mayoritaria.
- No se puede mejorar la clasificación, i.e.:

$$\forall Aj,\ Ganancia(Aj) \leq 0$$

Ejercicio 35 Adapta el algoritmo `ID3` para detener la expansión (early stopping), aplicando las siguientes heurísticas:

a) La clase mayoritaria supera el umbral del 90 % de los ejemplos.

b) No se puede mejorar la clasificación, *Ganancia* ≤ 0.

Ejercicio 36 Aplique el algoritmo de aprendizaje de árboles de decisión `ID3` al siguiente conjunto de ejemplos. Especifique el árbol aprendido y los valores de ganancia de los atributos elegidos en los diferentes nodos.

edad	saldo	Tarjeta
< 18	bajo	No
< 18	alto	No
18-25	medio	Si
18-25	alto	Si
18-25	medio	No

Ejercicio 37 Aplique el algoritmo de aprendizaje de árboles de decisión `ID3` al siguiente conjunto de ejemplos. Especifique el árbol aprendido y los valores de ganancia de los atributos elegidos en los diferentes nodos.

Pelo	Altura	Peso	Protección	Quemadura
rubio	media	bajo	no	si
rubio	alta	medio	si	no
moreno	baja	medio	si	no
rubio	baja	medio	no	si
rojo	media	alto	no	si
moreno	alta	alto	no	no
moreno	media	alto	no	no
rubio	baja	bajo	si	no

10.3. Sobreajuste y generalización

- **Definición**: Sobreajuste (overfitting) ocurre cuando el modelo se ajusta excesivamente a los datos de entrenamiento, capturando ruido en lugar de patrones generales.
  - Error bajo en entrenamiento.
  - Error alto en datos nuevos.
- **Definición**: La generalización es la capacidad del modelo para funcionar correctamente en datos no vistos.
- Técnicas para evitar sobreajuste (sin perder generalidad):
  - Poda de árboles.
  - Regularización.
  - Aumento de datos.
  - Validación cruzada.

10.4. Validación cruzada

- **Definición**: La Validación Cruzada (Cross-Validation, CV) es una técnica para estimar el rendimiento real de un modelo en datos no vistos:
  - Se divide el conjunto de datos en k particiones (folds).
  - Se entrena con $k-1$ particiones.
  - Se evalúa con la partición restante.
  - Se repite el proceso k veces.
- Cálculo del error promedio en validación cruzada k-fold:

$$Error_{CV} = \frac{1}{k} \sum_{i=1}^{k} Error_i$$

- Ventajas:
  - Uso eficiente de los datos.
  - Reduce la varianza en la estimación del error.

Universidad Rey Juan Carlos

Tema 11

Aprendizaje por refuerzo

11.1 Fundamentos **73**
- 11.1.1 Definición y formalización básica 73
- 11.1.2 Interacción agente-entorno 74
- 11.1.3 Motivación, aprendizaje sin modelo 74

11.2 Resolución basada en modelo **75**
- 11.2.1 El entorno como MDP 75
- 11.2.2 Algoritmo de iteración de valores 75

11.3 Aprendizaje sin modelo **77**
- 11.3.1 Fundamentos y Q-learning 77
- 11.3.2 Q-learning para entornos deterministas 78
- 11.3.3 Q-learning para entornos no deterministas 80

11.1. Fundamentos

11.1.1. Definición y formalización básica

- **Definición**: El aprendizaje por refuerzo (Reinforcement Learning, RL) es un paradigma del aprendizaje automático en el que un agente:
  - Aprende a tomar decisiones mediante prueba y error.
  - Interactúa con el entorno.
  - Trata de maximizar la recompensa acumulada.
- No hay respuestas correctas directas. El agente descubre la mejor estrategia explorando y recibiendo retroalimentación de sus acciones.
- **Formalización**: Se basa en un agente que:

- Percibe el estado del entorno: $s \in S$.
- Toma acciones en un estado s: $a \in A(s)$
- Recibe recompensas al tomar una acción en un estado. $r(s,a)$
- Aprende una política óptima, la estrategia que maximiza la recompensa total. $\pi(s)$

11.1.2. Interacción agente-entorno

- En el aprendizaje por refuerzo, la interacción entre agente y entorno se describe como un ciclo de interacción:
  1. El agente observa el estado (s_t) del entorno.
  2. Elige una acción (a_t) según su política (π).
  3. El entorno responde proporcionando:
     - Un nuevo estado (s_{t+1})
     - Una recompensa (r_t)
  4. El agente actualiza su política usando esta experiencia.
- **Ejemplo**: Un robot limpiador en una habitación:
  - Estado (s_t): posición actual y suciedad de la habitación.
  - Acción (a_t): moverse, limpiar, girar.
  - Recompensa (r_t): +1 por limpiar, -1 por chocar con un obstáculo.
  - Objetivo: maximizar la limpieza total sin dañar el entorno.

11.1.3. Motivación, aprendizaje sin modelo

- **Motivación**: El aprendizaje sin modelo es necesario porque no siempre conocemos el modelo del entorno:
  - Del conjunto de estados posibles S sólo los estados visitados.
  - El conjunto de acciones posibles A (y sus relaciones con los estados (s,a) solo es conocido parcialmente.
  - El modelo de transición $T : S \times A \to Pr(S)$ es desconocido.
  - El modelo de recompensas $R : S \times A \to \mathbb{R}$ es desconocido.
- Ejemplos:
  - Robot que debe moverse en en un almacén.
  - Laberinto.
  - Jugar al ajedrez.
- **Objetivo:** Aprender la política que maximiza las recompensas acumuladas esperadas (política óptima $\pi* : S \to A$)

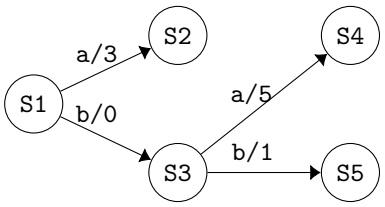
11.2. Resolución basada en modelo

11.2.1. El entorno como MDP

- **Definición**: Cuando se conoce el entorno, este se puede modelar como un Proceso de Decisión de Markov (MDP).[1]
- Un MDP está definido por una tupla $\langle S, A, T, R \rangle$:
  - S es el conjunto de estados.
  - A es el conjunto de posibles acciones.
  - $T : S \times A \times S \to [0,1]$ es un modelo de transición:
    - T: es una distribución de probabilidades sobre el conjunto S.
    - $T(s,a,s')$ (o $P(s' \mid s,a)$: denota la probabilidad de llegar al estado s' si se realiza la acción a en el estado s.
  - $R : S \times A \to \mathbb{R}$ es el modelo de recompensas.
    - $R(s,a)$ denota la recompensa que se recibe al realizar la acción a en el estado s.
- **Propiedad de Markov**:

  $$\Pr(s_{t+1}, r_t \mid s_t, a_t) = \Pr(s_{t+1}, r_t \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0)$$

  La probabilidad de un estado y su recompensa depende solo del estado y acción previos, no del historial completo.
- **Ejemplo**: Dado el siguiente grafo resolver el MDP, calculando Q*, V* y π*.[2] Al ser un grafo acíclico su resolución es sencilla.



$Q^*(s_3,a) = 5+0 = 5$ $\quad$ $Q^*(s_1,a) = 3+0 = 3$

$Q^*(s_3,b) = 1+0 = 1$ $\quad$ $Q^*(s_1,b) = 0+\text{máx}(1,5) = 5$

$V(s_3) = \text{máx}(5,1) = 5$ $\quad$ $V(s_1) = \text{máx}(3,5) = 5$

$\pi(s_3) = a$ $\quad$ $\pi(s_1) = b$

11.2.2. Algoritmo de iteración de valores

```
ALGORITMO Iteracion_Valores
   PARA CADA s ∈ S y a ∈ A(s) HACER
       Q*(s,a) ← 0   # inicialización
```

[1] Véase el Tema 8 para una introducción previa a los MDP.

[2] En S4 y S5: Q* y V* son 0 (y no tienen política) porque no hay acciones saliendo de ellos.

```
REPETIR
    cambio ← 0
    PARA CADA s ∈ S y a ∈ A(s) HACER
        old ← Q*(s,a)
        Q*(s,a) ← R(s,a) +
              + γ ∑_s' T(s,a,s') · max_b Q*(s',b)
        cambio ← max(cambio, |old - Q*(s,a)|)
HASTA QUE cambio < ε

PARA CADA s ∈ S HACER
    V*(s) ← max_{a∈A} Q*(s,a)
    π*(s) ← argmax_{a∈A} Q*(s,a)

DEVUELVE Q*, V*, π*
```

- Pseudocódigo del algoritmo de iteración de valores para `V*`, `Q*` y $\pi*$.
  - Permite calcular grafos con ciclos.

  `Q*` se suele inicializar en 0 (pero no solo).

  $|x|$ devuelve el valor absoluto de x.

  ε valor positivo, define el criterio de parada.

- Hay otros algoritmos para resolver MDPs, p.ej., Iteración de políticas.
- Hay extensiones, p.ej., para entornos con recompensas estocásticas (como el Poker):
  - POMDP: Partially Observable MDP.

- **Ejemplo**: Resolver el grafo de la Figura 11.1 con `Iteracion_Valores`:
  - Inicializando Q* a 0.
  - Orden de ejecución: Estados de s_0 a s_4 y acciones a; b.
  - Factor de descuento $\gamma = 1$.

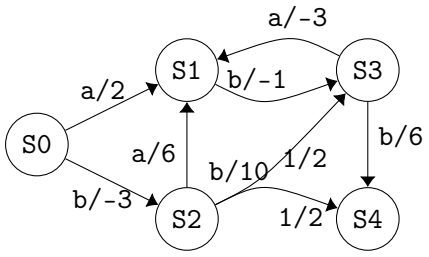


Figura 11.1: Grafo no determinista.

It.	$Q^*(s_0,a)$	$Q^*(s_0,b)$	$Q^*(s_1,b)$	$Q^*(s_2,a)$	$Q^*(s_2,b)$	$Q^*(s_3,a)$	$Q^*(s_3,b)$
	0	0	0	0	0	0	0
1	2	-3	-1	5	10	-4	6
2	1	7	5	11	13	2	6
3	7	10	5	11	13	2	6
4	7	10	5	11	13	2	6

Estado	s_0	s_1	s_2	s_3	s_4
V^*	10	5	13	6	0
π^*	b	b	b	b	-

Ejercicio 38 Dado el grafo de la Figura 11.1, resolver con `Iteracion_Valores`, iniciando Q* a 0 y $\gamma = 0{,}9$.

11.3. Aprendizaje sin modelo

11.3.1. Fundamentos y Q-learning

- **Definición**: La idea básica del aprendizaje por refuerzo sin conocer el modelo del entorno consiste en:
  - Actuar por "prueba y error".
  - Mejorar la política según las recompensas recibidas.
- **Clasificación de los métodos:**
  1. Libres del modelo: Solo intentan aprenden la política, p.ej.:
     - Q-learning. [Watkins, 1989]
  2. Métodos basados en el modelo: Se aprenden también el modelo de transiciones y recompensas. Con esta información se actualizar la política aprendida.
- **Definición**: Q-learning aprende una función $Q(s,a)$ que aproxime $Q^*(s,a)$, sin conocer el modelo del entorno.
- La política del agente:

$$\pi(s) = \arg\max_a Q(s,a)$$

mejora mediante *prueba y error* a lo largo de muchos episodios.

- Es similar al algoritmo `Iteracion_Valores` para resolver MDPs, pero Q-learning no requiere la función de transición T ni de recompensas R.
- Veremos dos versiones de Q-learning, según el tipo de entorno:
  - Deterministas: Cada acción lleva a un único estado y recompensa.
  - No deterministas: Las acciones pueden producir varios estados y recompensas probabilísticas.

11.3.2. Q-learning para entornos deterministas

```
ALGORITMO Q_Learning_v1
   s ← observar(E_actual)
   IF s ∉ E_conocidos
      E_conocidos.añade(s)
      PARA CADA a ∈ A(s) HACER
          Q(s,a) ← 0

   MIENTRAS !terminal(s) HACER
       a ← siguiente_accion(s)
       ejecutar(a)
       s' ← observar(E_siguiente)
       r ← observar(R(s,a,s'))
       SI s' ∉ E_conocidos ENTONCES
           E_conocidos.añade(s')
           PARA CADA b ∈ A(s') HACER
               Q(s',b) ← 0
       Q(s,a) ← r + γ·máx_b Q(s',b)
       s ← s'
```

- Pseudocódigo para algoritmo Q-learning en entornos deterministas.

  `Q` Inicializar (i) de manera pesimista, p.ej., 0 o (ii) con valores mayores a las recompensas esperadas.

  `siguiente_accion` Equilibrar la **exploración** de nuevas acciones con la **explotación** de las mejores acciones encontradas. Algunas estrategias son:

  ***greedy*:** en cada estado elige la mejor acción conocida (si hay empate, aleatoriamente).

  ***ε-greedy*:** la mejor acción con probabilidad ε o aleatoriamente.

  ***SOFTMAX*:** explora más al principio y explota más al final del aprendizaje.

- **Ejemplo**: Dado el grafo de la Figura 11.2, resolver aplicando `Q_Learning_v1`:

Opción A:

- El agente esta en s_0 y sabe $Q(s_0,a) = 2$, $Q(s_0,b) = 0$, $Q(s_1,b) = -1$.
- Factor de descuento $\gamma = 0{,}9$.

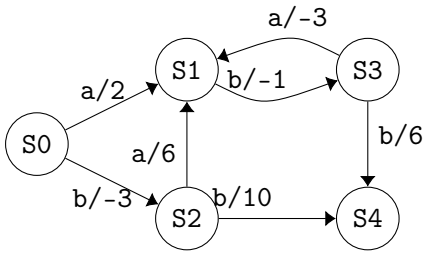
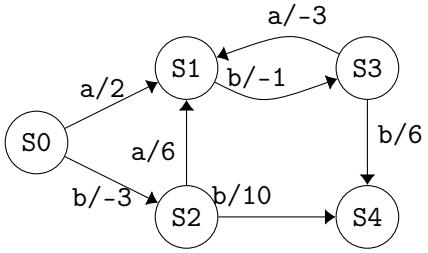


Figura 11.2: Grafo determinista

- En s_0 el agente conoce las acciones a y b:
  - Supongamos que elige a y va al estado s_1.
  - Como conoce s_1 actualiza

    $$Q(s_0,a) = r + \gamma \cdot max_{\text{acc} \in \{b\}} Q(s_1, \text{acc}) =$$
    $$= 2 + 0{,}9 \cdot (-1) =$$
    $$= 1{,}1$$
- Ahora el agente esta en el estado s_1:
  - Solo tiene una acción posible, la b y va al estado s_3, desconocido.
  - Inicializa $Q(s_3,a)$ y $Q(s_3,b)$.
  - Actualiza $Q(s_1,b) = \ldots$

...y sigue iterando.



Opción B:

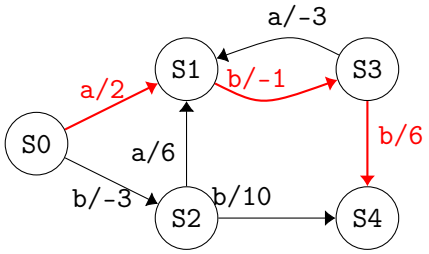
- El agente esta en s_0.
- $Q(s,a)$ se inician a 0.
- Selecciona acción con mayor valor (o acción “arriba”).
- Factor de descuento $\gamma = 1$.

It.	s	$A(s)$	s'	r	$Q(s_0,a)$	$Q(s_0,b)$	$Q(s_1,b)$	$Q(s_3,a)$	$Q(s_3,b)$
	s_0				0	0			
1	s_0	a	s_1	2	2		0		
2	s_1	b	s_3	-1			-1	0	0
3	s_3	a	s_1	-3				-4	
4	s_1	b	s_3	-1			-1		
5	s_3	b	s_4	6					6
	Valores finales				2	0	-1	-4	6

Segundo episodio desde s_0

It.	s	$A(s)$	s'	r	$Q(s_0,a)$	$Q(s_0,b)$	$Q(s_1,b)$	$Q(s_3,a)$	$Q(s_3,b)$
	s_0				2	0	-1	-4	6
1	s_0	a	s_1	2	1				
2	s_1	b	s_3	-1			-5		
3	s_3	b	s_4	6					6
	Valores finales				1	0	5	-4	6

Ejemplo (cont.): Tras varios episodios aprendemos la estrategia:



$$Q(s_0,a) = \mathbf{7}$$
$$Q(s_0,b) = 0$$
$$Q(s_1,b) = 5$$
$$Q(s_3,a) = 6$$
$$Q(s_3,b) = -4$$

- Pero no encuentra la óptima $s_0 \to s_2 \to s_1 \to s_3 \to s_4$ con recompensa 8.

Ejercicio 39 Aplicar el algoritmo `Q_Learning_v1` al grafo de la Figura 11.2 usando:

a) Distintas estrategias de selección de acciones, p.ej., ε-greedy.

b) Diferentes inicializaciones de Q, p.ej., $Q(s_0,a) = Q(s_0,b) = 20$.

11.3.3. Q-learning para entornos no deterministas

```
ALGORITMO Q_Learning_v2
   s ← observar(E_actual)
   IF s ∉ E_conocidos
```

```
    E_conocidos.añade(s)
    PARA CADA a ∈ A(s) HACER
        Q(s,a) ← 0

MIENTRAS !terminal(s) HACER
    a ← siguiente_accion(s)
    ejecutar(a)
    s' ← observar(E_siguiente)
    r ← observar(R(s,a,s'))
    SI s' ∉ E_conocidos ENTONCES
        E_conocidos.añade(s')
        PARA CADA b ∈ A(s') HACER
            Q(s',b) ← 0
    Q(s,a) ← (1-α)Q(s,a) +
              + α (r + γ·máx_b Q(s',b))
    s ← s'
```

- Pseudocódigo para algoritmo Q-learning en entornos NO deterministas.
- Al ejecutar una acción a en s, la transición puede ser estocástica, i.e., $Pr(s' \mid s,a) < 1$:
  - La actualización de $Q(s,a)$ debe incorporar solo parcialmente la información observada.

$\alpha \in (0,1)$ es la tasa de aprendizaje (learning rate):

- Si es grande acelera el aprendizaje pero puede desestabilizar.
- Normalmente pequeño para dar mas peso a la estimación anterior.

- Q-learning converge a $Q^*(s,a)$ si:
  1. El entorno se modela como un MDP.
  2. La tasa de aprendizaje α_i satisface:

$$\sum_{i=0}^{\infty} \alpha_i = \infty, \qquad \sum_{i=0}^{\infty} \alpha_i^2 < \infty,$$

     es decir, α_i decrece con el tiempo

     p.ej., $\alpha = 1/n(s,a)$.

  3. Cada par (s,a) se visita infinitas veces.
  4. El entorno es estacionario

     i.e., las $Pr(s' \mid s,a)$ no varían.

Ejercicio 40 Dado el grafo de la Figura 11.1, aplicar el algoritmo `Q_learning_v2` considerando:

- Iniciar $Q(s,a)$ en 0, estado inicial `S0`, $\gamma = 1$, $\alpha = 0{,}1$.
- Repetir 3 episodios, tal que en:

1 y 2: Elegir la acción con mayor Q o b.

3: Igual que antes, salvo que:

En S0 seleccionar acción b.

En S2, con b, la primera transición va S3, luego alterna S4 y S3.

a) Completar la tabla:

It.	s	$A(s)$	s'	r	$Q(s_0,a)$	$Q(s_0,b)$	...
	s_0	...					
1	...						

Bibliografía

Arias, Joaquín y Carro, Manuel (2019). **Incremental Evaluation of Lattice-Based Aggregates in Logic Programming Using Modular TCLP**. En: *21st Int'l. Symposium on Practical Aspects of Declarative Languages, PADL'19*. Vol. 11372. LNCS. Springer, págs. 98-114. DOI: 10.1007/978-3-030-05998-9_7.

Arias, Joaquín (2023). **Apuntes de Lógica: desde Aristóteles hasta Prolog**. Servicio de Publicaciones de la URJC. ISBN: 978-8409526345. URL: https://hdl.handle.net/10115/24277.

Arias, Joaquín (2024a). **Apuntes de Programación Lógica: hasta Prolog y más allá**. BURJC. URL: https://hdl.handle.net/10115/41640.

Arias, Joaquín (2024b). **Value Awareness Engineering**. Workshop on Ethical and Trustworthy AI. URL: https://shorturl.at/q2NYq.

Aristotle (-350). **Organon**. Original work in Greek, Ancient Greece.

Haugeland, John (1985). **Artificial intelligence: The very idea**. MIT press. ISBN: 978-0262081535.

Kurzweil, Ray (1990). **The age of intelligent machines**. MIT press. ISBN: 978-0262111218.

Lighthill, James (1973). **Artificial Intelligence: A General Survey**. En: *Artificial Intelligence: Paper Symposium*. Science Research Council, UK. URL: https://www.chilton-computing.org.uk/inf/literature/reports/lighthill_report/p001.htm.

Luger, G.F. y Stubblefield, W.A. (1993). **Artificial Intelligence: Structures and Strategies for Complex Problem Solving**. Benjamin/Cummings Publishing Company. ISBN: 9780805347807.

McCarthy, John, Minsky, Marvin, Rochester, Nathaniel y Shannon, C. E. (1955). **A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence**. Research Proposal. Considered the founding document of Artificial Intelligence research. Dartmouth College. URL: http://raysolomonoff.com/dartmouth/boxa/dart564props.pdf.

McCorduck, Pamela (1979). **Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence**. First edition. A. K. Peters. ISBN: 978-0716710721.

McCorduck, Pamela (1991). **Máquinas que piensan: Una incursión personal en la historia y perspectivas de la inteligencia artificial**. Editorial Tecnos. ISBN: 978-8430920167.

Minsky, Marvin y Papert, Seymour (1969). **Perceptrons: An introduction to computational geometry**. MIT Press. ISBN: 978-0262130431.

Moto-Oka, T. (2012). **Fifth Generation Computer Systems**. North Holland. ISBN: 978-0444600141.

Newell, Allen y Simon, Herbert A. (1961). **Computer simulation of human thinking**. En: *Science* 134(3495), págs. 2011-2017. DOI: 10.1126/science.134.3495.2011.

Nilsson, Nils J. (2001). **Inteligencia Artificial: Nueva Síntesis**. McGraw-Hill. ISBN: 978-8448128241.

Rumelhart, David E., Hinton, Geoffrey E. y Williams, Ronald J. (1986). **Learning representations by back-propagating errors**. En: *Nature* 323, págs. 533-536. DOI: 10.1038/323533a0.

Russell, Stuart J. y Norvig, Peter (2021). **Artificial Intelligence: A Modern Approach**. 4th. Pearson. ISBN: 978-1292401133.

Turing, Alan M. (1950). **Computing machinery and intelligence**. En: *Mind* 59(236), págs. 433-460. DOI: 10.1093/mind/LIX.236.433.

VaST (2025). **Los 10 científicos actuales MÁS IMPORTANTES del mundo**. Video en YouTube. URL: https://www.youtube.com/watch?v=i2buuqjkxrI.

Weise, Karen y Metz, Cade (2023). **When A.I. Chatbots Hallucinate**. En: *The New York Times*. URL: https://www.nytimes.com/2023/05/01/business/ai-chatbots-hallucination.html.



Universidad
Rey Juan Carlos

Escuela Técnica Superior
Ingeniería Informática

Módulo I: **Introducción a la Inteligencia Artificial**

Asignatura: Inteligencia Artificial

Joaquín Arias

Universidad Rey Juan Carlos
Escuela Técnica Superior de Ingeniería Informática

Curso 2025-2026

Índice I

Tema 1: **Introducción a la Inteligencia Artificial**
- **Definición y objetivos de la IA**
- **Historia y evolución de la Inteligencia Artificial**
- **Áreas principales de la IA por y para la Ciberseguridad**
  - IA y ciberseguridad: oportunidades y amenazas
- **Agentes inteligentes: tipos y entornos**
  - Aplicaciones (básicas) de IA
  - Ejercicios para casa I
  - Ejercicios para casa II



Copyright ©2026 Joaquín Arias. Esta obra, basada en trasparencias de Sascha, Holger y Alberto (URJC'23) & de Carmen y Paco (URJC'21), está bajo la licencia CC BY-SA 4.0, **Creative Commons Atribución-CompartirIgual 4.0 Internacional**. Cómo citar esta obra: Arias, Joaquín y Fidilio-Allende, Luciana (2026). ***LIBRO: Inteligencia Artificial, por y para la ciberseguridad***. BURJC, Madrid.

Tema 1: **Introducción a la Inteligencia Artificial**

Definición y objetivos de la IA [Russell y Norvig 2021]

Definición: La Inteligencia Artificial (IA) [McCarthy et al. 1955] es la disciplina que estudia y desarrolla sistemas capaces de realizar tareas que, tradicionalmente, requieren inteligencia humana, como razonar, aprender, tomar decisiones, percibir el entorno y adaptarse a él.

Desde la perspectiva de la ciberseguridad, la IA se concibe como:

- una herramienta para automatizar, mejorar y reforzar la detección, prevención y respuesta ante amenazas digitales.
- una amenaza capaz de adaptar ataques de forma autónoma, optimizar la evasión y escalar su impacto, aumentando la sofisticación y rapidez de las amenazas digitales.
- Ejercicio: Indica otras áreas o enfoques de la IA desde la perspectiva de la ciberseguridad.

Definición y objetivos de la IA (cont.)

Clasificación sistemas autónomos (IA Fuerte vs. IA Débil):

- Sistemas que piensan como humanos
  "La interesante tarea de lograr que las computadoras piensen... Máquinas con mente, en su amplio sentido literal" [Haugeland 1985]
- Sistemas que actúan como humanos
  "El arte de crear máquinas con capacidad de realizar funciones que realizadas por personas requieren inteligencia" [Kurzweil 1990]
- Sistemas que actúan de forma racional
  "La rama de la Informática que se ocupa de la automatización del comportamiento inteligente" [Luger y Stubblefield 1993]

Definición y objetivos de la IA (cont.)

Objetivos principales de la IA:

- Actuar como humanos:
  - Prueba de Turing: [Turing 1950]
- Pensar como humanos:
  - Modelado cognitivo: crear teoría acerca de la mente humana.
  - General Problem Solver (GPS): [Newell y Simon 1961]
- Actuar de forma racional:
  - Razonamiento irrefutable, inferencia lógica, programación lógica... [Arias 2023, 2024a; Aristotle -350]
  - Agentes racionales: actuar de forma "correcta/óptima/eficiente" en un entorno (con respecto a unos valores). [Arias 2024b]
- (Dejar de) alucinar: [Weise y Metz 2023]

Historia y evolución de la Inteligencia Artificial

La evolución de la IA está afectada por avances en distintas disciplinas:

- Filosofía (lógica, dualismo, materialismo, empirismo, positivismo, utilitarismo).
- Matemáticas (algoritmo, teorema de incompletitud, P vs. NP, teoría de la decisión).
- Psicología (comportamiento humano, conductistas, cognitiva).
- Lingüística, Sociología, Economía, Neurociencia, etc...

Orígenes (años 50–60)

- Término "Inteligencia Artificial" acuñado por McCarthy en 1956 [9].
- Primeros sistemas basados en lógica y reglas.
- Primeros modelos de neuronas artificiales (McCulloch/Pitts).
- Optimismo inicial sobre la rápida creación de máquinas inteligentes.

[McCorduck 1979, 1991]

Historia y evolución de la Inteligencia Artificial (cont.)

Sistemas expertos (años 60–75)

- Programas que replicaban el conocimiento de expertos humanos.
- Representación especializada del conocimiento:
  - Uso intensivo de reglas "si–entonces".
  - Reglas, marcos, guiones.
- Primeros sistemas expertos (Dendral, Prospector, Mycin).

Invierno de la IA (años 75-80)

- Motivada por el informe Lighthill en 1973 [7].
- Declive de la computación neuronal [Minsky y Papert 1969].
- Limitaciones en escalabilidad y mantenimiento.

Historia y evolución de la Inteligencia Artificial (cont.)

Proyecto FGCS (años 80-90)

- Proyecto de software de "quinta generación", Fifth Generation Computer Systems (FGCS) [Moto-Oka 2012].
  - Procesamiento en paralelo de gran cantidad de información.
  - Uso de lenguajes de programación lógica, (concurrent) Prolog, a niveles muy integrados con el hardware
    ...vs. enfoques basados en arquitectura Von Neumann.
- No logró el éxito comercial esperado y se "abandona" la IA simbólica.

Aprendizaje automático (años 90-2000)

- Enfoque en datos y estadística, regreso de las redes de neuronas:
  - Algoritmo de propagación hacía atrás. [Rumelhart et al. 1986]
- Modelos rigorosos y computacionalmente eficientes de manejo de incertidumbre (cadenas de Markov, redes Bayesianas).

Historia y evolución de la Inteligencia Artificial (cont.)

IA moderna y aprendizaje profundo (2010–actualidad)

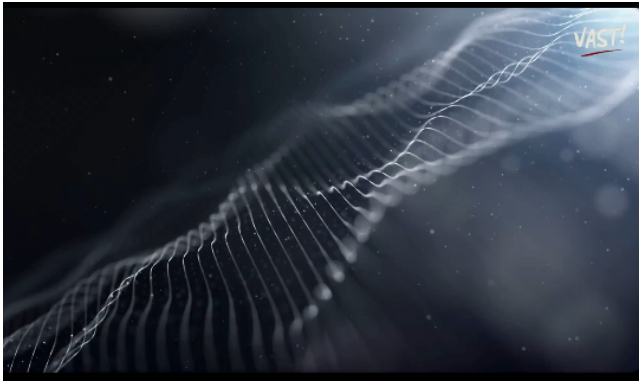
- Redes neuronales profundas (Deep Learning), múltiples capas y niveles de aprendizaje.
- Avances en visión artificial, procesamiento del lenguaje natural, ciberseguridad, sistemas autónomos / multiagente.
- Dependencia de grandes volúmenes de datos (Big Data) y potencia de cálculo para entrenamiento y/o inferencia (GPU's).
- Conceptos Trending Topic:
  - Modelos pre-entrenados (fine-tuning).
  - Aprendizaje reforzado (Reinforcement Learning, RL).
  - Computación distribuida (TensorFlow, PyTorch, JAX).
  - IA confiable (ética, seguridad, sesgos, explicabilidad)
  - Modelos multimodales (LLMs).
  - ...

Historia y evolución de la Inteligencia Artificial (cont.)

Nombre propios de la IA (hoy):

- Judea Pearl (7/10).
- Yann Lecun (4/10).
- Geoffrey Hinton (3/10).
- Demis Hassabis (1/10).

Selección extraída de [VaST 2025]...



Link

Áreas principales de la IA por y para la Ciberseguridad

- Aprendizaje automático (Machine Learning)
  Clasificación de malware, detección de intrusiones, análisis de tráfico.
- Aprendizaje profundo (Deep Learning)
  Detección avanzada de amenazas, análisis de comportamiento.
- Procesamiento del Lenguaje Natural (PLN)
  Análisis de correos de phishing, ingeniería social automatizada.
- Visión artificial
  Seguridad física, reconocimiento facial (con implicaciones éticas).
- Sistemas multiagente
  Simulación de ataques, defensa distribuida.

IA y ciberseguridad: oportunidades y amenazas

Oportunidades:

- Detección temprana de intrusiones.
- Análisis automático de vulnerabilidades.
- Respuesta rápida a incidentes.
- Reducción de falsos positivos.
- Defensa adaptativa frente a ataques cambiantes.

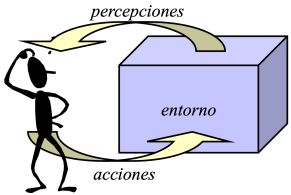
Amenazas:

- Uso de IA para generar malware avanzado.
- Ataques automatizados y personalizados.
- Deepfakes y desinformación.
- Ataques adversarios contra modelos de IA.
- Dependencia excesiva de sistemas opacos (black-box).

La IA es, por tanto, una herramienta dual:
puede fortalecer o debilitar la seguridad según su uso.

Agentes inteligentes: tipos y entornos

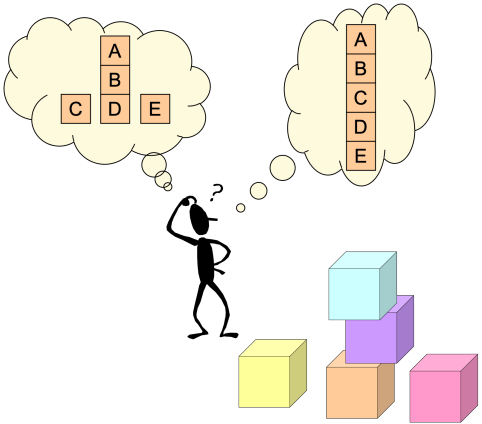
- Un agente inteligente es una entidad que:
  - Percibe su entorno mediante sensores.
  - Actúa sobre él mediante actuadores.
  - Toma decisiones para maximizar un objetivo.
- Entorno (en el que esta embebido):
  - Accesible / inaccesible.
  - Deterministas / estocástico.
  - Estáticos / dinámicos.
  - Discreto / continuo.



- Tipos de agentes:
  - Agentes reactivos simples: responden a estímulos inmediatos.
  - Agentes basados en modelos: mantienen información del estado interno.
  - Agentes basados en objetivos: toman decisiones para alcanzar metas.
  - Agentes basados en utilidad: optimizan resultados según criterios.
  - Agentes con aprendizaje: mejoran su comportamiento con la experiencia.

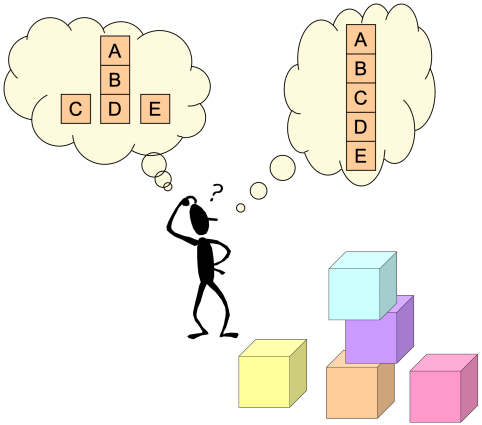
Aplicaciones (básicas) de IA

Problemas de planificación: Ordenar bloques



Aplicaciones (básicas) de IA

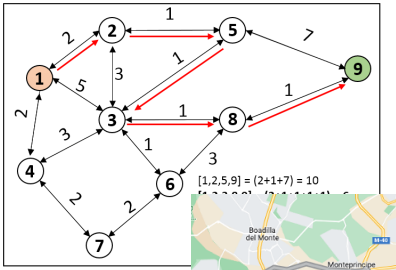
Problemas de planificación: Ordenar bloques

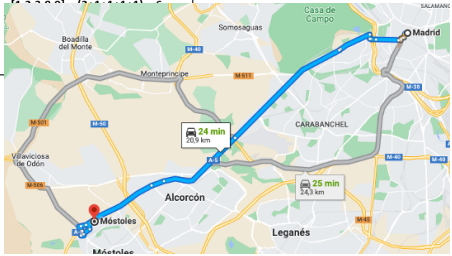




Aplicaciones (básicas) de IA (cont.)

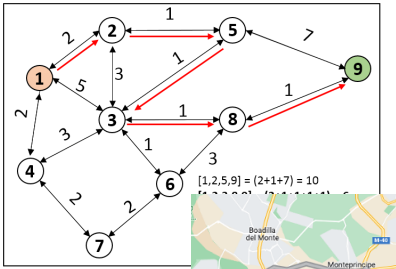
Problemas de optimización: Seleccionar camino más corto

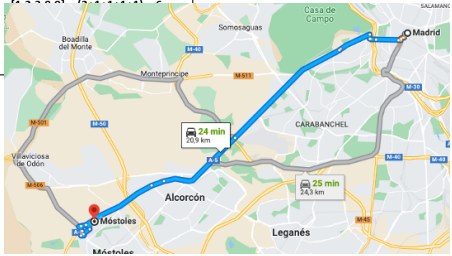


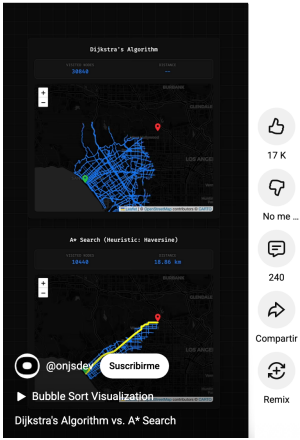


Aplicaciones (básicas) de IA (cont.)

Problemas de optimización: Seleccionar camino más corto



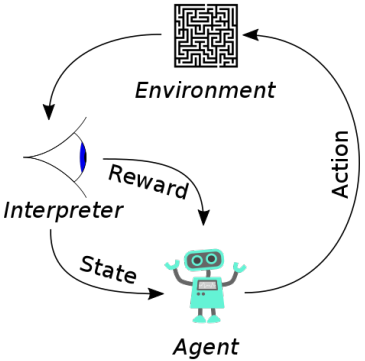




Link

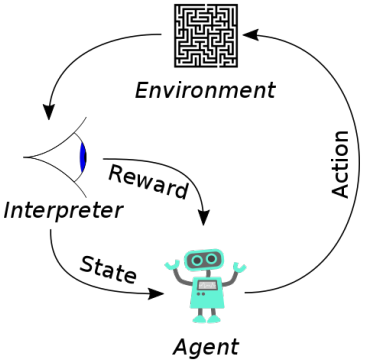
Aplicaciones (básicas) de IA (cont.)

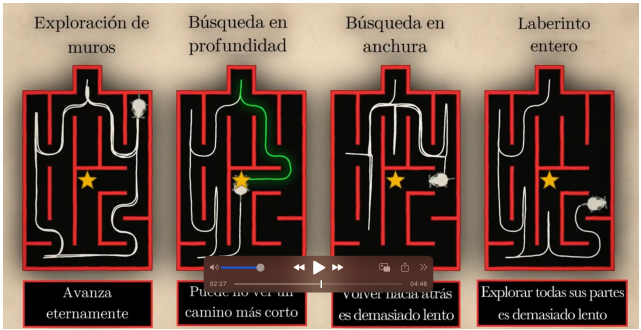
Técnicas para problemas complejos: Aprendizaje por refuerzo



Aplicaciones (básicas) de IA (cont.)

Técnicas para problemas complejos: Aprendizaje por refuerzo

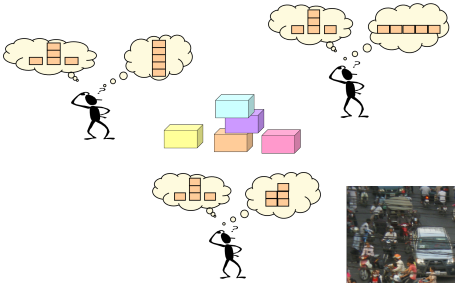




Link

Aplicaciones (básicas) de IA (cont.)

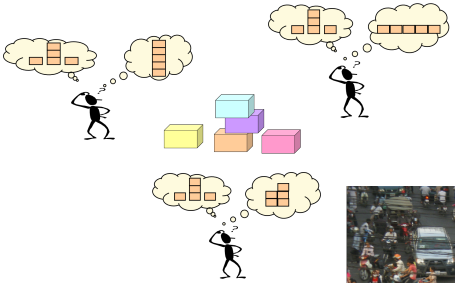
Sistemas multiagente I: Distintos objetivos



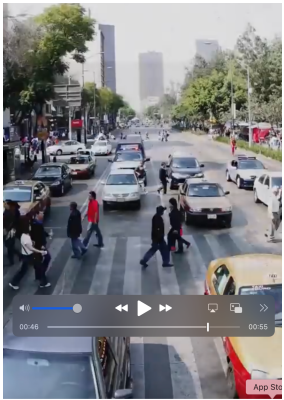


Aplicaciones (básicas) de IA (cont.)

Sistemas multiagente I: Distintos objetivos







Link

Aplicaciones (básicas) de IA (cont.)

Sistemas multiagente II: Entorno competitivo (Óptimo local)



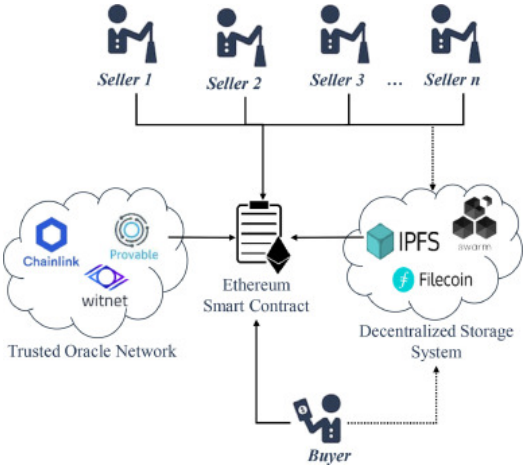
Aplicaciones (básicas) de IA (cont.)

Sistemas multiagente III: Entorno cooperativo (Óptimo global)



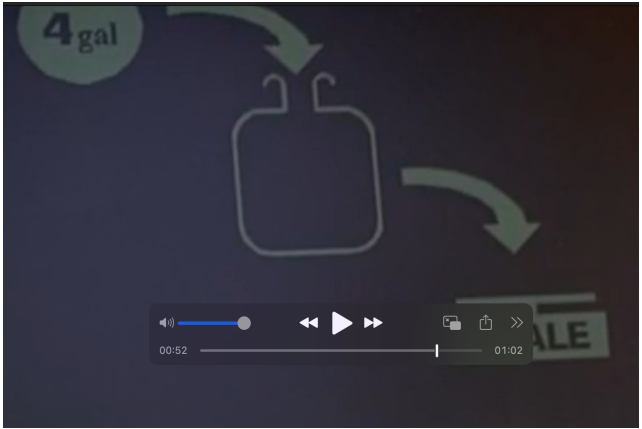
Aplicaciones (básicas) de IA (cont.)

Sistemas multiagente IV: Estrategias mixtas (Subastas)



Ejercicio para casa I: El problema de las garrafas

- Escenario: Tenemos dos garrafas, una de 5 galones y otra de 3.
- Objetivo: Depositar 4 galones exactos para que la bomba no explote.



Link

Ejercicio para casa II:

- Escenario: Una familia tiene que cruzar un puente frágil y sin iluminar. La hija tarda 1 min, el padre 2 min, la madre embarazada 5 min y el abuelo 8 min.
- Objetivo: Que todos crucen (max 2 por viaje y usando la linterna), antes de que se caiga (en 15 min).



Link

Referencias I

[1] J. Arias (2023). **Apuntes de Lógica: desde Aristóteles hasta Prolog**. Servicio de Publicaciones de la URJC. ISBN: 978-8409526345. URL: https://hdl.handle.net/10115/24277.

[2] — (2024a). **Apuntes de Programación Lógica: hasta Prolog y más allá**. BURJC. URL: https://hdl.handle.net/10115/41640.

[3] — (2024b). **Value Awareness Engineering**. Workshop on Ethical and Trustworthy AI. URL: https://shorturl.at/q2NYq.

[4] Aristotle (-350). **Organon**. Original work in Greek, Ancient Greece.

[5] J. Haugeland (1985). **Artificial intelligence: The very idea**. MIT press. ISBN: 978-0262081535.

[6] R. Kurzweil (1990). **The age of intelligent machines**. MIT press. ISBN: 978-0262111218.

Referencias II

[7] J. Lighthill (1973). **Artificial Intelligence: A General Survey**. En: *Artificial Intelligence: Paper Symposium*. Science Research Council, UK. URL: https://www.chilton-computing.org.uk/inf/literature/reports/lighthill_report/p001.htm.

[8] G. Luger y W. Stubblefield (1993). **Artificial Intelligence: Structures and Strategies for Complex Problem Solving**. Benjamin/Cummings Publishing Company. ISBN: 9780805347807.

[9] J. McCarthy, M. Minsky, N. Rochester y C. E. Shannon (1955). **A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence**. Research Proposal. Considered the founding document of Artificial Intelligence research. Dartmouth College. URL: http://raysolomonoff.com/dartmouth/boxa/dart564props.pdf.

[10] P. McCorduck (1979). **Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence**. First edition. A. K. Peters. ISBN: 978-0716710721.

[11] — (1991). **Máquinas que piensan: Una incursión personal en la historia y perspectivas de la inteligencia artificial**. Editorial Tecnos. ISBN: 978-8430920167.

Referencias III

[12] M. Minsky y S. Papert (1969). **Perceptrons: An introduction to computational geometry**. MIT Press. ISBN: 978-0262130431.

[13] T. Moto-Oka (2012). **Fifth Generation Computer Systems**. North Holland. ISBN: 978-0444600141.

[14] A. Newell y H. A. Simon (1961). **Computer simulation of human thinking**. En: *Science* 134.3495, págs. 2011-2017. DOI: 10.1126/science.134.3495.2011.

[15] D. E. Rumelhart, G. E. Hinton y R. J. Williams (1986). **Learning representations by back-propagating errors**. En: *Nature* 323, págs. 533-536. DOI: 10.1038/323533a0.

[16] S. J. Russell y P. Norvig (2021). **Artificial Intelligence: A Modern Approach**. 4th. Pearson. ISBN: 978-1292401133.

[17] A. M. Turing (1950). **Computing machinery and intelligence**. En: *Mind* 59.236, págs. 433-460. DOI: 10.1093/mind/LIX.236.433.

[18] VaST (2025). **Los 10 científicos actuales MÁS IMPORTANTES del mundo**. Video en YouTube. URL: https://www.youtube.com/watch?v=i2buuqjkxrI.

Referencias IV

[19] K. Weise y C. Metz (2023). **When A.I. Chatbots Hallucinate**. En: *The New York Times*. URL: https://www.nytimes.com/2023/05/01/business/ai-chatbots-hallucination.html.



Universidad
Rey Juan Carlos

Escuela Técnica Superior
Ingeniería Informática

Módulo II: **Resolución de problemas y búsqueda**

Asignatura: Inteligencia Artificial

Joaquín Arias

Universidad Rey Juan Carlos
Escuela Técnica Superior de Ingeniería Informática

Curso 2025-2026

Índice I

Tema 2: **Búsqueda no informada**
- **Representación de problemas como espacios de estados**
- **Resolución del problema**
  - Búsqueda en profundidad
  - Búsqueda en anchura
  - Búsqueda de coste uniforme
- **Comparación de algoritmos: complejidad y completitud**



Copyright ©2026 Joaquín Arias. Esta obra, basada en trasparencias de Sascha, Holger y Alberto (URJC'23) & de Carmen y Paco (URJC'21), está bajo la licencia CC BY-SA 4.0, **Creative Commons Atribución-CompartirIgual 4.0 Internacional**. Cómo citar esta obra: Arias, Joaquín y Fidilio-Allende, Luciana (2026). ***LIBRO: Inteligencia Artificial, por y para la ciberseguridad***. BURJC, Madrid.

Índice II

Tema 3: **Búsqueda heurística**
- **Heurísticas: definición y propiedades**
- **Búsqueda voraz**
- **Algoritmo A***
- **Heurísticas admisibles y consistentes**

Tema 4: **Búsqueda multiagente**
- **Entornos multiagente**
- **Juegos competitivos y cooperativos**
- **Algoritmo Minimax**
- **Poda alfa-beta**
- **Introducción a juegos estocásticos**

Tema 2: **Búsqueda no informada**

Representación de problemas como espacios de estados

- Un algoritmo de búsqueda es un conjunto de instrucciones.[1]
- La búsqueda en IA es un método universal de resolución de problemas:
  - Encontrar un elemento *e* en una lista.
  - Determinar el camino más corto en un grafo.
  - Planificar la secuencia de tareas a realizar en un proyecto.
- Dado un espacio de estados, busca las transiciones que llevan de un estado inicial a un estado objetivo, p.ej.:
  - Siguiente movimiento en el juego del tres en raya.

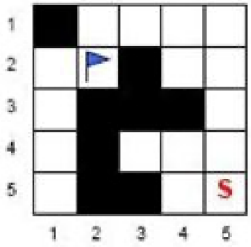
[1] Consultar [Russell y Norvig 2021, sección 6] y [Nilsson 2001, capítulos 8 y 9].

Representación de problemas como espacios de estados (cont.)

- Según la información del entorno disponible tenemos:
  - Algoritmos no informados o ciegos.
  - Algoritmos informados.
- EJEMPLO: búsqueda de un elemento en una lista tenemos:
  - Búsqueda secuencial, no sabemos si la lista está ordenada.
  - Búsqueda binaria, sabemos que la lista está ordenada.
- En general los algoritmos informados son más eficientes:
  - Algoritmos heurísticos: (i) Voraz o Greedy y (ii) A*.
  - Algoritmos de mejora iterativa: (i) Escalada Simple y (ii) Escalada por Máxima Pendiente.
  - Algoritmos con adversario: (i) Minimax y (ii) Poda alfa-beta.

Representación de problemas como espacios de estados (cont.)

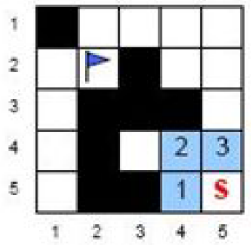
- Sparki quiere llegar a la bandera azul: las celdas de color negro son obstáculos y las celdas blancas son caminos posibles:

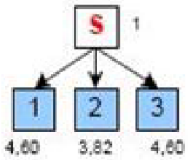




Representación de problemas como espacios de estados (cont.)

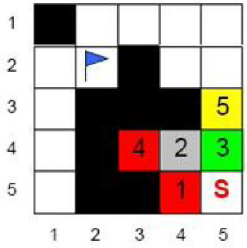
- La manera de representar el espacio de estados es mediante árboles.

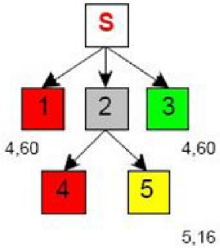




Representación de problemas como espacios de estados (cont.)

- Según exploramos el espacio algunos movimientos no permiten continuar (celdas marcadas en rojo).

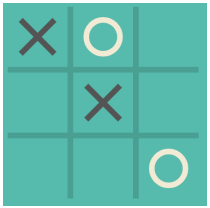




Representación de problemas como espacios de estados (cont.)

Ejercicio 1 Diferencia entre un algoritmo ciego y un algoritmo informado.

Ejercicio 2 Dado el siguiente tablero del tres-en-raya:



a) Dibujar el espacio de estados a partir de dicho tablero (juega la X).
b) Representar este tablero concreto usando dos tipos de dato algebraicos diferentes.

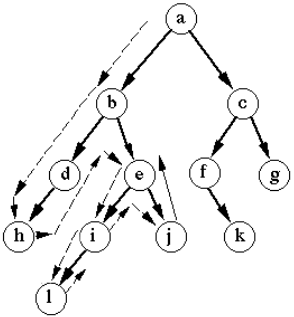
Resolución del problema

- Dada la representación del espacio de estados del problema:
  - Hay que encontrar un "camino" al estado objetivo.
  - Existen varias estrategias de búsqueda. P.ej.:

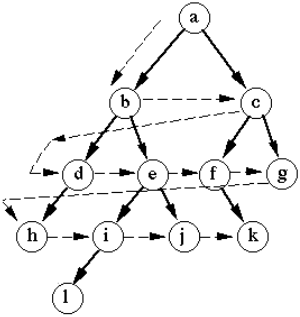
Resolución del problema

- Dada la representación del espacio de estados del problema:
  - Hay que encontrar un "camino" al estado objetivo.
  - Existen varias estrategias de búsqueda. P.ej.:

En profundidad



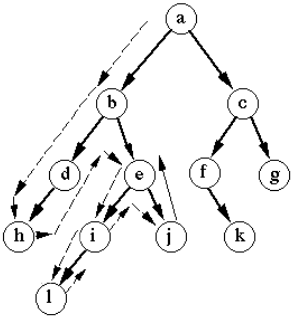
En Anchura



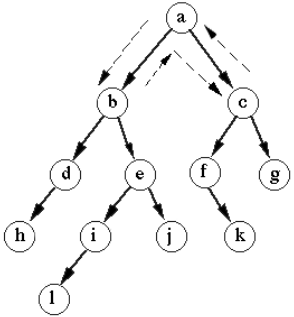
Resolución del problema

- Dada la representación del espacio de estados del problema:
  - Hay que encontrar un "camino" al estado objetivo.
  - Existen varias estrategias de búsqueda. P.ej.:

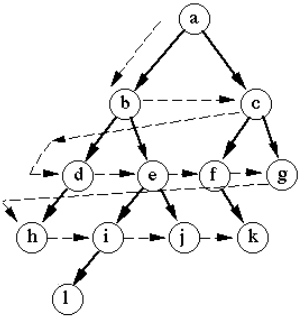
En profundidad



Profundización iterativa



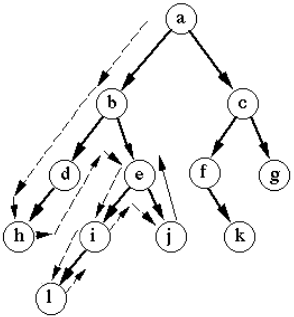
En Anchura



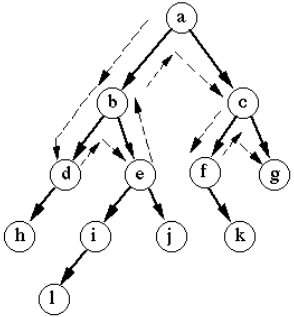
Resolución del problema

- Dada la representación del espacio de estados del problema:
  - Hay que encontrar un "camino" al estado objetivo.
  - Existen varias estrategias de búsqueda. P.ej.:

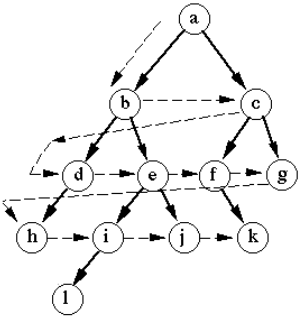
En profundidad



Profundización iterativa



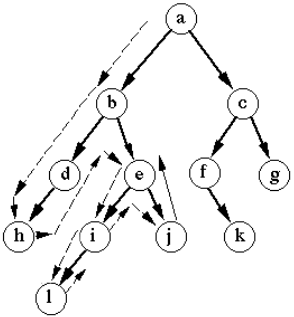
En Anchura



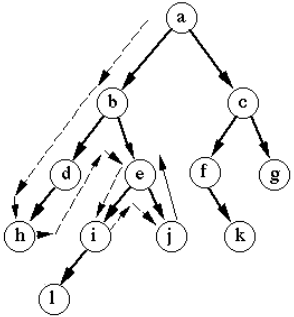
Resolución del problema

- Dada la representación del espacio de estados del problema:
  - Hay que encontrar un "camino" al estado objetivo.
  - Existen varias estrategias de búsqueda. P.ej.:

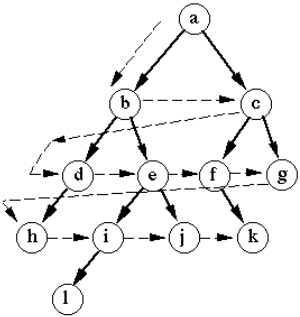
En profundidad



Profundización iterativa



En Anchura



Resolución del problema (cont.)

- El espacio de estados modeliza el mundo mediante un grafo:
  - Cada estado posible es un nodo.
  - Los estados inicial, actual, objetivo son nodos específicos...
  - Una acción para pasar de un estado a otro es un arco entre nodos.
  - Una secuencia de acciones es un camino desde el nodo inicial al nodo objetivo.
- Hay problemas que se representan como grafos dirigidos (ver Fig. 1)

  ...otros como grafos no dirigidos (ver Fig. 2).

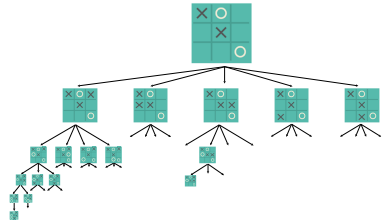


Figura 1: **Tres-en-raya:** Espacio de estados y posibles partidas desde `[X,O,_,_,X,_,_,_,O]`.

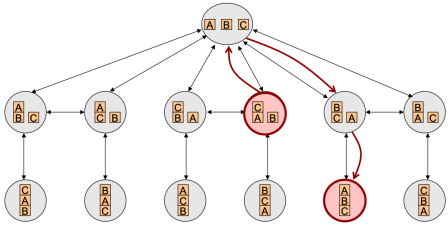


Figura 2: **3-bloques:** Espacio de estados y secuencia mínima para ir de `[[A,C],[B]]` a `[[C,B,A]]`.

Búsqueda en profundidad

```
1  ALGORITMO En_Profundidad_v0(E_Inicial, E_Objetivo)
2     Q ← [E_Inicial]
3
4     MIENTRAS Q ≠ [] HACER
5        [E_Actual | Tail] ← Q
6        Q ← Tail
7
8        SI E_Actual == E_Objetivo ENTONCES
9           DEVUELVE E_Actual
10
11       PARA CADA Vecino EN vecinos(E_Actual) HACER
12          Q ← [Vecino | Q]
13    DEVUELVE Error
```

- Pseudocódigo para búsqueda en profundidad, versión 0:
  - Añade los nodos en la cabeza de la lista `Q`:

`[]`: lista vacía.

`[A|As]`: `A` es la cabeza (un nodo). `As` es la cola (una lista).

  - `vecinos(A)` son los nodos alcanzables desde `A`.

Búsqueda en profundidad

```
1  ALGORITMO En_Profundidad_v0(E_Inicial, E_Objetivo)
2     Q ← [E_Inicial]
3
4     MIENTRAS Q ≠ [] HACER
5        [E_Actual | Tail] ← Q
6        Q ← Tail
7
8        SI E_Actual == E_Objetivo ENTONCES
9           DEVUELVE E_Actual
10
11       PARA CADA Vecino EN vecinos(E_Actual) HACER
12          Q ← [Vecino | Q]
13    DEVUELVE Error
```

- Pseudocódigo para búsqueda en profundidad, versión 0:
  - Añade los nodos en la cabeza de la lista `Q`:

`[]`: lista vacía.

`[A|As]`: `A` es la cabeza (un nodo). `As` es la cola (una lista).

  - `vecinos(A)` son los nodos alcanzables desde `A`.

Ejercicio 3 Explica porque al aplicar `En_Profundidad_v0` podemos resolver el tres-en-raya de la Fig. 1

...pero entra en bucle con los 3-bloques de la Fig. 2.

Búsqueda en profundidad (cont.)

```
1  ALGORITMO En_Profundidad_v1(E_Inicial, E_Objetivo)
2     Q ← [E_Inicial]
3     Visitados ← [E_Inicial]
4
5     MIENTRAS Q ≠ [] HACER
6        [E_Actual | Tail] ← Q
7        Q ← Tail
8
9        SI E_Actual == E_Objetivo ENTONCES
10          DEVUELVE E_Actual
11
12       PARA CADA Vecino EN vecinos(E_Actual) HACER
13          SI Vecino ∉ Visitados ENTONCES
14              Visitados ← [Vecino | Visitados]
15              Q ← [Vecino | Q]
16    DEVUELVE Error
```

- Pseudocódigo para búsqueda en profundidad, versión 1:
  - Evita expansión de nodos repetidos, recordando los nodos visitados.

`A`∉`As`: comprueba que el elemento `A` no está en la lista `As`.

Búsqueda en profundidad (cont.)

```
1  ALGORITMO En_Profundidad_v1(E_Inicial, E_Objetivo)
2      Q ← [E_Inicial]
3      Visitados ← [E_Inicial]
4
5      MIENTRAS Q ≠ [] HACER
6          [E_Actual | Tail] ← Q
7          Q ← Tail
8
9          SI E_Actual == E_Objetivo ENTONCES
10             DEVUELVE E_Actual
11
12         PARA CADA Vecino EN vecinos(E_Actual) HACER
13             SI Vecino ∉ Visitados ENTONCES
14                 Visitados ← [Vecino | Visitados]
15                 Q ← [Vecino | Q]
16     DEVUELVE Error
```

- Pseudocódigo para búsqueda en profundidad, versión 1:
  - Evita expansión de nodos repetidos, recordando los nodos visitados.

`A`∉`As`: comprueba que el elemento `A` no está en la lista `As`.

Ejercicio 4 Adapta `En_Profundidad_v1` para que devuelva la secuencia de nodos para llegar al desde `E_Inicial` hasta `E_Objetivo`.

Búsqueda en anchura

```
1  ALGORITMO En_Anchura(E_Inicial, E_Objetivo)
2     Q ← [E_Inicial]
3     Visitados ← [E_Inicial]
4
5     MIENTRAS Q ≠ [] HACER
6        [E_Actual | Tail] ← Q
7        Q ← Tail
8
9        SI E_Actual == E_Objetivo ENTONCES
10          DEVUELVE E_Actual
11
12       PARA CADA Vecino EN vecinos(E_Actual) HACER
13          SI Vecino ∉ Visitados ENTONCES
14              Visitados ← [Vecino | Visitados]
15              Q ← Q ++ [Vecino]
16    DEVUELVE Error
```

- Pseudocódigo para búsqueda en anchura:
  - Añade los nodos al final de la lista `Q`:

`A ++ B`: concatena las listas `A` y `B`.

  - No expande nodos visitados. No es necesario pero evita búsquedas redundantes.

Búsqueda en anchura

```
1  ALGORITMO En_Anchura(E_Inicial, E_Objetivo)
2      Q ← [E_Inicial]
3      Visitados ← [E_Inicial]
4
5      MIENTRAS Q ≠ [] HACER
6          [E_Actual | Tail] ← Q
7          Q ← Tail
8
9          SI E_Actual == E_Objetivo ENTONCES
10             DEVUELVE E_Actual
11
12         PARA CADA Vecino EN vecinos(E_Actual) HACER
13             SI Vecino ∉ Visitados ENTONCES
14                 Visitados ← [Vecino | Visitados]
15                 Q ← Q ++ [Vecino]
16     DEVUELVE Error
```

- Pseudocódigo para búsqueda en anchura:
  - Añade los nodos al final de la lista `Q`:

`A ++ B`: concatena las listas `A` y `B`.

  - No expande nodos visitados. No es necesario pero evita búsquedas redundantes.

Ejercicio 5 Aplica `En_Anchura` al 3-bloques (Fig. 2) y explica, con tus propias palabras, diferencias al aplicar `En_Profundidad_v1`.

Búsqueda de coste uniforme

```
1  ALGORITMO Coste_Uniforme(E_Inicial, E_Objetivo)
2     Q ← [(E_Inicial,0)]
3
4     MIENTRAS Q ≠ [] HACER
5        [(E_Actual,C) | Tail] ← Q
6        Q ← Tail
7
8        SI E_Actual == E_Objetivo ENTONCES
9           DEVUELVE E_Actual
10
11       PARA CADA Vecino EN vecinos(E_Actual) HACER
12          Q ← Q ++ [(Vecino,C+1)]
13          Q ← sort(Q)
14    DEVUELVE Error
```

- Pseudocódigo para búsqueda de coste uniforme (constante):
  - Considera coste `1` para cada acción:

`(A,C)`: el nodo `A` tiene un coste asociado `C`.

  - Añade el nodo y el coste a `Q` y los ordena de manera ascendente según el coste.

Búsqueda de coste uniforme

```
1  ALGORITMO Coste_Uniforme(E_Inicial, E_Objetivo)
2     Q ← [(E_Inicial,0)]
3
4     MIENTRAS Q ≠ [] HACER
5        [(E_Actual,C) | Tail] ← Q
6        Q ← Tail
7
8        SI E_Actual == E_Objetivo ENTONCES
9           DEVUELVE E_Actual
10
11       PARA CADA Vecino EN vecinos(E_Actual) HACER
12          Q ← Q ++ [(Vecino,C+1)]
13          Q ← sort(Q)
14    DEVUELVE Error
```

- Pseudocódigo para búsqueda de coste uniforme (constante):
  - Considera coste `1` para cada acción:

`(A,C)`: el nodo `A` tiene un coste asociado `C`.

  - Añade el nodo y el coste a `Q` y los ordena de manera ascendente según el coste.

Ejercicio 6 Aplica `Coste_Uniforme` al 3-bloques (Fig. 2) y explica si hace falta (o no) recordar algún(os) nodo(s) visitado(s).

Búsqueda de coste uniforme (cont.)

```
1  ALGORITMO Coste_Uniforme_G(E_Inicial, E_Objetivo)
2     Q ← [(E_Inicial,0)]
3
4     MIENTRAS Q ≠ [] HACER
5        [(E_Actual,C) | Tail] ← Q
6        Q ← Tail
7
8        SI E_Actual == E_Objetivo ENTONCES
9           DEVUELVE E_Actual
10
11       PARA CADA <Vecino,G> EN vecinos(E_Actual) HACER
12          Q ← Q ++ [(Vecino,C+G)]
13          Q ← sort(Q)
14    DEVUELVE Error
```

- Pseudocódigo para búsqueda de coste uniforme:
  - Considera un coste `G` diferente para cada transición entre nodos:

`<A,G>`: el nodo `A` esta conectado con arista de peso `G`.

Búsqueda de coste uniforme (cont.)

```
1  ALGORITMO Coste_Uniforme_G(E_Inicial, E_Objetivo)
2     Q ← [(E_Inicial,0)]
3
4     MIENTRAS Q ≠ [] HACER
5        [(E_Actual,C) | Tail] ← Q
6        Q ← Tail
7
8        SI E_Actual == E_Objetivo ENTONCES
9           DEVUELVE E_Actual
10
11       PARA CADA <Vecino,G> EN vecinos(E_Actual) HACER
12          Q ← Q ++ [(Vecino,C+G)]
13          Q ← sort(Q)
14    DEVUELVE Error
```

- Pseudocódigo para búsqueda de coste uniforme:
  - Considera un coste `G` diferente para cada transición entre nodos:

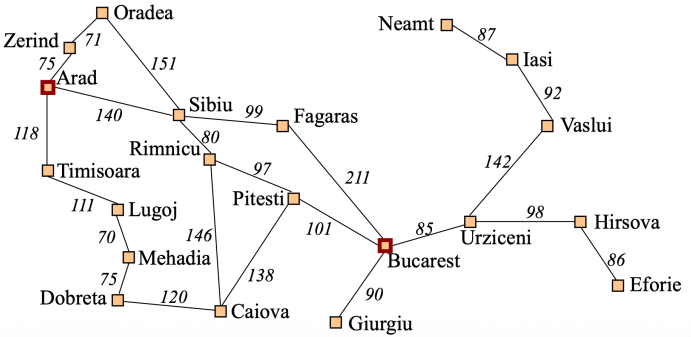
`<A,G>`: el nodo `A` esta conectado con arista de peso `G`.

Ejercicio 7 Adapta los algoritmos `En_Profundidad_v1` y `En_Anchura` para que acumulen el coste de transiciones (peso de las aristas).

Comparación de algoritmos: complejidad y completitud

Ejercicio 8 Encontrar un camino desde Arad hasta Bucarest. Representar los distintos valores de `Q` (en al menos 10 iteraciones). Usando los algoritmos:[2]

a) `En_Profundidad_v1`.
b) `En_Anchura`.
c) `Coste_Uniforme`.
d) `Coste_Uniforme_G`.



[2] Adaptados para acumular costes (Ejercicio 7).

Comparación de algoritmos: complejidad y completitud (cont.)

Algoritmo	Tiempo	Espacio	Completo	Óptimo	Comentarios
En anchura	$O(b^d)$	$O(b^d)$	Sí	Sí (costes uniformes)	Explora por niveles; alto consumo de memoria.
En profundidad	$O(b^m)$	$O(bm)$	No (ciclos)	No	Bajo consumo de memoria; puede quedar atrapada.
Profundización iterativa	$O(b^d)$	$O(bd)$	Sí	Sí (costes uniformes)	Combina completitud de anchura con bajo consumo de memoria de profundidad.
Coste uniforme	$O(b^{\lceil C^*/\varepsilon \rceil})$	$O(b^{\lceil C^*/\varepsilon \rceil})$	Sí ($\varepsilon > 0$)	Sí	Generaliza anchura a costes no uniformes.

Notación: b: Factor de ramificación. d: Profundidad de la solución más superficial. m: Profundidad máxima del árbol de búsqueda. C^*: Coste de la solución óptima. ε: Coste mínimo de una acción.

Tema 3: **Búsqueda heurística**

Heurísticas: definición y propiedades

Definición:
Una **heurística** es una función $h(n)$ que estima el coste mínimo necesario para alcanzar el estado objetivo desde un nodo n.

- Su objetivo principal es guiar el proceso de búsqueda hacia las regiones más prometedoras del espacio de estados, reduciendo el número de nodos explorados.
- Las heurísticas no garantizan por sí mismas una solución óptima, pero permiten mejorar significativamente la eficiencia de los algoritmos de búsqueda informada.

Heurísticas: definición y propiedades

Definición:
Una **heurística** es una función $h(n)$ que estima el coste mínimo necesario para alcanzar el estado objetivo desde un nodo n.

- Su objetivo principal es guiar el proceso de búsqueda hacia las regiones más prometedoras del espacio de estados, reduciendo el número de nodos explorados.
- Las heurísticas no garantizan por sí mismas una solución óptima, pero permiten mejorar significativamente la eficiencia de los algoritmos de búsqueda informada.

Propiedades de las heurísticas

- **Informatividad**: cuanto más cercana sea la estimación al coste real, más eficiente será la búsqueda.
- **Bajo coste computacional**: la heurística debe ser rápida de calcular.
- **No negatividad**: generalmente se cumple que $h(n) \geq 0$.
- **Calidad**: una heurística domina a otra si proporciona estimaciones más precisas sin sobreestimar el coste real.

Búsqueda voraz

Definición:
La **búsqueda voraz** (*Greedy Best-First Search*) es un algoritmo de búsqueda informada que selecciona en cada iteración el nodo que parece estar más cercano al objetivo según la heurística.

- La función de evaluación se define como:

$$f(n) = h^*(n)$$

Búsqueda voraz

Definición:
La **búsqueda voraz** (*Greedy Best-First Search*) es un algoritmo de búsqueda informada que selecciona en cada iteración el nodo que parece estar más cercano al objetivo según la heurística.

- La función de evaluación se define como:

$$f(n) = h^*(n)$$

- Este enfoque ignora el coste acumulado desde el nodo inicial y se basa únicamente en la estimación heurística.

Características

- No garantiza encontrar la solución óptima.
- No siempre es completa en espacios de búsqueda infinitos.
- Es rápida y sencilla de implementar.

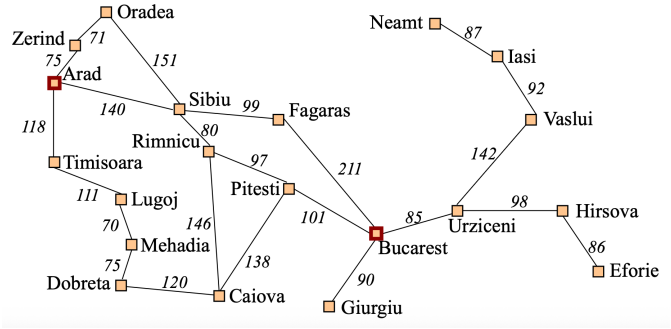
Búsqueda voraz (cont.)

```
1  ALGORITMO Voraz(E_Inicial, E_Objetivo, h*)
2     Q ← [(E_Inicial, h*(E_Inicial) )]
3
4     MIENTRAS Q ≠ [] HACER
5        [(E_Actual, _ ) | Tail] ← Q
6        Q ← Tail
7
8        SI E_Actual == E_Objetivo ENTONCES
9           DEVUELVE E_Actual
10
11       PARA CADA Vecino EN vecinos(E_Actual) HACER
12          Q ← Q ++ [(Vecino, h*(Vecino) )]
13          Q ← sort(Q)
14    DEVUELVE Error
```

- Pseudocódigo para búsqueda voraz:
  - Solo considera la estimación `h*(n)` en cada nodo: `(A,h*(A))`: el nodo `A` tiene un coste estimado a destino de `h*(A)`.
  - Añade el nodo y el coste a `Q` y los ordena de manera ascendente según el coste.

Búsqueda voraz (cont.)

Ejercicio 9 Encontrar un camino desde Arad hasta Bucarest usando el algoritmo `Voraz` y la función heurística $h^*(n)$ adjunta.



Ciudad	$h^*(n)$		
Arad	366	Mehadia	241
Bucarest	0	Neamt	234
Caiova	160	Oradea	380
Dobreta	242	Pitesti	98
Eforie	161	Rimnicu	193
Fagaras	178	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Cuadro 1: $h^*(n)$: distancia desde n hasta Bucarest.

Algoritmo A*

Definición:
El algoritmo **A*** es un método de búsqueda informada que combina el coste real desde el inicio con una estimación heurística del coste restante.

- La función de evaluación se define como:

$$f(n) = g(n) + h^*(n)$$

donde:

- $g(n)$ representa el coste real desde el nodo inicial hasta el nodo n.
- $h^*(n)$ es la estimación heurística del coste desde n hasta el objetivo.

Algoritmo A*

Definición:
El algoritmo **A*** es un método de búsqueda informada que combina el coste real desde el inicio con una estimación heurística del coste restante.

- La función de evaluación se define como:

$$f(n) = g(n) + h^*(n)$$

donde:

- $g(n)$ representa el coste real desde el nodo inicial hasta el nodo n.
- $h^*(n)$ es la estimación heurística del coste desde n hasta el objetivo.

- El algoritmo selecciona el nodo con menor valor de $f(n)$ para su expansión, garantizando soluciones óptimas (si la heurística utilizada es admisible [pág. 27]).

Ventajas y desventajas

- **Ventajas**: completo y óptimo con heurísticas adecuadas.
- **Desventajas**: elevado consumo de memoria y dependencia de la calidad de la heurística.

Algoritmo A* (cont.)

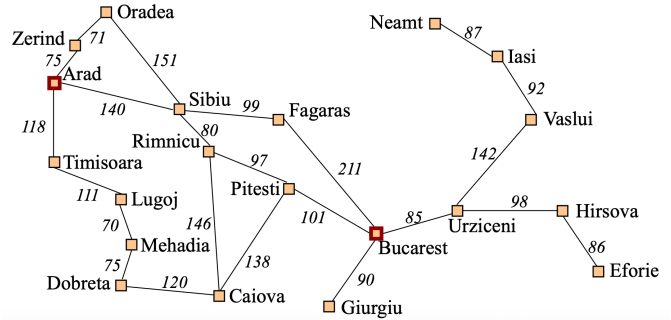
```
1  ALGORITMO A*(E_Inicial, E_Objetivo, h*)
2     Q ← [(E_Inicial, 0, h*(E_Inicial) )]
3
4     MIENTRAS Q ≠ [] HACER
5        [(E_Actual, G_Actual, _ ) | Tail] ← Q
6        Q ← Tail
7
8        SI E_Actual == E_Objetivo ENTONCES
9           DEVUELVE E_Actual
10
11       PARA CADA <Vecino,G> EN vecinos(E_Actual) HACER
12          G_Vecino ← G_Actual + G
13          F_Vecino ← G_Vecino + h*(Vecino)
14          Q ← Q ++ [(Vecino, G_Vecino, F_Vecino)]
15          Q ← sort(Q)
16    DEVUELVE Error
```

(A,g(A),f(A)):

- Pseudocódigo para algoritmo A*:
  - Considera el coste real `G` de ir de `E_Actual` a `Vecino`: el nodo `A` tiene una coste real desde inicio de `g(A)` y un coste estimado total de inicio a destino de `f(A)`.
  - Añade el nodo y los costes a `Q` y los ordena de manera ascendente según `f(A)`.

Algoritmo A* (cont.)

Ejercicio 10 Encontrar un camino desde Arad hasta Bucarest usando el algoritmo A* con la función heurística h^* adjunta.



Ciudad	$h^*(n)$
Arad	366
Bucarest	0
Caiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Cuadro 2: $h^*(n)$: distancia desde n hasta Bucarest.

Heurísticas admisibles y consistentes

Heurísticas admisibles

Una heurística es **admisible** (u optimista) si nunca sobreestima el coste real mínimo para alcanzar el objetivo:

$$h^*(n) \leq h(n)$$

donde $h(n)$ representa el coste real óptimo desde el nodo n hasta el objetivo.

- El uso de heurísticas admisibles garantiza que el algoritmo A* encuentre una solución óptima.

Heurísticas admisibles y consistentes

Heurísticas admisibles

Una heurística es **admisible** (u optimista) si nunca sobreestima el coste real mínimo para alcanzar el objetivo:

$$h^*(n) \leq h(n)$$

donde $h(n)$ representa el coste real óptimo desde el nodo n hasta el objetivo.

- El uso de heurísticas admisibles garantiza que el algoritmo A* encuentre una solución óptima.

Heurísticas consistentes

Una heurística es **consistente** (o monótona) si cumple la siguiente condición para cualquier nodo n y su sucesor n':

$$h^*(n) \leq g(n, n') + h^*(n')$$

donde $g(n, n')$ es el coste de la acción que lleva de n a n'.

- Las heurísticas consistentes aseguran que los valores de $f(n)$ no disminuyan a lo largo de un camino y simplifican la implementación del algoritmo A*.
- Cabe destacar que toda heurística consistente es admisible, pero no toda heurística admisible es consistente.

Tema 4: **Búsqueda multiagente**

Entornos multiagente

- La búsqueda multiagente estudia problemas en los que intervienen varios agentes que toman decisiones de forma simultánea o secuencial.
- A diferencia de la búsqueda clásica, el resultado de una acción depende no solo del entorno, sino también de las acciones realizadas por otros agentes.
- En este tipo de problemas:
  - Existen múltiples agentes con objetivos propios.
  - El entorno es dinámico y estratégico.
  - Cada agente debe razonar teniendo en cuenta el comportamiento de los demás.
- Ejemplos típicos de búsqueda multiagente incluyen juegos de mesa, robots colaborativos y sistemas de negociación automática.

Entornos multiagente (cont.)

- Un entorno multiagente se caracteriza por la interacción entre dos o más agentes. Estos entornos pueden clasificarse según distintas dimensiones:
  - **Cooperativos**: los agentes comparten objetivos comunes.
  - **Competitivos**: los objetivos de los agentes están en conflicto.
  - **Mixtos**: algunos objetivos compartidos, otros opuestos.
- Otras propiedades relevantes son:
  - Deterministas o estocásticos.
  - De información completa o incompleta.
  - Por turnos o con acciones simultáneas.
- La complejidad de estos entornos surge de la necesidad de anticipar las decisiones de otros agentes.

Escenarios competitivos: Teoría de Juegos

Juegos competitivos

La teoría de juegos estudia situaciones en las que varios agentes toman decisiones estratégicas y el resultado para cada uno depende de las acciones de los demás.

En los **escenarios competitivos**:

- Los agentes tienen objetivos opuestos o en conflicto.
- La ganancia de un agente implica la pérdida de otro (juegos de suma cero).
- Cada agente asume que sus oponentes actúan de forma racional.

Escenarios competitivos: Teoría de Juegos

Juegos competitivos

La teoría de juegos estudia situaciones en las que varios agentes toman decisiones estratégicas y el resultado para cada uno depende de las acciones de los demás.

En los **escenarios competitivos**:

- Los agentes tienen objetivos opuestos o en conflicto.
- La ganancia de un agente implica la pérdida de otro (juegos de suma cero).
- Cada agente asume que sus oponentes actúan de forma racional.

- Tipos de juegos dependen:
  - Jugadores: dos (tres-en-raya) | ...
  - Azar: con (parchís) | sin (ajedrez).
  - Información: completa (damas) | incompleta (poker).
- Objetivo: anticipar la acción del adversario y encontrar la estrategia óptima.
- Consideraciones / notación:
  - Hay un estado inicial s_0.
  - Hay estados terminales (objetivo): $terminal(s) \rightarrow true|false$
  - Los estados terminales tienen valor de utilidad: $utilidad(s) \rightarrow k,\ k \in \mathbb{R}$

Algoritmo Minimax

Definición: El algoritmo Minimax es un método de búsqueda utilizado en juegos competitivos por turnos, deterministas y de suma cero, con información completa. [John von Neumann, 1928]

- Se basa en dos tipos de jugadores:
  - **MAX**: intenta maximizar la utilidad.
  - **MIN**: intenta minimizar la utilidad.
- El algoritmo construye un árbol de juego donde los nodos representan estados y las hojas (estados terminales) contienen valores de utilidad. La decisión óptima se obtiene propagando los valores a la raíz del árbol.
- Aunque Minimax garantiza la mejor decisión posible, su principal inconveniente es el alto coste computacional.

Algoritmo Minimax (cont.)

```
1  ALGORITMO Minimax(E_Inicial)
2     DEVUELVE valor_MAX(E_Inicial)
3
4  FUNCION valor_MAX(E_Actual)
5     SI terminal(E_Actual) ENTONCES
6        DEVUELVE utilidad(E_Actual)
7     Valor ← −∞
8     PARA CADA Vecino EN vecinos(E_Actual) HACER
9        Valor ← max(Valor, valor_MIN(Vecino) )
10    DEVUELVE Valor
11
12 FUNCION valor_MIN(E_Actual)
13    SI terminal(E_Actual) ENTONCES
14       DEVUELVE utilidad(E_Actual)
15    Valor ← ∞
16    PARA CADA Vecino EN vecinos(E_Actual) HACER
17       Valor ← min(Valor, valor_MAX(Vecino) )
18    DEVUELVE Valor
```

- Pseudocódigo para Minimax:
  - Empieza el agente MAX
  - Dos funciones mutuamente recursivas.

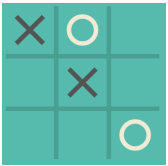
Heurísticas

- Introducir prueba de suspensión en cierta profundidad, en estados reposo...
- Estimar valor de utilidad en estados no terminales: $utilidad^*(s) \rightarrow k,\ k \in \mathbb{R}$
- Decisión no óptima, depende de la calidad de la heurística.

Algoritmo Minimax (cont.)

Ejercicio 11 Adapta el algoritmo `Minimax` (llámalo `Minimax*`) para suspender a un profundidad `p` y estimar el valor de utilidad con `utilidad*`.

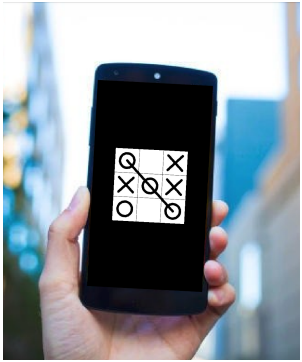
Ejercicio 12 Dada la siguiente posición del tres-en-raya (MAX juega X):



a) Calcula el valor de Minimax para los siguientes 5 estados posibles.
b) ¿Cuál es el siguiente movimiento óptimo de MAX según Minimax?
c) Estima el valor de Minimax* (Ejercicio 11) para profundidad 3 y $utilidad^*(s)$ = líneas abiertas para MAX − líneas abiertas para MIN.

Algoritmo Minimax (cont.)

- **Una cosa más**: Minimax se ha aplicado tradicionalmente en el tres-en-raya, inocente juego que, en Juegos de guerra (1983), se transforma en una metáfora del conflicto nuclear:
  - En `https://github.com/Xuaco/Tic-Tac-Toe` tenéis código, y manualc para implementarlo en Java usando Processing.
  - En [Arias y Carro 2019] se emplea como benchmark (también en su variante 4x4) para evaluar mejoras avanzadas en la estrategia de búsqueda de Prolog (ver [Arias 2024, Tema 3]).



Poda alfa-beta

Definición: La poda alfa–beta es una optimización del algoritmo Minimax que reduce el número de nodos evaluados sin alterar el resultado final.

- Se utilizan dos parámetros:
  - α: el mejor valor encontrado hasta el momento para MAX.
  - β: el mejor valor encontrado hasta el momento para MIN.
- Cuando se detecta que una rama no puede mejorar la solución actual, se deja de explorar. En el mejor de los casos, la complejidad se reduce de $O(b^d)$ a $O(b^{d/2})$, en el caso medio a $O(b^{3d/4})$.

Poda alfa-beta (cont.)

```
1  ALGORITMO Alfa-Beta(E_Inicial)
2     DEVUELVE valor_MAX(E_Inicial, −∞, ∞)
```

```
1  FUNCION valor_MAX(E_Actual, α, β)
2     SI terminal(E_Actual) ENTONCES
3        DEVUELVE utilidad(E_Actual)
4     PARA CADA Vecino EN vecinos(E_Actual) HACER
5        α ← max(α, valor_MIN(Vecino,α,β) )
6        SI α ≥ β ENTONCES
7           SALIR DEL BUCLE  % poda beta
8     DEVUELVE α
```

```
1  FUNCION valor_MIN(E_Actual, α, β)
2     SI terminal(E_Actual) ENTONCES
3        DEVUELVE utilidad(E_Actual)
4     PARA CADA Vecino EN vecinos(E_Actual) HACER
5        β ← min(β, valor_MAX(Vecino,α,β) )
6        SI β ≤ α ENTONCES
7           SALIR DEL BUCLE  % poda alfa
8     DEVUELVE β
```

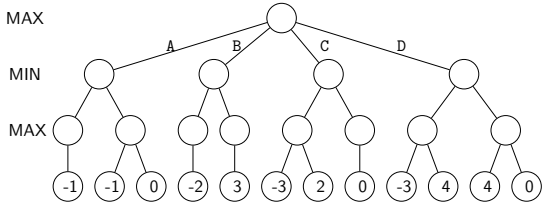
- Pseudocódigo para Minimax con poda alfa-beta:
  - α: es el mejor valor de evaluación para MAX en el camino hasta estado actual.
  - β: es el mejor para MIN.
- Poda alfa:
  - $\beta \leq \alpha$
  - `valor_MIN = min(`β`, ...)` $\leq$ β
  - `valor_MAX = max(`α`,valor_MIN,...)`
  - Como $\beta \leq \alpha$, entonces `valor_MAX` no depende de `valor_MIN` y no hay que explorar otros vecinos de MIN.

Poda alfa-beta (cont.)

Ejercicio 13 Explica con tus palabras la poda beta en un nodo MAX.

Ejercicio 14 Dibuja un árbol de juego con poda alfa profunda, i.e., poda de los hijos de un nodo MIN al comparar con el α de un ancestro lejano.

Ejercicio 15 Dado el árbol de juego de la figura:



a) ¿Cuántos nodos evitamos generar respecto al algoritmo `Minimax` si aplicamos el algoritmo `Alfa-Beta`?

b) ¿Cuál es el siguiente movimiento óptimo de MAX `(A,B,C o D)`?

Introducción a juegos estocásticos

Los juegos estocásticos

Incorporan incertidumbre / aleatoriedad en los resultados de las acciones.

Características:

- El estado siguiente depende de una probabilidad.
- Puede haber eventos aleatorios (dados, cartas, ruido del entorno).

Ejemplos:

- Póker.
- Backgammon.
- Juegos con fallos de sensores o actuadores.

ExpectiMinimax: Extensión del Minimax:

- Se Añade un nuevo jugador, AZAR, cuya decisión es aleatoria.
- Los vecinos de un nodo AZAR:
  - Son los posibles resultados del evento, p.ej., `cara` o `cruz`.
  - Cada resultado tiene asociado la probabilidad de que ocurra: para la moneda es `cara;1/2` y `cruz;1/2`.

Algorimgo ExpectiMinimax (cont.)

```
1  ALGORITMO ExpectiMinimax(E_Inicial)
2     DEVUELVE valor(E_Inicial, MAX)
3
4  FUNCION valor(E_Actual, Juega)
5     SI terminal(E_Actual) ENTONCES
6         Valor ← utilidad(E_Actual)
7     SINO SI Juega == MAX ENTONCES
8         Valor ← −∞
9         PARA CADA Vecino EN vecinos(E_Actual) HACER
10            Valor ← max(Valor, valor(AZAR_MAX,Vecino) )
11    SINO SI Juega == MIN ENTONCES
12        Valor ← ∞
13        PARA CADA Vecino EN vecinos(E_Actual) HACER
14            Valor ← min(Valor, valor(AZAR_MIN,Vecino) )
15    SINO
16        Valor ← 0
17        PARA CADA <Vecino,P> EN vecinos(E_Actual) HACER
18            Valor ← Valor + P * ( Juega == AZAR_MAX ? valor(Vecino, MIN) : valor(Vecino,MAX) )
19    DEVUELVE Valor
```

- Pseudocódigo para ExpectiMinimax:

`<Vecino,P>` el nodo `Vecino` tiene una probabilidad `P` asociada.

`C ? S : N`: abreviación de `SI C ENTONCES S SINO N`.

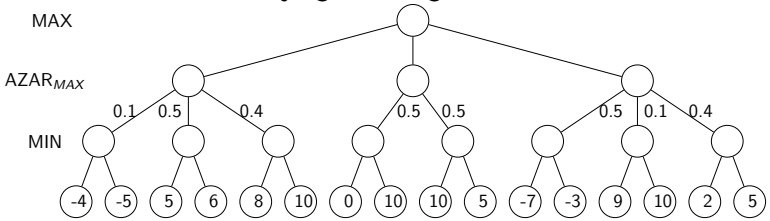
- La escala de los valores de `utilidad` afectan al resultado:
  - No devolver ∞ o $-\infty$.
  - Escalar los valores en intervalo finito, p.ej. $[1, 10]$.

Algorimgo ExpectiMinimax (cont.)

Ejercicio 16 * Implementa ExpectiMinimax con poda alfa-beta.

[Russell y Norvig 2021, sección 6.5.1]

Ejercicio 17 Dado el árbol de juego de la figura:



a) Calcula el valor de cada nodo aplicando `ExpectiMinimax`.

b) * ¿Cuántos nodos evitamos generar respecto `ExpectiMinimax` si realizamos una pofa alfa-beta, sabiendo que $utilidad(n) \in [-10, 10]$?

* Ejercicio avanzado; no se espera su resolución dentro del marco de la asignatura.

Referencias I

[1] J. Arias y M. Carro (2019). **Incremental Evaluation of Lattice-Based Aggregates in Logic Programming Using Modular TCLP**. En: *21st Int'l. Symposium on Practical Aspects of Declarative Languages, PADL'19*. Vol. 11372. LNCS. Springer, págs. 98-114. DOI: 10.1007/978-3-030-05998-9_7.

[2] J. Arias (2024). **Apuntes de Programación Lógica: hasta Prolog y más allá**. BURJC. URL: https://hdl.handle.net/10115/41640.

[3] N. J. Nilsson (2001). **Inteligencia Artificial: Nueva Síntesis**. McGraw-Hill. ISBN: 978-8448128241.

[4] S. J. Russell y P. Norvig (2021). **Artificial Intelligence: A Modern Approach**. 4th. Pearson. ISBN: 978-1292401133.



Universidad
Rey Juan Carlos

Escuela Técnica Superior
Ingeniería Informática

Módulo III: **Representación del conocimiento**

Asignatura: Inteligencia Artificial

Joaquín Arias

Universidad Rey Juan Carlos
Escuela Técnica Superior de Ingeniería Informática

Curso 2025-2026

Índice I

Tema 5: **Representación probabilística del conocimiento**
- **Incertidumbre y razonamiento probabilístico**
- **Probabilidad condicional**
- **Regla de Bayes**
- **Independencia y dependencia probabilística**

Tema 6: **Redes Bayesianas**
- **Redes Bayesianas**
  - Definición y componentes
  - Grafos dirigidos acíclicos
  - Tablas de probabilidad condicional
- **Construcción de modelos bayesianos**



Copyright ©2026 Joaquín Arias. Esta obra, basada en trasparencias de Sascha, Holger y Alberto (URJC'23) & de Carmen y Paco (URJC'21), está bajo la licencia CC BY-SA 4.0, Creative Commons Atribución-CompartirIgual 4.0 Internacional. Cómo citar esta obra: Arias, Joaquín y Fidilio-Allende, Luciana (2026). *LIBRO: Inteligencia Artificial, por y para la ciberseguridad*. BURJC, Madrid.

Índice II

Tema 7: **Inferencia en Redes Bayesianas**
- **Inferencia exacta**
  - Algoritmo de Enumeración
  - Algoritmo de Eliminación de variables
  - 'Implementacón' en Python
- **Inferencia aproximada**
  - Métodos principales
  - Algoritmos de propagación de creencias

Índice III

Tema 8: **Teoría de la decisión**
- **Fundamentos de la teoría de la decisión**
- **Utilidad**
  - Funciones de utilidad
  - Utilidad Esperada
- **Modelos de decisión**
  - Redes de decisión
  - Árboles de Decisión
  - Secuencias y procesos de Markov

Tema 5: Representación probabilística del conocimiento

Incertidumbre y razonamiento probabilístico

- En la vida real, en la IA (y en ciberseguridad), muchas decisiones deben tomarse con información incompleta o incierta (p.ej., alertas ruidosas, datos faltantes, ataques desconocidos). La representación probabilística permite modelar esta incertidumbre de forma rigurosa.
- **Incertidumbre**: falta de certeza sobre el estado real del sistema (por ejemplo, si una alerta corresponde a un ataque real).
- **Razonamiento probabilístico**, usa la probabilidad como medida cuantitativa de creencia.
  - Permite combinar evidencias, actualizar creencias y tomar decisiones informadas.
- **Ejemplo**: Probabilidad de que un tráfico sea malicioso dado un conjunto de indicadores (IP sospechosa, puerto inusual, patrón de comportamiento).

Probabilidad condicional

- Variable aleatoria: parte del mundo con incertidumbre, con un dominio de valores.
- Distribución de probabilidad: asigna una probabilidad P a cada valor x del dominio:

$$0 \leq P(V = x) \leq 1 \qquad \sum_{x \in Dom(V)} P(V = x) = 1$$

- **Definición**: La probabilidad condicional expresa la probabilidad de un evento A dado que otro ya ocurrió B:

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

A: evento de interés.

B: evidencia observada.

- **Ejemplo**: Distribución de probabilidad de:
  - T: Temperatura.
    Dominio: Dom(T) = {frio, calor}.
    Probabilidad: P(frio)=P(calor)=0,5.
  - C: Clima.

C	P(C)
sol	0,2
lluvia	0,5
niebla	?

Cuadro 1: Distribución de probabilidad.

Ejercicio 1 Dada la distribución de probabilidad de la variable C de la tabla 1, ¿Cuál es la probabilidad P(C=niebla)?

Probabilidad condicional (cont.)

- Distribución conjunta de $P(V_1 = x_1, V_2 = x_2, \ldots, V_n = x_n)$, se denota como, $P(x_1, x_2, \ldots, x_n)$, y cumple:

$$P(x_1, x_2, \ldots, x_n) \geq 0$$

$$\sum_{x_1 \in Dom(V_1)} \cdots \sum_{x_n \in Dom(V_n)} P(x_1, x_2, \ldots, x_n) = 1$$

- **Ejemplo**: Temperatura y clima:

T	C	P(T,C)
frio	sol	0,2
frio	lluvia	0,3
calor	sol	0,4
calor	lluvia	0,1

Cuadro 2: Distribución conjunta.

- Un evento E es un conjunto de posibles valores de la distribución. Su probabilidad se calcula como:

$$P(E) = \sum_{(x_1, \ldots, x_n) \in E} P(x_1, \ldots, x_n)$$

Ejercicio 2 Dada la tabla 2, calcula la probabilidad P(E) para los eventos:

a) E = calor y sol.
b) E = calor.
c) E = calor o sol.

Probabilidad condicional (cont.)

- La Distribución marginal de un subconjunto de variables $\vec{S}$ se obtiene a partir de la distribución conjunta de todas las variables $\vec{X}$.

$$P(\vec{S}) = \sum_{\vec{X} \setminus \vec{S}} P(\vec{X})$$

Ejercicio 3 Dada una distribución conjunta con tres variables (V_1, V_2, V_3), indica las distribuciones marginales y la fórmula correspondiente.

- **Ejemplo**: Dada la distribución conjunta para temperatura T y clima C de la Tabla 2 calcula las distribución marginal correspondientes.

P(T):

$$\text{P(T=frio)} = \sum_{C} \text{P(T=frio,C)} =$$
$$= \text{P(T=frio, C=sol)} + \text{P(T=frio, C=lluvia)} =$$
$$= 0,2 + 0,3 = 0,5$$
$$\text{P(T=calor)} = 0,4 + 0,1 = 0,5$$

P(C):

$$\text{P(C=sol)} = 0{,}2 + 0{,}4 = 0{,}6$$
$$\text{P(C=lluvia)} = 0{,}3 + 0{,}1 = 0{,}4$$

Probabilidad condicional (cont.)

- **Ejemplo**: Dada la distribución conjunta de T y C de la tabla 2, calcularemos la probabilidad de C cuando hace calor, es decir, la probabilidad condicional $P(C|T = calor)$.

a) Primero aplicando la fórmula:

$$\text{P(C=sol} \mid \text{T=calor)} = \frac{\text{P(C=sol,T=calor)}}{\text{P(calor)}} = \frac{0,4}{0,5} = 0,8$$

$$\text{P(C=lluvia} \mid \text{T=calor)} = \frac{\text{P(C=lluvia,T=calor)}}{\text{P(calor)}} = \frac{0,1}{0,5} = 0,2$$

b) Ahora, seleccionando las probabilidades conjuntas que concuerdan con la evidencia:

T	C	P(T,C)
calor	sol	0,4
calor	lluvia	0,1

Y normalizamos para que sume 1:

C	P(C | T=calor)
sol	0,8
lluvia	0,2

Ejercicio 4 Calcula la probabilidad de C cuando hace frio, i.e., P(C | T=frio).

Regla de Bayes

- **Definición**: La regla de Bayes permite invertir probabilidades condicionales:

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

  Donde:
  - $P(A)$: probabilidad a priori.
  - $P(B|A)$: verosimilitud.
  - $P(A|B)$: probabilidad a posteriori.

- **Ejemplo**: Actualizar la probabilidad de intrusión tras observar un comportamiento anómalo.
  - $P(\text{Intrusión}|\text{Alerta IDS}) = 0,90$
  - $P(\text{Intrusión}|\text{Alerta IDS}, \text{Comportamento anómalo}) = 0,91$

Regla de Bayes (cont.)

- **Regla del producto**: Obtenemos la probabilidad conjunta $P(A, B)$ a partir de la probabilidad condicional $P(A|B)$:

$$P(A|B) = \frac{P(A, B)}{P(B)} \Leftrightarrow P(A, B) = P(A|B) \cdot P(B)$$

- **Regla de la cadena**: Nos permite escribir una distribución conjunta como producto incremental de distribuciones condicionales:

$$P(x_1, x_2, x_3) = P(x_1) \cdot P(x_2|x_1) \cdot P(x_3|x_1, x_2)$$

  En general $P(x_1, x_2, \ldots, x_n) = \prod_i P(x_i|x_1, \ldots, x_{i-1})$

Ejercicio 5 Demuestra la Regla de Bayes $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$ a partir de la regla del producto y la regla de la cadena.

Independencia y dependencia probabilística

- **Definición** En IA (aplicada o no a la ciberseguridad), no todos los eventos son iguales: algunos ocurren de forma aislada y otros están relacionados. Por lo tanto disntinguimos entre varios escenarios:

1. **Independencia probabilística**
   Dos eventos aleatorios A y B son independientes si la ocurrencia de uno no afecta a la probabilidad de que ocurra el otro.
   $$P(A, B) = P(A) \cdot P(B)$$
   Esto implica también que:
   $$P(A|B) = P(A) \text{ y } P(B|A) = P(B)$$
   Es decir, conocer que B ha ocurrido no aporta información sobre A, y viceversa. Interpretación intuitiva:
   - No existe relación causal ni informativa entre los eventos.
   - Observar uno no cambia nuestras creencias sobre el otro.

Independencia y dependencia probabilística

2. **Dependencia probabilística**
   Dos eventos A y B son dependientes si la ocurrencia de uno sí modifica la probabilidad del otro:
   $$P(A|B) \neq P(A)$$
   La observación de un evento aporta información relevante sobre el otro.
   - Existe relación causal, correlación o una causa común.
   - Ver un evento cambia nuestro grado de creencia sobre el otro.

3. **Independencia condicional**
   Dos eventos pueden ser dependientes en general, pero independientes dado un tercer evento C.
   $$P(A, B|C) = P(A|C) \cdot P(B|C)$$
   - Ejemplo: Cuando aumenta el consumo de helado (A), también aumenta el número de asesinatos (B). [Reichenbach, H. (1956)]

Ejercicio 6 ¿Quién es el evento C en el ejemplo?

Detectar la independencia condicional es fundamental en redes bayesianas, porque permite simplificar el modelo sin perder capacidad explicativa.

Tema 6: **Redes Bayesianas**

Redes Bayesianas: Definición y componentes

- **Definición**: Una red bayesiana es un modelo probabilístico gráfico que representa un conjunto de variables aleatorias y sus relaciones de dependencia condicional mediante un grafo dirigido acíclico (DAG).
- Formalmente, una red bayesiana define una distribución de probabilidad conjunta sobre un conjunto de variables $V = \{V_1, V_2, \ldots, V_n\}$, que se factoriza de acuerdo con la estructura del grafo como:

$$P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P(V_i \mid \mathrm{Pa}(V_i))$$

donde $\mathrm{Pa}(V_i)$ denota el conjunto de nodos padres de la variable V_i.

Redes Bayesianas: Definición y componentes (cont.)

- **Modelo probabilístico**: Las redes bayesianas permiten representar y razonar bajo incertidumbre, combinando conocimiento previo, evidencia observada y relaciones de dependencia entre variables. Son especialmente útiles para modelar sistemas complejos, inciertos o parcialmente observables.
- ¡Los modelos son siempre simplificaciones! ¡Son aproximaciones!
  - Pueden no representar a todas las variables o todas las interacciones entre variables.
  - "All models are wrong; but some are useful."

[George E. P. Box (1976)]

Redes Bayesianas: Grafos dirigidos acíclicos

- **Definición**: En un grafo dirigido acíclico que define la estructura de una red bayesiana:
  - Los nodos representan variables aleatorias.
  - Las aristas dirigidas representan dependencias probabilísticas directas.
- La ausencia de una arista entre dos nodos indica una relación de independencia condicional bajo ciertas condiciones.
- Cada variable V_i es condicionalmente independiente de sus no-descendientes dado el valor de sus padres:

$$V_i \perp\!\!\!\perp \text{NoDesc}(V_i) \mid \text{Pa}(V_i)$$

esta propiedad permite representar de forma compacta de la distribución conjunta sobre todas las variables.

Redes Bayesiana: Tablas de probabilidad condicional

- **Definición**: En una red bayesiana, cada nodo V tiene asociada una tabla de probabilidad condicional (CPT) que define la probabilidad del nodo dado sus padres:

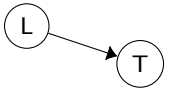
$$P(V \mid \mathrm{Pa}(V))$$

  - Si no tiene padres, la probabilidad es a priori P(V).

- Para cada valor de Pa(V) se debe cumplir:

$$\sum_{x \in V} \mathrm{P(V{=}x \mid Pa(V){=}y)} = 1$$

- **Ejemplo**: Dada la red bayesiana:



L	P(L)
sí	0.3
no	0.7

L	T	P(T | L)
sí	alto	0.8
sí	bajo	0.2
no	alto	0.3
no	bajo	0.7

Observamos que la lluvia $L \in \{si, no\}$ influye en el trafico $T \in \{alto, bajo\}$.

...y a partir de la CPT sabemos que:

- Si llueve, la probabilidad de tráfico alto es del 80 %.
- Si no llueve, esta baja a 30 %.
- etc...

Construcción de modelos bayesianos

- Pasos para **construir un modelo bayesiano**:
  1. Identificar variables relevantes
  2. Definir relaciones de dependencia
  3. Diseñar el DAG
  4. Especificar las CPT (expertos o datos)
  5. Validar el modelo

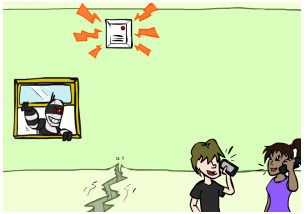
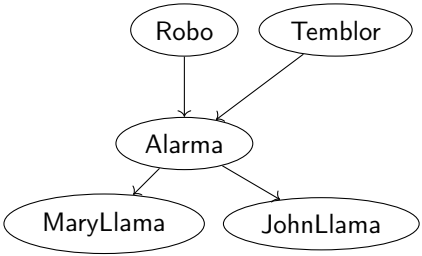


Figura 1: Suena la alarma por robo o terremoto...

- Ejemplo: Alarma (Fig. 1):
  1. Variables relevantes:
     - R: Robo
     - A: Alarma
     - M: Mary llama
     - J: John llama
     - T: Temblor
  2. Dependencias
     - Robo y Temblor son independientes.
     - Dada la variable Alarma, JohnLlama y MaryLlama son condicionalmente independientes del resto de variables.
     - En particular, no dependen de si hubo robo, temblor ni de si la otra persona llamó.

Construcción de modelos bayesianos (cont.)

3. DAG:



4. CPT:

R	P(R)
si	0.001
no	0.999

T	P(T)
si	0.002
no	0.998

R	T	A	P(A|R,T)
si	si	si	0.95
si	si	no	0.05
si	no	si	0.94
si	no	no	0.06
no	si	si	0.29
no	si	no	0.71
no	no	si	0.001
no	no	no	0.999

A	M	P(M|A)
si	si	0.7
si	no	0.3
no	si	0.01
no	no	0.99

A	J	P(J|A)
si	si	0.9
si	no	0.1
no	si	0.05
no	no	0.95

5. Para validar el modelo no necesitamos calcular el CPT conjunta ($2^5 = 32$) es suficiente con estas 20:

$$P(V_1, V_2, \ldots, V_n) = \prod_{i=1}^{n} P(V_i \mid \mathrm{Pa}(V_i))$$

Construcción de modelos bayesianos (cont.)

Ejercicio 7 Dado el ejemplo Alarma de la Figura 1 calcula las siguientes probabilidades:

a) Se ha producido un robo. No se han percibido temblores, pero la alarma ha sonado. Mary ha llamado para avisar, mientras que John no ha dado señales.

b) Dado $\vec{V} = (R, T, A, M, J)$ calcula $P(\vec{V}) = P(si, no, si, si, no)$.

Ejercicio 8 Verifica que todas las tablas CPTs del ejemplo Alarma de la Figura 1 están normalizadas (instancia las fórmulas requeridas para ello).

Tema 7: **Inferencia en Redes Bayesianas**

Inferencia exacta

- La inferencia en redes bayesianas permite determinar la probabilidad de ciertos eventos o estados dado un conjunto de evidencias.
  - Se trata de calcular la probabilidad de nodos desconocidos (ocultos) dado la información observada.
- Ejemplo: ¿Cuál es la probabilidad de que exista malware si la alarma del IDS ha sonado y el tráfico de red es anómalo?

- **Definición**: La inferencia exacta busca calcular probabilidades precisas usando todas las dependencias y la estructura de la red. Métodos comunes:
  - Enumeración: Se calcula toda la distribución y luego se eliminan las variables ocultas.
  - Eliminación de variables: se suman probabilidades marginalizando nodos intermedios.
- Ventaja: resultados exactos.
- Desventaja: computacionalmente costoso en redes grandes.

Inferencia exacta: Algoritmo de Enumeración

- **Algoritmo de Enumeración**: Consiste en calcular todas las combinaciones posibles de las variables no observadas.
  - Muy simple conceptualmente.
  - Muy costoso para redes medianas o grandes (explosión combinatoria).

- **Ejemplo:** Dado el ejemplo de la Alarma de la Figura 1, calcular la probabilidad de que haya habido un robo dado que la alarma ha sonado, Mary ha llamado y John no:

$$P(R=si \mid A=si, M=si, J=no)$$

  - Variables ocultas: Robo (R) y Temblor (T)
  - Variables observadas: Alarma (A=si), Mary (M=si), John (J=no)

Inferencia exacta: Algoritmo de Enumeración (cont.)

- **Ejemplo (cont.):**

Paso 1: Enumerar todas las combinaciones de variables ocultas R y T

R	T	$P(R)$	$P(T)$	$P(A\|R,T)$	$P(M\|A)$	$P(J\|A)$	Numerador
si	*si*	0,001	0,002	0,95	0,7	0,1	$1{,}33 \cdot 10^{-7}$
si	*no*	0,001	0,998	0,94	0,7	0,1	$6{,}55 \cdot 10^{-5}$
no	*si*	0,999	0,002	0,29	0,7	0,1	$4{,}06 \cdot 10^{-5}$
no	*no*	0,999	0,998	0,001	0,7	0,1	$6{,}99 \cdot 10^{-5}$

Numerador (producto de probabilidades para cada fila)

Paso 2: Calcular numerador de Bayes para R=si:

solo sumas las filas donde R=si

Numerador = $\sum_T$ P(R=si, T, A=si, M=si, J=no) =

$$= 1{,}33 \cdot 10^{-7} + 6{,}55 \cdot 10^{-5} \approx 6{,}56 \cdot 10^{-5}$$

Inferencia exacta: Algoritmo de Enumeración (cont.)

- **Ejemplo (cont.):**

Paso 3: Calcular denominador de Bayes:

sumar sobre todas las combinaciones de las variables ocultas R y T.

$$\sum_{R,T} \text{P(R, T, A=si, M=si, J=no)} =$$
$$= \left(6{,}56 \cdot 10^{-5}\right) + 4{,}06 \cdot 10^{-5} + 6{,}99 \cdot 10^{-5} \approx 0{,}000161$$

Paso 4: Inferencia por enumeración: resultado del paso 2 entre el del paso 3.

$$\text{P(R=si} \mid \text{A=si, M=si, J=no)} = \frac{\text{P(R=si,A=si,M=si,J=no)}}{\text{P(A=si,M=si,J=no)}} = \frac{6{,}56 \cdot 10^{-5}}{0{,}000161} \approx 0{,}408$$

Resultado: dadas las evidencias hay un **41 %** de probabilidad de robo.

Inferencia exacta: Algoritmo de Eliminación de variables

- **Algoritmo de Eliminación de Variables**: Consiste en sumar (marginalizar) sistemáticamente las variables no observadas para simplificar el cálculo de probabilidades.

- **Ejemplo:** Dado el ejemplo de la Alarma de la Figura 1, calcular la probabilidad de que haya habido un robo dado que la alarma ha sonado, Mary ha llamado y John no:

$$P(R=si \mid A=si, M=si, J=no)$$

  - Variables ocultas: Robo (R) y Temblor (T)
  - Variables observadas: Alarma (A=si), Mary (M=si), John (J=no)

Inferencia exacta: Algoritmo de Eliminación de variables (cont.)

- **Ejemplo (cont.):**

Paso 1: Factorización de la probabilidad conjunta:

$$P(R,T,A,M,J) = P(R) \cdot P(T) \cdot P(A \mid R,T) \cdot P(M \mid A) \cdot P(J \mid A)$$

Paso 2: Elegir orden de eliminación:

Eliminamos T primero, por lo tanto calculamos f(R,A,M,J).

Paso 3: Marginalización sobre T.

$$f(R,A,M,J) = \sum_T P(R) \cdot \Big(P(T) \cdot P(A \mid R,T) \Big) \cdot P(M \mid A) \cdot P(J \mid A) =$$
$$= P(R) \cdot \Big(P(T{=}si) \cdot P(A \mid R,T{=}si) + P(T{=}no) \cdot P(A \mid R,T{=}no) \Big) \cdot P(M \mid A) \cdot P(J \mid A)$$

Inferencia exacta: Algoritmo de Eliminación de variables (cont.)

- **Ejemplo (cont.):**

Paso 4: Sustituir probabilidades numéricas para $R = si$ y $R = no$.

$$f(R = \text{si}, A = \text{si}, M = \text{si}, J = \text{no}) = \\ = 0{,}001 \cdot \big(0{,}002 \cdot 0{,}95 + 0{,}998 \cdot 0{,}94\big) \cdot 0{,}7 \cdot 0{,}1 \approx 6{,}56 \cdot 10^{-5}$$

$$f(R = \text{no}, A = \text{si}, M = \text{si}, J = \text{no}) = \\ = 0{,}999 \cdot \big(0{,}002 \cdot 0{,}29 + 0{,}998 \cdot 0{,}001\big) \cdot 0{,}7 \cdot 0{,}1 \approx 1{,}105 \cdot 10^{-4}$$

Paso 5: Calcular la probabilidad condicionada

$$\text{P(R=si} \mid \text{A=si,M=si,J=no)} = \frac{\text{f(R=si, A=si,M=si,J=no)}}{\text{f(A=si,M=si,J=no)}} = \frac{6{,}56 \cdot 10^{-5}}{6{,}56 \cdot 10^{-5} + 1{,}105 \cdot 10^{-4}} \approx 0{,}373$$

Resultado: dadas las evidencias hay un **37 %** de probabilidad de robo.[1]

[1] La diferencia con respecto lo obtenido con Enumeración se debe a redondeo interno.

Inferencia Exacta: 'Implementación' en Python

```python
# Probabilidad conjunta P(R,T,A,M,J)
def P(r, t, a, m, j):
  return ( P_R[r] *
           P_T[t] *
           P_cond(a, P_A[(r,t)]) *
           P_cond(m, P_M[a]) *
           P_cond(j, P_J[a]) )
# Factor f(R,A,M,J) eliminando T
def f(r, a, m, j):
  suma_T = (
    P_T['si'] * P_cond(a, P_A[(r,'si')]) +
    P_T['no'] * P_cond(a, P_A[(r,'no')])
  )
  return ( P_R[r] *
           suma_T *
           P_cond(m, P_M[a]) *
           P_cond(j, P_J[a]) )
# Probabilidades condicionales binarias
def P_cond(v, p_si):
    return p_si if v == 'si' else 1 - p_si
```

```python
# Probabilidades marginales / CPT
P_R = {'si': 0.001, 'no': 0.999}
P_T = {'si': 0.002, 'no': 0.998}
P_A = {('si','si'): 0.95,
       ('si','no'): 0.94,
       ('no','si'): 0.29,
       ('no','no'): 0.001}   # P(A=si | R,T)
P_M = {'si': 0.7, 'no': 0.01}  # P(M=si | A)
P_J = {'si': 0.9, 'no': 0.05}  # P(J=si | A)
# Calculo por Enumeración
n1 = sum(P('si',t,'si','si','no') for t in ['si','no'])
x1 = sum(P('no',t,'si','si','no') for t in ['si','no'])
P_enum = n1 / (n1+x1)
# Calculo por Eliminación de Variables
n2,x2 = f('si', 'si','si','no'), f('no', 'si','si','no')
P_elim = n2 / (n2+x2)

desc = "P(R=si | A=si, M=si, J=no)"
print(f"Enumeración:        {desc} = {P_enum}")
print(f"Eliminación de Var: {desc} = {P_elim}")
```

- Implementación del ejemplo de la Figura 1. Try it in https://paiza.io.

Inferencia aproximada

- **Definición**: La inferencia aproximada es un conjunto de métodos que estiman las probabilidades en redes bayesianas cuando la inferencia exacta es muy costosa o inviable.
- Se utiliza en redes grandes o con muchas dependencias, donde calcular probabilidades exactas es computacionalmente pesado.
  - Genera una estimación de la probabilidad posterior basada en muestras o simplificaciones del modelo.
- Objetivo: obtener resultados útiles con menor coste computacional que la inferencia exacta.

Inferencia aproximada: Métodos principales

- **Muestreo probabilístico** (Sampling).
  - Se generan múltiples escenarios (muestras) de la red según la distribución conjunta.
  - Se cuentan cuántas veces ocurren los eventos de interés para estimar probabilidades.

  Técnicas comunes:
  - Muestreo directo.
  - Muestreo por rechazo.
  - Muestreo de Gibbs.
  - Monte Carlo (MCMC).
- **Variational Inference**.
  - Se aproxima la distribución posterior compleja mediante una más simple.
  - Optimiza los parámetros de la distribución simplificada para que se acerque a la verdadera posterior.
  - Muy usado en redes grandes o con variables continuas.
- **Loopy Belief Propagation**.
  - Extiende la propagación de creencias a redes con ciclos.
  - No garantiza resultados exactos, pero suele converger a buenas aproximaciones.

Inferencia aproximada: Algoritmos de propagación de creencias

- **Definición**: Los algoritmos de propagación de creencias permiten realizar inferencia en redes bayesianas de manera eficiente, especialmente en redes tipo árbol o de cliques.
- Concepto:
  - Cada nodo envía y recibe mensajes probabilísticos a sus vecinos.
  - Estos mensajes representan la creencia de un nodo sobre otro nodo dado la evidencia observada.
  - Permite calcular probabilidades marginales de manera distribuida.
- Ventajas:
  - Reduce el costo computacional.
  - Distribuido: los cálculos se pueden realizar localmente en cada nodo.
  - Muy eficiente para redes de gran tamaño o de estructura jerárquica.
- Ejemplo: detección de intrusión con sensores distribuidos.
  - Cada sensor envía mensajes sobre la probabilidad de intrusión.
  - Después de varias iteraciones, cada nodo tiene una creencia actualizada de la probabilidad local y global.

Inferencia aproximada: Algoritmos de propagación de creencias

a) **Propagación en árboles**

Tree-structured Belief Propagation

1. Seleccionar un nodo raíz y recorrer el árbol *desde las hojas* (mensaje ascendente).
2. Cada nodo calcula un mensaje basado en sus propias CPT y los mensajes recibidos de sus hijos.
3. Recorrer el árbol desde la raiz hacia los nodos hojas (mensaje descendente) para actualizar creencias en todos los nodos.
4. Una vez finalizada la propagación, cada nodo puede calcular su probabilidad marginal exacta dado la evidencia.

b) **Propagación en grafos con ciclos**

Loopy Belief Propagation

- Extiende el algoritmo a redes que contienen ciclos.
- Los nodos envían mensajes de manera iterativa hasta que los valores convergen o se alcanza un número máximo de iteraciones.
- No garantiza exactitud, pero suele ofrecer aproximaciones útiles.

Tema 8: **Teoría de la decisión**

Fundamentos de la teoría de la decisión

- **Definición**; La teoría de la decisión en Inteligencia Artificial se centra en cómo elegir la mejor acción cuando existe incertidumbre. La idea fundamental es maximizar la utilidad esperada dadas las evidencias.
- Herramientas principales:
  - Redes de decisión.
  - Árboles de decisión.
  - Secuencias de decisiones.
  - Procesos de Decisión de Markov (MDP).

Utilidad: Funciones de utilidad

- **Definición**: La Utilidad es una medida (numérica) que representa el grado de beneficio o preferencia de un estado respecto a otro.
  - Permite comparar decisiones (que conducen a distintos estado) bajo incertidumbre.
- Una Función de utilidad asigna un valor numérico a cada estado posible:

$$U : \text{Estados} \rightarrow \mathbb{R}$$

- Tipos de funciones de utilidad:
  - Lineal: neutral al riesgo, $U(x) = kx$
  - Cóncava: adverso al riesgo, $U(x) = \sqrt{x}$
  - Convexa: amante del riesgo, $U(x) = x^2$

Utilidad: Utilidad Esperada (EU)

- **Definición**: Utilidad Esperada (EU, Expected Utility) se utiliza para decidir la acción óptima bajo incertidumbre:

$$EU(a) = \sum_i P(s_i \mid a) \cdot U(a, s_i)$$

- **Ejemplo**: Decide si conviene actualizar los servidores o no:

Acción a	Estado s	Probabilidad $P(s \mid a)$	Utilidad $U(a, s)$
Actualizar	Seguro	0.8	50
Actualizar	Fallo	0.2	-80
No actualizar	Seguro	0.5	100
No actualizar	Vulnerable	0.5	0

La acción con mayor EU es la decisión óptima.

...en este ejemplo es no actualizar ($24 < 50$).

Modelos de decisión: Redes de decisión

- **Definición**: Las Redes de decisión son una extensión de las Redes Bayesianas que permiten modelar acciones, utilidades y probabilidades.
  [Judea Pearl, 1988]
- Tipos de nodos:
  1. Azar o Probabilidad: representan la incertidumbre, la suma de la probabilidad de los eventos posibles es 1.
  2. Decisión: Quien toma la decisión puede elegir entre varias acciones posibles.
  3. Valor o Utilidad: resultado final de la ruta de decisión (objetivo a maximizar).
- **Objetivo**: Determinar la acción que maximiza la utilidad esperada dada la evidencia disponible.

Modelos de decisión: Redes de decisión (cont.)

- **Ejemplo**: Dada estas CPT's, conviene llevar o dejar el paraguas:

Clima c	Probabilidad $P(c)$
Sol	0.7
Lluvia	0.3

Acción a	Clima c	Utilidad $U(a, c)$
Dejar	Sol	100
Dejar	Lluvia	0
Llevar	Sol	20
Llevar	Lluvia	70

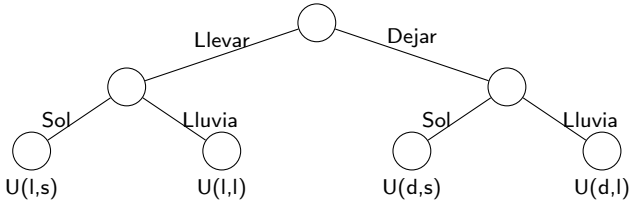
- **Procedimiento** para elegir la decisión óptima:
  1. Instanciamos las evidencias conocidas.
  2. Probamos cada acción como opción.
  3. Calculamos la utilidad esperada de cada acción (con la probabilidad posterior).
  4. Elegimos la acción con mayor utilidad.

- **Ejemplo (cont.)**: La decisión óptima es dejar el paraguas, EU(Dejar) $>$ EU(Llevar):

$$EU(\text{Dejar}) = 0{,}7 \cdot 100 + 0{,}3 \cdot 0 = 70$$
$$EU(\text{Llevar}) = 0{,}7 \cdot 20 + 0{,}3 \cdot 70 = 35$$

Modelos de decisión: Árboles de Decisión

- **Definición** En los árboles de decisión, cada nodo representa un estado del mundo, las ramas representan acciones posibles, y las hojas representan la utilidad.
- Objetivo: Visualizar la toma de decisiones secuenciales.
- **Ejemplo (cont.)**: crea el árbol de decisión del problema del paraguas:



Se calcula la utilidad esperada y se elige la **rama óptima**.[2]

[2]Calculo parecido al ExpectiMinimax (ver Tema 3), hay otras estrategias como MDPs, RL, etc.

Modelos de decisión: Secuencias y procesos de Markov

- **Definición**: Un Procesos de Decisión de Markov (MDP) es un modelo para tomar decisiones secuenciales cuando los resultados dependen de varios pasos en el tiempo:
  - Estados $S = \{s_0, s_1, ..., s_n\}$
  - Acciones $A = \{a_0, a_1, ..., a_m\}$
  - Transición de estados $P(s' \mid s, a)$
  - Recompensas/utilidad $R(s, a, s')$
- **Ejemplo:** Decidir como acelerar depende de la temperatura del motor.

E. actual s	Acción a	E. siguiente s'	Recompensa $R(s, a, s')$
Frío	Fuerte	Caliente	2
Caliente	Fuerte	Sobrecalentado	-10
Frío	Suave	Frío	1
...			

Modelos de decisión: Secuencias y procesos de Markov (cont.)

- Conceptos básicos: Para elegir la mejor acción en cada estado, usamos la política, y para evaluar sus efectos los valores de estado y de estado-acción.
  - Política π: asigna una acción a cada estado. $\pi : S \to A$
    - La política óptima maximiza la utilidad esperada a largo plazo.
  - Valor de un estado: utilidad esperada empezando en s

    $$V^*(s) = \max_a Q^*(s, a)$$
  - Valor de un estado-acción: utilidad esperada con acción a desde s

    $$Q^*(s, a) = \sum_{s'} P(s' \mid s, a)\Big(R(s, a, s') + \gamma \cdot V_k(s')\Big) \qquad \gamma \leq 1$$
- **Procedimiento**: La resolución básica consiste en:
  1. Inicializar $V_0(s) = 0$
  2. Repetir hasta convergencia: $V_{k+1}(s) = \max_a Q^*(s, a)$

Modelos de decisión: Secuencias y procesos de Markov (cont.)

- **Ejemplo (cont.)**: Dado el grafo de la Figura 2 que modela el problema de como acelerar con un MDP (con $\gamma = 1$):

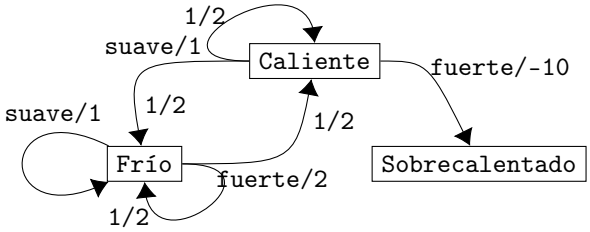


Figura 2: Conviene acelerar fuerte o suave.

a) Calcular V* (y Q*) de forma iterativa.[3]

$V_0(F) = 0 \quad V_0(C) = 0 \quad V_0(S) = 0$

$Q_1(C, f) = -10$
$Q_1(C, s) = \frac{1}{2}(1+0) + \frac{1}{2}(1+0) = 1$
$Q_1(F, f) = \frac{1}{2}(2+0) + \frac{1}{2}(2+0) = 2$
$Q_1(F, s) = 1 + 0 = 1$
$Q_1(S, \_) = 0$ (nodo terminal $V_x(S) = 0$)

$V_1(F) = 2 \quad V_1(C) = 1$

$Q_2(C, f) = -10$
$Q_2(C, s) = \frac{1}{2}(1+2) + \frac{1}{2}(1+1) = 2{,}5$
$Q_2(F, f) = \frac{1}{2}(2+1) + \frac{1}{2}(2+2) = 3{,}5$
$Q_2(F, s) = 1 + 2 = 3$

$V_2(F) = 3{,}5 \quad V_2(C) = 2{,}5$

$V_3(F) = 5 \quad V_3(C) = 4{,}25 \quad \ldots$

[3] Notación: $s \in$ {F=Frio, C=Caliente, S=Sobrecalentado}; $a \in$ {f=fuerte, s=suave}

Modelos de decisión: Secuencias y procesos de Markov (cont.)

- **Ejemplo (cont.)**

b) La iteración de valores no converge porque, con $\gamma = 1$, los ciclos con recompensas positivas hacen que $V(s)$ y $Q(s, a)$ crezcan indefinidamente.

c) Aún así tenemos política óptima:

$$\pi^*(C) = s \quad \pi^*(F) = f$$

NOTA: MDP calcula políticas completas,
ExpectiMinimax solo calcula la acción para un único estado...
...pero ¿si no conocemos $P(s' \mid s, a)$ ni $R(s, a, s')$?[4]

Ejercicio 9 Dado el grafo de la Figura 2, que modela el problema de como acelerar con un MDP, calcular V* (y Q*) de forma iterativa con $\gamma = 0{,}5$.

[4] En el Tema 11 veremos Aprendizaje por Refuerzo para aprender políticas óptimas interando con el entorno.



Universidad
Rey Juan Carlos

Escuela Técnica Superior
Ingeniería Informática

Módulo IV: **Aprendizaje automático**

Asignatura: Inteligencia Artificial

Joaquín Arias

Universidad Rey Juan Carlos
Escuela Técnica Superior de Ingeniería Informática

Curso 2025-2026

Índice I

Tema 9: **Aprendizaje probabilístico**
- **Aprendizaje automático**
  - Introducción
  - Modelos generativos y discriminativos
  - Clasificación probabilística
- **Naive Bayes**
  - Algoritmo
  - 'Implementación' en Python
- **Evaluación de modelos**



Copyright ©2026 Joaquín Arias. Esta obra, basada en trasparencias de Sascha, Holger y Alberto (URJC'23) & de Carmen y Paco (URJC'21), está bajo la licencia CC BY-SA 4.0, **Creative Commons Atribución-CompartirIgual 4.0 Internacional**. Cómo citar esta obra: Arias, Joaquín y Fidilio-Allende, Luciana (2026). ***LIBRO: Inteligencia Artificial, por y para la ciberseguridad***. BURJC, Madrid.

Índice II

Tema 10: **Aprendizaje inductivo**
- **Fundamentos del aprendizaje inductivo**
- **Árboles de decisión**
  - Fundamentos
  - Algoritmo ID3
- **Sobreajuste y generalización**
- **Validación cruzada**

Índice III

Tema 11: **Aprendizaje por refuerzo**
- **Fundamentos**
  - Definición y formalización básica
  - Interacción agente-entorno
  - Motivación, aprendizaje sin modelo
- **Resolución basada en modelo**
  - El entorno como MDP
  - Algoritmo de iteración de valores
- **Aprendizaje sin modelo**
  - Fundamentos y Q-learning
  - Q-learning para entornos deterministas
  - Q-learning para entornos no deterministas

Tema 9: **Aprendizaje probabilístico**

Aprendizaje automático: Introducción

- **Definición**: Rama de la inteligencia artificial que permite a las máquinas aprender de los datos sin ser explícitamente programadas.
  - Se basa en modelos matemáticos y estadísticos para capturar relaciones en los datos.
- Objetivo principal: hacer predicciones o tomar decisiones basadas en patrones aprendidos.
- Categorías de aprendizaje automático:
  - Aprendizaje Supervisado: Datos con etiquetas conocidas. Ejemplo: Clasificación de correos como spam/no spam.
  - Aprendizaje No Supervisado: Datos sin etiquetas, se buscan patrones o agrupamientos. Ejemplo: Segmentación de malware.
  - Aprendizaje por Refuerzo: Agente aprende mediante prueba y error con recompensas. Ejemplo: Malware que encuentra vulnerabilidades.

Aprendizaje automático: Introducción (cont.)

- Componentes Clave:
  - **Datos:** Información histórica o de entrenamiento.
  - **Modelo:** Representación matemática de patrones en los datos.
  - **Algoritmo de Aprendizaje:** Procedimiento para ajustar el modelo a los datos.
  - **Evaluación:** Medición de desempeño del modelo (precisión, error, F1-score, etc.).
- Ventajas del Aprendizaje Automático:
  - Capacidad de automatizar tareas complejas.
  - Adaptación a cambios en los datos.
  - Identificación de patrones no evidentes para humanos.

Aprendizaje automático: Modelos generativos y discriminativos

- **Modelos Generativos**: Intentan modelar la distribución conjunta $P(X, Y)$, donde X son características y Y etiquetas.
  - Ejemplo: Naive Bayes, Modelos de mezcla de Gaussianas (GMM).
  - Ventaja: Se pueden generar nuevos datos y manejar datos faltantes.
- **Modelos Discriminativos**: Modelan directamente $P(Y|X)$, la probabilidad de la etiqueta dada la entrada.
  - Ejemplo: Regresión logística, Redes Bayesianas discriminativas.
  - Ventaja: Normalmente mejor rendimiento predictivo cuando no interesa modelar los datos de forma completa.

Aprendizaje automático: Clasificación probabilística

- **Definición:** Asignar a cada instancia x la clase y con la mayor probabilidad condicional dado x:

$$\hat{y} = \arg\max_{y \in Y} P(Y = y \mid X = x)$$

- Ejemplo: Naive Bayes.[1]

$$P(Y \mid X_1, X_2, \ldots, X_n) \propto P(Y) \prod_{i=1}^{n} P(X_i \mid Y)$$

  - Supone independencia condicional de las características X_i.
  - La clase con mayor $P(Y|X)$ se asigna como predicción.

Relación con temas anteriores:

- **Representación probabilística:** Se basa en probabilidades condicionales y la regla de Bayes.
- **Redes Bayesianas:** Pueden usarse como modelo para representar dependencias entre variables.
- **Inferencia:** Se realiza inferencia para calcular $P(Y|X)$ a partir del modelo.
- **Teoría de la decisión:** La clase con mayor probabilidad puede considerarse la acción óptima bajo pérdida cero-uno.

[1] Usamos $\propto$ porque conserva el orden entre clases de la probabilidad posterior, aunque no esté normalizada.

Naive Bayes: Algoritmo

- **Algoritmo Naive Bayes**:
  1. Convierte los datos en tabla de frecuencias.
  2. Crea tabla de probabilidades.
  3. Calcula probabilidad posterior de cada clase (sin normalizar).
  4. Predicción: La clase con la probabilidad posterior más alta.
- **Ejemplo**: Predecir si se juega al tenis cuando hace sol a partir de las observaciones de la Figura 1:

Clima	s	n	l	l	s	s	n	l	n	n
Juegan	−	+	+	−	+	+	+	−	+	+

Figura 1: Clima y partidos disputados (o no) en US Open.

1. Tabla de frecuencias

Clima	Juegan	
	si (+)	no (−)
Nublado (n)	4	0
Sol (s)	2	1
Lluvia (l)	1	2

2. Tabla de Probabilidades

Clima	Juegan		P(C)
	si	no	
Nublado	4	0	0.4
Sol	2	1	0.3
Lluvia	1	2	0.3
P(J)	0.7	0.3	

Naive Bayes: Algoritmo (cont.)

- **Ejemplo (cont.)**

3. Calcular P(si | Sol) y P(no | Sol), usando:

$$P(Y \mid X_1, X_2, \ldots, X_n) \propto P(Y) \prod_{i=1}^{n} P(X_i \mid Y)$$

Nota que con solo un X_i no es muy útil...

- $P(si \mid Sol) \propto P(si) \cdot P(Sol \mid si) =$
  $= \frac{\#si}{\#total} \cdot \frac{\#\{si \wedge sol\}}{\#si} = \frac{7}{10} \cdot \frac{2}{7} = 0{,}2$
- $P(no \mid Sol) \propto P(no) \cdot P(Sol \mid no) =$
  $= \frac{\#no}{\#total} \cdot \frac{\#\{no \wedge sol\}}{\#no} = \frac{3}{10} \cdot \frac{1}{3} = 0{,}1$

4. Predicción: SI, cuando hace sol se juega.
   P(si | Sol) > P(no | Sol)

Ejercicio 1 Usando los datos de la Figura 1, estimar con Naive Bayes la probabilidad de jugar al tenis cuando llueve.

Ejercicio 2 Explica por qué Naive Bayes no aporta ventajas con un solo atributo.

Ejercicio 3 Razona por qué no es necesario normalizar la probabilidad posterior.

Ejercicio 4 Justifica por qué Naive Bayes es más útil con varios atributos.

Naive Bayes: 'Implementación' en Python

```
1  from sklearn.naive_bayes import CategoricalNB
2  # Datos del problema (mapeado a números
3  Atributos = [[2],[1],[0],[0],[2],[2],[1],[0],[1],[1]]
4  Clases    = [ 0 , 1 , 1 , 0 , 1 , 1 , 1 , 0 , 1 , 1 ]
5  # Aprendizaje usando Naive Bayes
6  modelo_nb = CategoricalNB()
7  modelo_nb.fit(Atributos, Clases)
8  # Inferencia ¿Se juegan si hace sol? ¿Y si llueve?
9  Preguntas    = [[2],[0]]
10 Predicciones = modelo_nb.predict(Preguntas)
11 # Imprime resultados
12 for pregunta, prediccion in zip(Preguntas, Predicciones):
13     print(f"Clima: {pregunta[0]} -> ¿Se juega?: {prediccion}")
```

- Código en Python del problema del tenis de la Figura 1:
  - Los datos deben mapearse a valores numéricos positivos.
  - `Atributos` y `Preguntas` son listas de observaciones, donde cada es una lista de atributo(s).
  - `pregunta[0]` es el primer atributo de la observación.
  - Try it in https://paiza.io.

Ejercicio 5 Predecir usando el código si se juega con clima nublado.

Ejercicio 6 Calcula e imprime la probabilidad de la predicción.[2]

[2] Consulta documentación en https://scikit-learn.org

Evaluación de modelos

- Tras entrenar un modelo probabilístico, es crucial medir su desempeño.
- La evaluación permite:
  - Comparar modelos.
  - Detectar sobreajuste (*overfitting*).
  - Ajustar parámetros y seleccionar el modelo. óptimo
- Para problemas de clasificación probabilística se usan métricas clásicas:
  - **Exactitud (Accuracy)**: proporción de predicciones correctas.
  - **Precisión (Precision)**: proporción de positivos predichos correctos.
  - **Sensibilidad (Recall0**: proporción de positivos reales correctos.
  - **F1-Score**: media armónica de precisión y recall.
- Lo aplicaremos, p.ej., para evaluar Naive Bayes.

Evaluación de modelos (cont.)

Matriz de Confusión:

- Herramienta visual:
  - Filas: clases reales.
  - Columnas: clases predichas
- Ejemplo: 2 clases (Positivo / Negativo)

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix}$$

  - TP: Verdaderos positivos
  - TN: Verdaderos negativos
  - FP: Falsos positivos
  - FN: Falsos negativos

Ejercicio 7 Calcula las métricas de evaluación:

a) Accuracy.
b) Precision.
c) Recall.
d) F1.

Para el problema del tenis con los observaciones y predicciones de la Figura 2.

Clima	s	n	l	l	s	s	n	l	n	n
Juegan	−	+	+	−	+	+	+	−	+	+
Predicción	+	+	−	−	+	+	+	−	+	+

Figura 2: Clima, partidos disputados (o no) y predicción.

Evaluación de modelos (cont.)

```
14  # (cont.) implementación Naive Bayes
15  from sklearn import metrics
16  import matplotlib.pyplot as plt
17  # Evaluación usando los mismos datos, NO HACER!
18  P_todo = modelo_nb.predict(Atributos)
19  # Calculamos distintas métricas
20  accuracy = metrics.accuracy_score(Clases, P_todo)
21  precision = metrics.precision_score(Clases, P_todo)
22  recall = metrics.recall_score(Clases, P_todo)
23  f1 = metrics.f1_score(Clases, P_todo)
24  matrix = metrics.confusion_matrix(Clases, P_todo)
25  # Imprimimos los resultados
26  print(f"\nExactitud (Accuracy): {accuracy:.2f}")
27  print(f"Precisión (Precision): {precision:.2f}")
28  print(f"Sensibilidad (Recall): {recall:.2f}")
29  print(f"F1-Score: {f1:.2f}")
30  print(f"\nMatriz de Confusión:\n {matrix}")
```

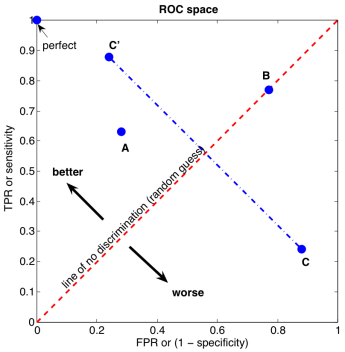
- Añadir al código en Python (pág. 11) del problema del tenis de la Figura 1.
  - Try it in https://paiza.io.
- NOTA: Los datos de Entrenamiento y Prueba no deben coincidir... (pág. 27)

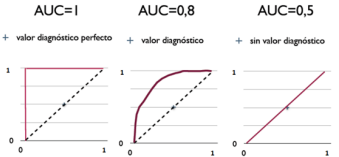
Ejercicio 8 Comprobar que los resultados coinciden con los calculados en el Ejercicio 7 .

Evaluación de modelos (cont.)

Curvas ROC y AUC

- Para clasificadores probabilísticos, la probabilidad predicha permite variar el umbral.
- **Curva ROC**:
  (Receiver Operating Characteristic)
  - Eje X: Tasa de falsos positivos (FPR)
  - Eje Y: Tasa de verdaderos positivos (TPR)
- **AUC**: (Area Under the Curve)
  - Medida global de desempeño
  - AUC = 1: perfecto.
  - AUC = 0.5: aleatorio.





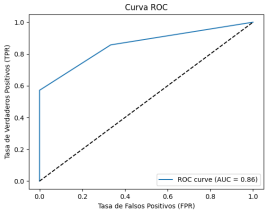
Evaluación de modelos (cont.)

```
31  # (cont.) implementación Naive Bayes + metrics
32  import matplotlib.pyplot as plt
33  # Necesitamos probabilidades predichas
34  probabilidades = modelo_nb.predict_proba(Atributos)[:,1]
35  fpr, tpr, thresholds = roc_curve(Clases, probabilidades)
36  auc = roc_auc_score(Clases, probabilidades)
37  # Imprime AUC
38  print(f"\nAUC: {auc:.2f}")
39  # Dibujar curva ROC
40  plt.figure()
41  plt.plot(fpr, tpr, label=f'ROC curve (AUC = {auc:.2f})')
42  plt.plot([0,1], [0,1], 'k--')  # diagonal de referencia
43  plt.xlabel('Tasa de Falsos Positivos (FPR)')
44  plt.ylabel('Tasa de Verdaderos Positivos (TPR)')
45  plt.title('Curva ROC')
46  plt.legend(loc='lower right')
47  plt.show()
```

- Añadir al código en Python (págs. 11 y 14) del problema del tenis de la Figura 1.

`[:,1]` Índice de la probabilidad de la clase positiva.

- Try it in https://pythononline.net/



Tema 10: **Aprendizaje inductivo**

Fundamentos del Aprendizaje Inductivo

- **Definición**: El aprendizaje inductivo es el proceso mediante el cual un sistema aprende una hipótesis general a partir de ejemplos particulares.
  - Se parte de un conjunto de datos etiquetados.
  - Se infiere una función h que aproxima una función objetivo f:
    $$f : \vec{X} \rightarrow Y.$$
  - El objetivo es generalizar correctamente a nuevos casos no observados.
- **Esquema formal**:
  $$D = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2), \ldots, (\vec{x_n}, y_n)\}$$
  Aprender una hipótesis $h \in H$ tal que:
  $$h(\vec{x}) \approx y$$

Árboles de Decisión: Fundamentos

- **Definición**: Modelo predictivo que utiliza una estructura jerárquica de decisiones para clasificar o predecir valores:
  - Cada nodo interno representa una prueba sobre un atributo.
  - Cada rama representa el resultado de la prueba.
  - Cada hoja representa una clase o valor predicho.
- **Criterios de división**:
  - Entropía e Información Ganada. (ID3, C4.5)
  - Índice Gini. (CART)
- **Ventajas:**
  - Interpretables.
  - Manejan datos categóricos y numéricos.
  - No requieren supuestos fuertes sobre la distribución.

Árboles de Decisiones: Algoritmo ID3

```
1  ALGORITMO ID3(Nodo, Ejemplos, Atributos)
2     SI termina(Nodo, Ejemplos, Atributos) ENTONCES
3         crea_Hoja(Nodo)
4     SINO
5         Aj ← mejorAtributo(Ejemplos, Atributos)
6         Etiqueta(Nodo) ← Aj
7
8         PARA CADA v ∈ Dominio(Aj) HACER
9             Hijo ← crea_Hijo(Nodo)
10            crea_Arista(<Nodo,v,Hijo>)
11            Es_v ← { e ∈ Ejemplos | Aj(e) = v }
12            ID3(Hijo, Es_v, Atributos \ {Aj} )
```

- Algoritmo depende de dos funciones/preguntas clave:
  `mejorAtributo/2`: ¿Cómo seleccionar el mejor atributo?
  `termina/3`: ¿Cuándo determinar si la rama debe terminar?

- Construir recursivamente el árbol desde el nodo raíz.

a) Si el nodo es terminal:
   - Etiquetarlo como hoja.

b) Si no:
   1. Seleccionar el atributo más relevante `Aj` y etiquetar el nodo.
   2. Crear un hijo para cada valor `v` en el dominio de `Aj`.
   3. Para cada hijo, construir `Es_v` con los ejemplos del nodo con `v` en `Aj`.
   4. Repite el proceso para todos los hijos.

Árboles de Decisiones: Algoritmo ID3: Seleccionar mejor Atributo

- **Definición**: El mejor atributo es aquel que:
  - Agrupa mejor los ejemplos según la clase que queremos aprender.
  - Reduce el "desorden" de los posibles valores de clase.

- **Ejemplo**: Decidir si el banco acepta la solicitud de tarjeta de crédito.

	edad	saldo	Tarjeta
E1:	< 18	bajo	No
E2:	< 18	alto	No
E3:	18-25	medio	Si
E4:	18-25	alto	Si
E5:	18-25	medio	No

a) Para saldo tenemos:
   - alto $\rightarrow$ {E2, E4} {No, Si}
   - medio $\rightarrow$ {E3, E5} {Si, No}
   - bajo $\rightarrow$ {E1} {No}

b) Para edad tenemos:
   - $< 18 \rightarrow$ {E1, E2} {No, No}
   - medio $\rightarrow$ {E3, E4, E5} {Si, Si, No}

A ojo parece que edad clasifica mejor.

Árboles de Decisiones: Algoritmo ID3: Seleccionar mejor Atributo (cont.)

```
1  FUNCION mejorAtributo(Ejemplos, Atributos):
2     # Calcular entropía global:
3     E_global ← −∑_c p_c(Ejemplos) log2 p_c(Ejemplos)
4     # Calcular ganancia de cada atributo Aj:
5     PARA CADA Aj ∈ Atributos HACER
6         PARA CADA v ∈ Dominio(Aj) HACER
7             S_v ← { e ∈ Ejemplos | Aj(e) = v }
8             E_Aj_v ← −∑_c p_c(S_v) log2 p_c(S_v)
9         E_Aj ← ∑_v p_S_v(Ejemplos) E_Aj_v
10        Ganancia(Aj) ← E_global - E_Aj
11    # Devuelve el Atributo con mayor ganancia
12    DEVUELVE  arg máx_{Aj ∈ Atributos}  Ganancia(Aj)
```

- Elegimos el atributo con mayor Ganancia, basada en la Entropía (desorden) global/local. [Shannon, 1948]
  - Sea $p_v(C)$ la probabilidad de tener v en C:

$$E(C) = -\sum_v p_v(C)\ log_2\ p_v(C)$$

- **Ejemplo (cont.)**

$$E\_global = -\sum_{v \in \{Si,No\}} p_v(C) \cdot log_2\ p_v(C) =$$

$$= -\left(\tfrac{2}{5} \cdot log_2\ \tfrac{2}{5} + \tfrac{3}{5} \cdot log_2\ \tfrac{3}{5}\right) = 0,97$$

Ganancia de `saldo` y `edad`:

$E\_saldo\_alto = \ldots = 1$

$E\_saldo\_medio = \ldots = 1$

$E\_saldo\_bajo = \ldots = 0$

$E\_saldo = \frac{2}{5} \cdot 1 + \frac{2}{5} \cdot 1 + \frac{1}{5} \cdot 0 = 0{,}8$

$Ganancia(saldo) = 0{,}97 - 0{,}8 = 0{,}17$

$E\_edad\_< 18 = \ldots = 0$

$E\_edad\_18\text{-}25 = \ldots = 0{,}91$

$E\_edad = \frac{2}{5} \cdot 0 + \frac{3}{5} \cdot 0{,}91 = 0{,}55$

$Ganancia(edad) = 0{,}97 - 0{,}55 = \mathbf{0{,}42}$

Efectivamente `edad` clasifica mejor.

Árboles de Decisiones: Algoritmo ID3: Determinar si es terminal

```
1  FUNCION termina(Nodo, Ejemplos, Atributos):
2     # No hay ejemplos
3     SI Ejemplos = ∅ ENTONCES
4         Etiqueta(Nodo) ← mejorClase(Padre(Nodo))
5         DEVUELVE True
6     # Todos los ejemplos son de la misma clase
7     PARA CADA c ∈ Clases HACER
8         SI p_c(Ejemplos) = 1 ENTONCES
9             Etiqueta(Nodo) ← c
10            DEVUELVE True
11    # No quedan atributos
12    SI Atributos = ∅ ENTONCES
13        Etiqueta(Nodo) ← mejorClase(Ejemplos)
14        DEVUELVE True
15    # En cualquier otro caso no es terminal
16    DEVUELVE False
```

- Una rama debe terminar (devuelve `True`) si:
  - No hay ejemplos para clasificar.
  - Todos los ejemplos son de la misma clase.
  - o No hay atributos con los que clasificar.

`mejorClase(S)` devuelve:

- La clase mayoritaria de `S`.
- o Una clase aleatoria (o aleatoria ponderada).

`Padre(N)` devuelve el padre del nodo `N`.

Heurísticas Detener expansión cuando:

- Hay una clase claramente mayoritaria.
- No se puede mejorar la clasificación, i.e.:

$$\forall Aj,\ Ganancia(Aj) \leq 0$$

Árboles de Decisiones: Algoritmo ID3: Ejercicios

Ejercicio 9 Adapta el algoritmo `ID3` para detener la expansión (early stopping), aplicando las siguientes heurísticas:

a) La clase mayoritaria supera el umbral del 90 % de los ejemplos.
b) No se puede mejorar la clasificación, *Ganancia* ≤ 0.

Ejercicio 10 Aplique el algoritmo de aprendizaje de árboles de decisión `ID3` al siguiente conjunto de ejemplos. Especifique el árbol aprendido y los valores de ganancia de los atributos elegidos en los diferentes nodos.

edad	saldo	Tarjeta
< 18	bajo	No
< 18	alto	No
18-25	medio	Si
18-25	alto	Si
18-25	medio	No

Árboles de Decisiones: Algoritmo ID3: Ejercicios

Ejercicio 11 Aplique el algoritmo de aprendizaje de árboles de decisión `ID3` al siguiente conjunto de ejemplos. Especifique el árbol aprendido y los valores de ganancia de los atributos elegidos en los diferentes nodos.

Pelo	Altura	Peso	Protección	Quemadura
rubio	media	bajo	no	si
rubio	alta	medio	si	no
moreno	baja	medio	si	no
rubio	baja	medio	no	si
rojo	media	alto	no	si
moreno	alta	alto	no	no
moreno	media	alto	no	no
rubio	baja	bajo	si	no

Sobreajuste y generalización

- **Definición**: Sobreajuste (overfitting) ocurre cuando el modelo se ajusta excesivamente a los datos de entrenamiento, capturando ruido en lugar de patrones generales.
  - Error bajo en entrenamiento.
  - Error alto en datos nuevos.
- **Definición**: La generalización es la capacidad del modelo para funcionar correctamente en datos no vistos.
- Técnicas para evitar sobreajuste (sin perder generalidad):
  - Poda de árboles.
  - Regularización.
  - Aumento de datos.
  - Validación cruzada.

Validación Cruzada

- **Definición**: La Validación Cruzada (Cross-Validation, CV) es una técnica para estimar el rendimiento real de un modelo en datos no vistos:
  - Se divide el conjunto de datos en k particiones (folds).
  - Se entrena con $k-1$ particiones.
  - Se evalúa con la partición restante.
  - Se repite el proceso k veces.
- Cálculo del error promedio en validación cruzada k-fold:

$$Error_{CV} = \frac{1}{k} \sum_{i=1}^{k} Error_i$$

- Ventajas:
  - Uso eficiente de los datos.
  - Reduce la varianza en la estimación del error.

Tema 11: **Aprendizaje por refuerzo**

Fundamentos: Definición y formalización básica

- **Definición**: El aprendizaje por refuerzo (Reinforcement Learning, RL) es un paradigma del aprendizaje automático en el que un agente:
  - Aprende a tomar decisiones mediante prueba y error.
  - Interactúa con el entorno.
  - Trata de maximizar la recompensa acumulada.
- No hay respuestas correctas directas. El agente descubre la mejor estrategia explorando y recibiendo retroalimentación de sus acciones.
- **Formalización**: Se basa en un agente que:
  - Percibe el estado del entorno: $s \in S$.
  - Toma acciones en un estado s: $a \in A(s)$
  - Recibe recompensas al tomar una acción en un estado. $r(s, a)$
  - Aprende una política óptima, la estrategia que maximiza la recompensa total. $\pi(s)$

Fundamentos: Interacción Agente-Entorno

- En el aprendizaje por refuerzo, la interacción entre agente y entorno se describe como un ciclo de interacción:
  1. El agente observa el estado (s_t) del entorno.
  2. Elige una acción (a_t) según su política (π).
  3. El entorno responde proporcionando:
     - Un nuevo estado (s_{t+1})
     - Una recompensa (r_t)
  4. El agente actualiza su política usando esta experiencia.
- **Ejemplo**: Un robot limpiador en una habitación:
  - Estado (s_t): posición actual y suciedad de la habitación.
  - Acción (a_t): moverse, limpiar, girar.
  - Recompensa (r_t): +1 por limpiar, -1 por chocar con un obstáculo.
  - Objetivo: maximizar la limpieza total sin dañar el entorno.

Fundamentos: Motivación, aprendizaje sin modelo

- **Motivación**: El aprendizaje sin modelo es necesario porque no siempre conocemos el modelo del entorno:
  - Del conjunto de estados posibles S sólo los estados visitados.
  - El conjunto de acciones posibles A (y sus relaciones con los estados (s,a) solo es conocido parcialmente.
  - El modelo de transición $T : S \times A \rightarrow Pr(S)$ es desconocido.
  - El modelo de recompensas $R : S \times A \rightarrow \mathbb{R}$ es desconocido.
- Ejemplos:
  - Robot que debe moverse en en un almacén.
  - Laberinto.
  - Jugar al ajedrez.
- **Objetivo:** Aprender la política que maximiza las recompensas acumuladas esperadas (política óptima $\pi* : S \rightarrow A$)

Resolución basada en modelo: El entorno como MDP

- **Definición**: Cuando se conoce el entorno, este se puede modelar como un Proceso de Decisión de Markov (MDP).[3]
- Un MDP está definido por una tupla $\langle S, A, T, R \rangle$:
  - S es el conjunto de estados.
  - A es el conjunto de posibles acciones.
  - $T : S \times A \times S \to [0, 1]$ es un modelo de transición:
    - T: es una distribución de probabilidades sobre el conjunto S.
    - $T(s, a, s')$ (o $P(s' \mid s, a)$): denota la probabilidad de llegar al estado s' si se realiza la acción a en el estado s.
  - $R : S \times A \to \mathbb{R}$ es el modelo de recompensas.
    - $R(s, a)$ denota la recompensa que se recibe al realizar la acción a en el estado s.

[3] Véase el Tema 8 para una introducción previa a los MDP.

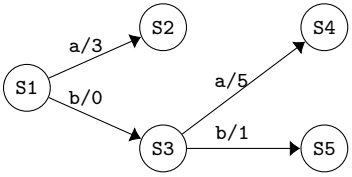
Resolución basada en modelo: El entorno como MDP (cont.)

- **Propiedad de Markov**:

  $$\Pr(s_{t+1}, r_t \mid s_t, a_t) = \Pr(s_{t+1}, r_t \mid s_t, a_t, s_{t-1}, a_{t-1}, \ldots, s_0, a_0)$$

  La probabilidad de un estado y su recompensa depende solo del estado y acción previos, no del historial completo.

- **Ejemplo**: Dado el siguiente grafo resolver el MDP, calculando Q*, V* y π*.[4] Al ser un grafo acíclico su resolución es sencilla.



$Q^*(s_3, a) = 5 + 0 = 5$
$Q^*(s_3, b) = 1 + 0 = 1$
$V(s_3) = \max(5, 1) = 5$
$\pi(s_3) = a$

$Q^*(s_1, a) = 3 + 0 = 3$
$Q^*(s_1, b) = 0 + \max(1, 5) = 5$
$V(s_1) = \max(3, 5) = 5$
$\pi(s_1) = b$

[4] En S4 y S5: Q* y V* son 0 (y no tienen política) porque no hay acciones saliendo de ellos.

Resolución basada en modelo: Algoritmo de Iteración de Valores

```
1  ALGORITMO Iteracion_Valores
2     PARA CADA s ∈ S y a ∈ A(s) HACER
3        Q*(s,a) ← 0   # inicialización
4
5     REPETIR
6        cambio ← 0
7        PARA CADA s ∈ S y a ∈ A(s) HACER
8           old ← Q*(s,a)
9           Q*(s,a) ← R(s,a) +
10                + γ ∑_s' T(s,a,s') · max_b Q*(s',b)
11          cambio ← max(cambio, |old - Q*(s,a)|)
12    HASTA QUE cambio < ϵ
13
14    PARA CADA s ∈ S HACER
15       V*(s) ← max_{a∈A} Q*(s,a)
16       π*(s) ← argmax_{a∈A} Q*(s,a)
17
18    DEVUELVE Q*, V*, π*
```

- Pseudocódigo del algoritmo de iteración de valores para `V*`, `Q*` y $\pi*$.
  - Permite calcular grafos con ciclos.
  - `Q*` se suele inicializar en 0 (pero no solo).
  - $|x|$ devuelve el valor absoluto de x.
  - ϵ valor positivo, define el criterio de parada.

- Hay otros algoritmos para resolver MDPs, p.ej., Iteración de políticas.
- Hay extensiones, p.ej., para entornos con recompensas estocásticas (como el Poker):
  - POMDP: Partially Observable MDP.

Resolución basada en modelo: Algoritmo de Iteración de Valores (cont.)

- **Ejemplo**: Resolver el grafo de la Figura 3 con `Iteracion_Valores`:
  - Inicializando Q* a 0.
  - Orden de ejecución: Estados de s_0 a s_4 y acciones a; b.
  - Factor de descuento $\gamma = 1$.

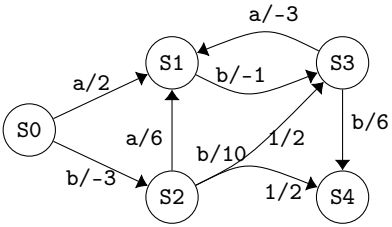


Figura 3: Grafo no determinista.

It.	$Q^*(s_0,a)$	$Q^*(s_0,b)$	$Q^*(s_1,b)$	$Q^*(s_2,a)$	$Q^*(s_2,b)$	$Q^*(s_3,a)$	$Q^*(s_3,b)$
	0	0	0	0	0	0	0
1	2	-3	-1	5	10	-4	6
2	1	7	5	11	13	2	6
3	7	10	5	11	13	2	6
4	7	10	5	11	13	2	6

Estado	s_0	s_1	s_2	s_3	s_4
V^*	10	5	13	6	0
π^*	b	b	b	b	-

Ejercicio 12 Dado el grafo de la Figura 3, resolver con `Iteracion_Valores`, iniciando Q* a 0 y $\gamma = 0{,}9$.

Aprendizaje sin modelo: Fundamentos y Q-learning

- **Definición**: La idea básica del aprendizaje por refuerzo sin conocer el modelo del entorno consiste en:
  - Actuar por "prueba y error".
  - Mejorar la política según las recompensas recibidas.

- **Clasificación de los métodos:**
  1. Libres del modelo: Solo intentan aprenden la política, p.ej.:
     - Q-learning. [Watkins, 1989]
  2. Métodos basados en el modelo: Se aprenden también el modelo de transiciones y recompensas. Con esta información se actualizar la política aprendida.

Aprendizaje sin modelo: Fundamentos y Q-learning (cont.)

- **Definición**: Q-learning aprende una función $Q(s, a)$ que aproxime $Q^*(s, a)$, sin conocer el modelo del entorno.
- La política del agente:

$$\pi(s) = \arg\max_a Q(s, a)$$

  mejora mediante *prueba y error* a lo largo de muchos episodios.
- Es similar al algoritmo `Iteracion_Valores` para resolver MDPs, pero Q-learning no requiere la función de transición T ni de recompensas R.
- Veremos dos versiones de Q-learning, según el tipo de entorno:
  - Deterministas: Cada acción lleva a un único estado y recompensa.
  - No deterministas: Las acciones pueden producir varios estados y recompensas probabilísticas.

Aprendizaje sin modelo: Q-learning para entornos deterministas

```
1  ALGORITMO Q_Learning_v1
2     s ← observar(E_actual)
3     IF s ∉ E_conocidos
4        E_conocidos.añade(s)
5        PARA CADA a ∈ A(s) HACER
6           Q(s,a) ← 0
7
8     MIENTRAS !terminal(s) HACER
9        a ← siguiente_accion(s)
10       ejecutar(a)
11       s' ← observar(E_siguiente)
12       r ← observar(R(s,a,s'))
13       SI s' ∉ E_conocidos ENTONCES
14          E_conocidos.añade(s')
15          PARA CADA b ∈ A(s') HACER
16             Q(s',b) ← 0
17       Q(s,a) ← r + γ · máx_b Q(s',b)
18       s ← s'
```

- Pseudocódigo para algoritmo Q-learning en entornos deterministas.

  `Q` Inicializar (i) de manera pesimista, p.ej., 0 o (ii) con valores mayores a las recompensas esperadas.

  `siguiente_accion` Equilibrar la **exploración** de nuevas acciones con la **explotación** de las mejores acciones encontradas. Algunas estrategias son:

  ***greedy*:** en cada estado elige la mejor acción conocida (si hay empate, aleatoriamente).

  ϵ-***greedy*:** la mejor acción con probabilidad ϵ o aleatoriamente.

  ***SOFTMAX*:** explora más al principio y explota más al final del aprendizaje.

Aprendizaje sin modelo: Q-learning para entornos deterministas (cont.)

- **Ejemplo**: Dado el grafo de la Figura 4, resolver aplicando `Q_Learning_v1`:

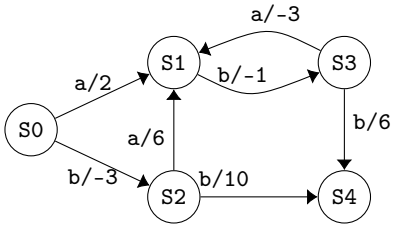


Figura 4: Grafo determinista

Opción A:

- El agente esta en s_0 y sabe $Q(s_0, a) = 2$, $Q(s_0, b) = 0$, $Q(s_1, b) = -1$.
- Factor de descuento $\gamma = 0{,}9$.

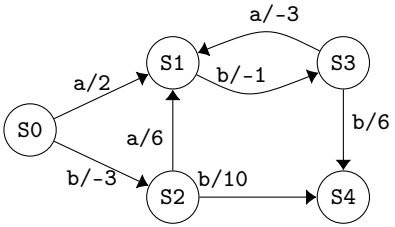
- En s_0 el agente conoce las acciones a y b:
  - Supongamos que elige a y va al estado s_1.
  - Como conoce s_1 actualiza

$$Q(s_0, a) = r + \gamma \cdot max_{\text{acc} \in \{b\}} Q(s_1, \text{acc}) =$$
$$= 2 + 0{,}9 \cdot (-1) =$$
$$= 1{,}1$$

- Ahora el agente esta en el estado s_1:
  - Solo tiene una acción posible, la b y va al estado s_3, desconocido.
  - Inicializa $Q(s_3, a)$ y $Q(s_3, b)$.
  - Actualiza $Q(s_1, b) = \ldots$

...y sigue iterando.

Aprendizaje sin modelo: Q-learning para entornos deterministas (cont.)



Opción B:

- El agente esta en s_0.
- $Q(s, a)$ se inician a 0.
- Selecciona acción con mayor valor (o acción "arriba").
- Factor de descuento $\gamma = 1$.

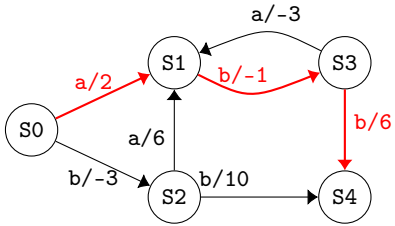
It.	s	$A(s)$	s'	r	$Q(s_0, a)$	$Q(s_0, b)$	$Q(s_1, b)$	$Q(s_3, a)$	$Q(s_3, b)$
	s_0				0	0			
1	s_0	a	s_1	2	2		0		
2	s_1	b	s_3	-1			-1	0	0
3	s_3	a	s_1	-3				-4	
4	s_1	b	s_3	-1			-1		
5	s_3	b	s_4	6					6
	Valores finales				2	0	-1	-4	6

Segundo episodio desde s_0

It.	s	$A(s)$	s'	r	$Q(s_0, a)$	$Q(s_0, b)$	$Q(s_1, b)$	$Q(s_3, a)$	$Q(s_3, b)$
	s_0				2	0	-1	-4	6
1	s_0	a	s_1	2	1				
2	s_1	b	s_3	-1			-5		
3	s_3	b	s_4	6					6
	Valores finales				1	0	5	-4	6

Aprendizaje sin modelo: Q-learning para entornos deterministas (cont.)

Ejemplo (cont.): Tras varios episodios aprendemos la estrategia:



$$Q(s_0, a) = 7$$
$$Q(s_0, b) = 0$$
$$Q(s_1, b) = 5$$
$$Q(s_3, a) = 6$$
$$Q(s_3, b) = -4$$

- Pero no encuentra la óptima $s_0 \to s_2 \to s_1 \to s_3 \to s_4$ con recompensa 8.

Ejercicio 13 Aplicar el algoritmo `Q_Learning_v1` al grafo de la Figura 4 usando:

a) Distintas estrategias de selección de acciones, p.ej., ϵ-greedy.
b) Diferentes inicializaciones de Q, p.ej., $Q(s_0, a) = Q(s_0, b) = 20$.

Aprendizaje sin modelo: Q-learning para entornos no deterministas

```
1  ALGORITMO Q_Learning_v2
2     s ← observar(E_actual)
3     IF s ∉ E_conocidos
4        E_conocidos.añade(s)
5        PARA CADA a ∈ A(s) HACER
6           Q(s,a) ← 0
7
8     MIENTRAS !terminal(s) HACER
9        a ← siguiente_accion(s)
10       ejecutar(a)
11       s' ← observar(E_siguiente)
12       r ← observar(R(s,a,s'))
13       SI s' ∉ E_conocidos ENTONCES
14          E_conocidos.añade(s')
15          PARA CADA b ∈ A(s') HACER
16             Q(s',b) ← 0
17       Q(s,a) ← (1-α)Q(s,a) +
18                + α (r + γ · máx_b Q(s',b))
19       s ← s'
```

- Pseudocódigo para algoritmo Q-learning en entornos NO deterministas.
- Al ejecutar una acción a en s, la transición puede ser estocástica, i.e., $Pr(s' \mid s, a) < 1$:
  - La actualización de $Q(s, a)$ debe incorporar solo parcialmente la información observada.

$\alpha \in (0, 1)$ es la tasa de aprendizaje (learning rate):

- Si es grande acelera el aprendizaje pero puede desestabilizar.
- Normalmente pequeño para dar mas peso a la estimación anterior.

Aprendizaje sin modelo: Q-learning para entornos no deterministas (cont.)

- Q-learning converge a $Q^*(s,a)$ si:
  1. El entorno se modela como un MDP.
  2. La tasa de aprendizaje α_i satisface:

$$\sum_{i=0}^{\infty} \alpha_i = \infty, \qquad \sum_{i=0}^{\infty} \alpha_i^2 < \infty,$$

     es decir, α_i decrece con el tiempo
     p.ej., $\alpha = 1/n(s,a)$.
  3. Cada par (s,a) se visita infinitas veces.
  4. El entorno es estacionario
     i.e., las $Pr(s' \mid s,a)$ no varían.

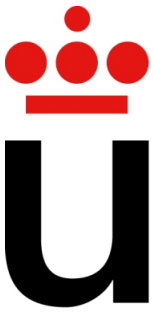
Ejercicio 14 Dado el grafo de la Figura 3, aplicar el algoritmo `Q_learning_v2` considerando:

- Iniciar $Q(s,a)$ en 0, estado inicial S0, $\gamma = 1$, $\alpha = 0{,}1$.
- Repetir 3 episodios, tal que en:

1 y 2: Elegir la acción con mayor Q o b.
3: Igual que antes, salvo que:
En S0 seleccionar acción b.
En S2, con b, la primera transición va S3, luego alterna S4 y S3.

a) Completar la tabla:

It.	s	$A(s)$	s'	r	$Q(s_0,a)$	$Q(s_0,b)$	...
	s_0	...					
1	...						



Universidad
Rey Juan Carlos

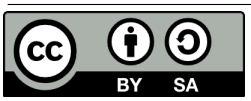
ESCUELA TÉCNICA SUPERIOR
DE INGENIERÍA INFORMÁTICA

Inteligencia Artificial por y para la Ciberseguridad

SELECCIÓN DE EJERCICIOS

1ª EDICIÓN
CURSO 2025-2026

Autores: **Joaquín Arias**
Luciana Fidilio-Allende



Copyright (c) 2026 Joaquín Arias, Luciana Fidilio-Allende. Esta obra está bajo la licencia CC BY-SA 4.0, Creative Commons Atribuciónn-CompartirIgual 4.0 Internacional. Cómo citar esta obra: Arias, Joaquín y Fidilio-Allende, Luciana (2026). *LIBRO: Inteligencia Artificial, por y para la ciberseguridad*. BURJC, Madrid.

Índice

1. Resolución de problemas y búsqueda **3**

2. Representación del conocimiento **10**

3. Aprendizaje automático **14**

A. Soluciones **22**

Agradecimientos

Para elaborar esta obra nos hemos inspirado en transparencias, apuntes y compendios de ejercicios de diversas fuentes, incluyendo trabajos de: Sascha, Holger, Alberto (URJC'23), Carmen, Paco (URJC'21), Francisco Casacuberta (UPV'22).

1. Resolución de problemas y búsqueda

Ejercicio 1.1 El enfoque de los Agentes Inteligentes concibe el objetivo de la Inteligencia Artificial como el intento de construir sistemas

a) Que actúen como humanos.

b) Que actúen de forma racional en un entorno.

c) Que piensen como humanos.

d) Ninguna de las respuestas es correcta.

(Sol. en pág. 22)

Ejercicio 1.2 ¿Para cual(es) de las siguientes tareas pueden construirse agentes basados en algoritmos de búsqueda en espacios de estados (p.e. la búsqueda de coste uniforme)?

a) Encontrar una solución al problema de las Torres de Hanoi.

b) Gestionar el tráfico rodado en una red de autopistas urbanas.

c) Conducir un taxi en las calles de Barcelona.

d) Jugar a las cuatro en raya contra un jugador humano.

(Sol. en pág. 22)

Ejercicio 1.3 ¿Cuáles de las siguientes afirmaciones acerca de los algoritmos de búsqueda no informados es (son) cierta(s)?

a) Los algoritmos de búsqueda no informados requieren de información heurística para que sean óptimos.

b) La búsqueda en anchura es óptima y completa siempre y cuando el coste de los operadores sea constante.

c) La búsqueda en profundidad es óptima y completa siempre que el coste de los operadores sea constante.

d) Tanto la complejidad en tiempo como la complejidad en espacio de la búsqueda en anchura se pueden expresar en función del número de nodos expandidos

(Sol. en pág. 22)

Ejercicio 1.4 El juego de las jarras: una con una capacidad de 3 litros (llamada Tres), y la otra con una capacidad de 4 litros (llamada Cuatro). Inicialmente, Tres y Cuatro están vacías. Cualquiera de ellas puede llenarse con el agua de un grifo. Asimismo, el contenido tanto de Tres como de Cuatro puede vaciarse en una pila. Es posible echar todo el agua de una jarra a la otra. No se dispone de dispositivos de medición adicionales. Se trata de encontrar una secuencia de operadores que deje exactamente 2 litros de agua en Cuatro.

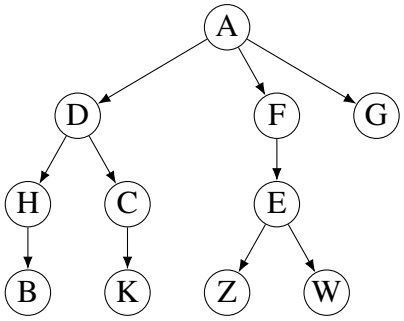
a) Modele este problema como un problema de búsqueda. Con tal fin, defina el estado inicial, el conjunto de estados objetivo, los operadores (especificando sus precondiciones y postcondiciones), así como el coste de cada operador.

b) Caracterice el conocimiento a priori del agente de resolución del problema correspondiente. Facilite ejemplos de los resultados de la función vecinos.

c) Encuentre una solución al problema.

(Sol. en pág. 22)

Ejercicio 1.5 El grafo que se muestra al lado determina un problema de búsqueda. Cada nodo representa un estado; los arcos modelan la aplicación de operadores. Suponga que A es el estado inicial y que K y E son estados meta:



a) Desarrolle el árbol de búsqueda que genera la búsqueda en amplitud, sin filtrar estados repetidos. ¿Cuál de los nodos meta se encuentra primero?

b) Indique el orden en que se expanden los nodos.

c) Ponga el estado de la lista abierta en cada paso del algoritmo.

Ejercicio 1.6 Un hombre se encuentra en la orilla izquierda de un río junto con un lobo, una oveja y una col. Quiere cruzar el río llevando consigo el lobo, la oveja y la col. En la barca sólo hay dos plazas, una de las cuales debe ir ocupada por el hombre. Cada uno de los restantes pasajeros (lobo, oveja, col) ocupa una plaza, de tal modo que sólo uno puede acompañar al hombre en cada viaje. Además, no puede dejar solos en una orilla al lobo con la oveja, ni a la oveja con la col (ni los tres), porque la primera se comería a la segunda en cada paso

a) Si modelizamos el problema como un espacio de estados, donde cada estado se describe como un par de conjuntos, indicando quien(es) se encuentran en cada orilla del río (c = col; o = oveja; l = lobo; h = hombre), el estado inicial del problema es (c,o,l,h,). ¿Cuál seria el estado objetivo?

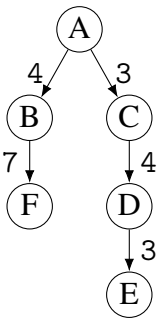
b) Simule la estrategia de búsqueda en amplitud para este problema, asumiendo que se filtran todos los estados repetidos. Expanda los sucesores de los nodos

siguiendo las preferencias del hombre, las cuales se ordenan (de mayor a menor) como sigue: viajar con la oveja, viajar con la col, viajar sólo, viajar con el lobo. Dibuje el árbol de búsqueda correspondiente e indique el orden en el que se exploran los nodos.

c) Simule ahora la estrategia de búsqueda en profundidad para este problema. Dibuje el árbol de búsqueda correspondiente e indique el orden en el que se exploran los nodos, asumiendo que se filtran todos los estados repetidos y siguiendo las preferencias del apartado anterior.

d) Suponga ahora que no se filtra ningún estado repetido. ¿La búsqueda en amplitud y la búsqueda en profundidad seguirían siendo completas? Razone brevemente su respuesta.

Ejercicio 1.7 El grafo que se muestra en la figura describe un problema de búsqueda. Suponga que A es el estado inicial y que F y E son estados meta. Los arcos están etiquetados con el coste real de los operadores.



a) Asigne los valores de la función heurística h*, de modo que resulte ser optimista y no consistente.

b) Desarrolle el árbol de búsqueda que genera el algoritmo A*. ¿Es el algoritmo A* óptimo en este caso?

(Sol. en pág. 23)

Ejercicio 1.8 Considere el problema de los 3-bloques cuyo estado inicial es [[A,C,B]] y estado objetivo es [[C,B,A]].

a) Desarrolle el árbol de búsqueda que expande el algoritmo A*, utilizando la siguiente heurística: h*(n) = número de bloques descolocados.

Nota: Considera que un bloque está descolocado si por debajo no tiene el elemento correcto (bien el bloque deseado o bien la mesa). Filtre los ciclos simples, indique el orden de expansión de los estados y muestre en cada paso los valores de f , g y h*. Suponga que el coste de cada operador es 1.

Ejercicio 1.9 Considere el 8-puzzle cuyo estado inicial y estado meta se muestran en la siguiente figura, y desarrolle el árbol de búsqueda que expande el algoritmo A* utilizando las siguientes heurísticas. Evite ciclos generales, indique el orden de expansión de los estados y muestre en cada paso los valores de h*, g, y f. Cuando puede elegir entre varios nodos para ser expandidos, asuma el "peor caso".

1	2	3
6	4	
8	7	5

Estado inicial

1	2	3
8		4
7	6	5

Estado objetivo

a) h_a*(n) = número de piezas descolocados en n respecto al estado objetivo.

b) h_b*(n) = suma de las distancias de Manhattan de las piezas en n respecto al estado objetivo.

(Sol. en pág. 24)

Ejercicio 1.10 A cuáles de los siguientes juegos puede aplicarse la búsqueda Minimax?

a) Parchís.

b) Juego de las Damas.

c) Comecocos.

d) Tres-en-Raya.

(Sol. en pág. 27)

Ejercicio 1.11 Cuáles de las siguientes afirmaciones acerca del algoritmo Minimax son verdaderas y cuáles son falsas?

a) El algoritmo Minimax realiza una exploración primero en anchura del árbol de juego.

b) El algoritmo Minimax maximiza la utilidad mínima que puede conseguir el jugador max.

c) Puede haber estrategias que funcionan mejor que Minimax si el contrincante no es óptimo.

d) Puede haber estrategias que funcionan mejor que Minimax si el contrincante es óptimo. Con la poda alfa-beta se eliminan nodos que nunca influirán en la decisión final.

(Sol. en pág. 27)

Ejercicio 1.12 La incorporación de la poda alfa-beta en el algoritmo Minimax...

a) ...en algunos casos, puede empeorar la calidad de la solución (i.e. es posible que se elija una jugada peor que el Minimax simple).

b) ...puede mejorar la velocidad con la que se encuentra una solución (i.e. se elige una jugada más rápidamente que el Minimax simple).

c) ...suele mejorar la calidad de la solución (i.e. se elige una mejor jugada que el Minimax simple).

d) ...produce siempre la misma solución (i.e. se elige la misma jugada que el Minimax simple).

(Sol. en pág. 27)

Ejercicio 1.13 Cuáles de las siguientes afirmaciones acerca del algoritmo ExpectiMinimax son ciertas y cuáles son falsas?

a) Los nodos azar representan una mala jugada del jugador MIN.

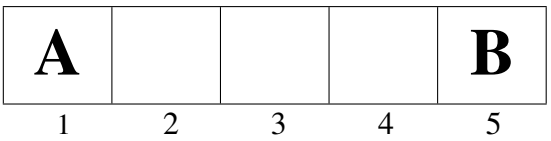
b) Los nodos azar representan una buena jugada del jugador MIN.

c) Los nodos azar representan un evento aleatorio.

d) Si un nodo azar tiene k vecinos equiprobables entonces, al evaluar el árbol de juego, se pueden sumar las evaluaciones de los vecinos del nodo azar y dividir dicha suma por k.

(Sol. en pág. 27)

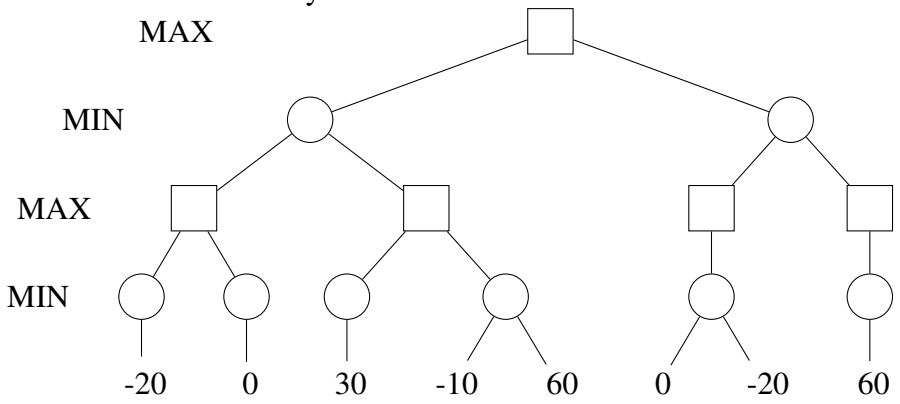
Ejercicio 1.14 Considera el siguiente juego: Dos contrincantes (MIN y MAX) manejan una ficha en un tablero como en figura 1. MIN y MAX mueven respectivamente las fichas B y A. Los dos jugadores mueven por turno, empezando por MAX. Cada jugador debe mover su ficha a un espacio vacío adyacente en una u otra dirección. Si el adversario ocupa un espacio adyacente, entonces un jugador puede saltar sobre el adversario al siguiente espacio vacío, si existe. Por ejemplo, si A está sobre 3 y B está sobre 2, entonces MAX puede mover A hacia atrás en el espacio 1. El juego termina cuando un jugador alcanza el extremo opuesto del tablero. Si MAX alcanza el espacio 5 primero, el valor de utilidad para MAX será ∞, si MIN alcanza el espacio 1 primero el valor de utilidad de MAX será $-\infty$.



a) Defina una función de evaluación e(s) para el genérico estado s, donde s es una hoja del árbol, y no necesariamente un estado donde uno de los jugadores gana.

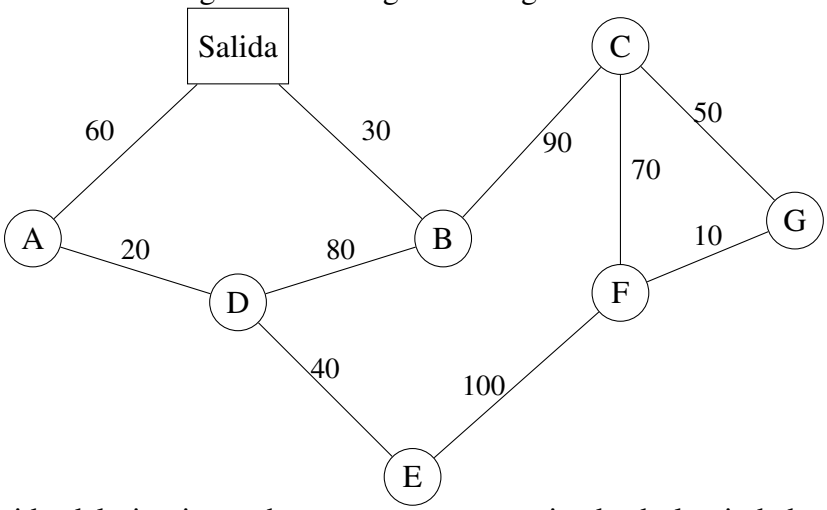
b) Aplica el algoritmo Minimax con suspensión hasta el nivel 6, empezando con el tablero de figura 1 (nótese que empezando con el nodo raíz con etiqueta MAX de nivel 0, las hojas de nivel 6 también tendrán etiqueta MAX). Especifica los valores calculados para cada nodo y determina la mejor jugada para MAX.

Ejercicio 1.15 Considera el árbol en figura 2 de un juego de dos personas, donde los cuadrados son nodos MAX y los círculos son nodos MIN.



a) Aplique el algoritmo Minimax con poda alfa-beta, propaga los valores de evaluación hasta el nodo raíz, marca la mejor jugada para MAX, y marca todos los subárboles que se podan.

Ejercicio 1.16 Dos conductores, el agente y su contrario, se plantean competir sobre el circuito de ciudades de la figura con las siguientes reglas:



- El recorrido del circuito se hace por tramos, partiendo de la ciudad marcada como Salida.
- Los jugadores alternativamente eligen el tramo a recorrer entre aquellos que parten de la ciudad en la que se encuentran. Una vez elegido el tramo, ambos conductores lo recorren y se apunta los kilómetros del tramo el conductor que lo eligió, sin importar quién llega antes a la ciudad de destino. Es decir, si el agente empieza el recorrido y decide ir a B, el agente se apuntará 30 kilómetros. A continuación el contrario debe elegir, partiendo de B, entre ir a C o a D.
- Un conductor no puede ir a una ciudad en la que haya estado antes, por lo que la competición acaba cuando el conductor al que le toca moverse no puede ir a ninguna ciudad no visitada anteriormente.
- Gana el conductor que haya acumulado más kilómetros con los tramos recorri-

dos.

a) Defina una representación eficiente para los estados del juego.

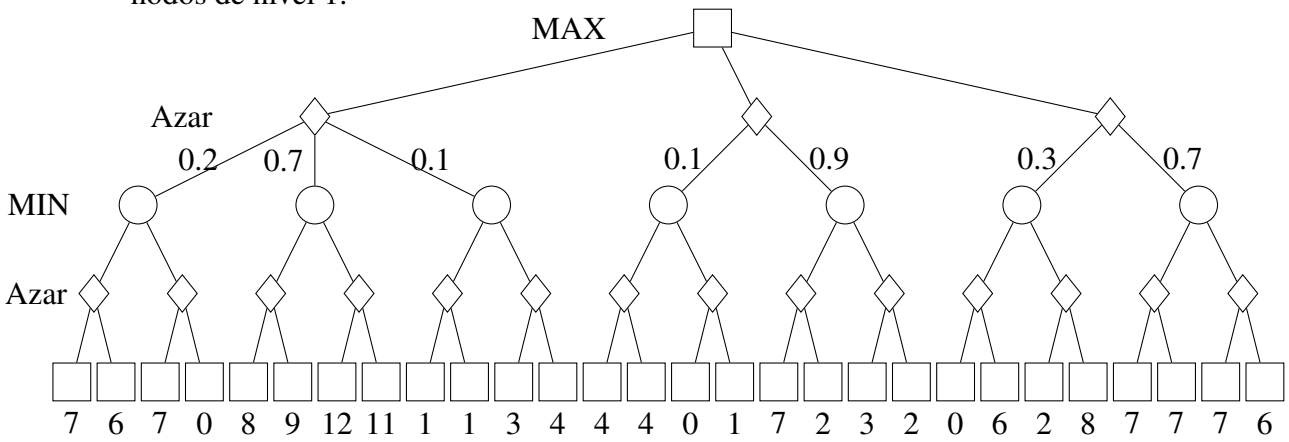
b) Desarrolle el árbol de búsqueda Minimax hasta nivel 4 (dos jugadas del agente y del contrario, respectivamente; empieza el agente). Genere los vecinos de un nodo en orden alfabético, es decir, el primer sucesor del nodo Salida sería A y el segundo B.

c) Defina una función de evaluación adecuada para los nodos hoja, y propague sus valores a través del árbol. Qué jugada haría el agente?

d) Qué partes del árbol del apartado c) no se expandirían si se aplicara la poda?

(Sol. en pág. 27)

Ejercicio 1.17 Considere el siguiente árbol de juego. Evalúe el árbol utilizando el algoritmo ExpectiMinimax. Las probabilidades de los diferentes nodos son 0,5 para cada acción en los nodos de azar del nivel 3 y los que se indican en el árbol para los nodos de nivel 1.



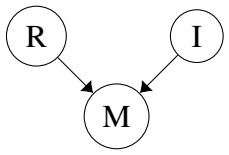
(Sol. en pág. 29)

2. Representación del conocimiento

Ejercicio 2.1 Ha habido un brote de paperas en la universidad. Te sientes bien, pero te preocupa que ya estés infectado y por lo tanto no estés lo suficientemente sano para hacer vida normal. Decides utilizar las redes de Bayes para analizar la probabilidad de que hayas contraído las paperas. Consideras los siguientes factores:

- Crees que tienes inmunidad a las paperas (+i) por haberte vacunado recientemente, pero la vacuna no es completamente eficaz, por lo que podrías no ser inmune (-i).
- Tu compañero de piso no se encontró bien ayer y, aunque no estás seguro todavía, sospechas que podría tener las paperas (+r).

Notación: $I =$ inmunidad, $R =$ estado del compañero y $M =$ paperas. La red de Bayes que describe tus posibilidades de estar enfermo es:



I	P(I)
+i	0.8
-i	0.2

R	P(R)
+r	0.4
-r	0.6

I	R	M	P(M | I,R)
+i	+r	+m	0
+i	+r	-m	1
+i	-r	+m	0
+i	-r	-m	1
-i	+r	+m	0.7
-i	+r	-m	0.3
-i	-r	+m	0.2
-i	-r	-m	0.8

a) Completa la siguiente tabla con la distribución conjunta de I, M y R, es decir, $P(I, M, R)$. Solo se han calculado algunos valores; rellena los demás siguiendo la regla de la cadena y usando las probabilidades condicionales y marginales dadas.

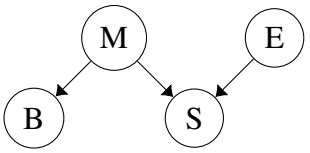
I	R	M	P(I,M,R)
+i	+r	+m	0
+i	+r	-m	
+i	-r	+m	0
+i	-r	-m	
-i	+r	+m	0.056
-i	+r	-m	0.024
-i	-r	+m	0.024
-i	-r	-m	0.096

b) ¿Cuál es la probabilidad marginal P(+m) de que tengas paperas?

(Sol. en pág. 29)

Ejercicio 2.2 El olor a azufre (S) puede ser causado por huevos podridos (E) o como una señal de la fatalidad traída por el Apocalipsis Maya (M). Además, el Apocalipsis Maya también provoca la ebullición de los océanos (B).

La red bayesiana que describe esta situación es:



Se proporcionan las correspondientes tablas de probabilidad condicional:

E	P(E)
+e	0.05
-e	0.95

M	P(M)
+m	0.01
-m	0.99

E	M	S	P(S | E,M)
+e	+m	+s	0.95
+e	+m	-s	0.05
+e	-m	+s	0.9
+e	-m	-s	0.1
-e	+m	+s	0.8
-e	+m	-s	0.2
-e	-m	+s	0.01
-e	-m	-s	0.99

M	B	P(B | M)
+m	+b	0.9
+m	-b	0.1
-m	+b	0.05
-m	-b	0.95

Calcule los siguientes valores:

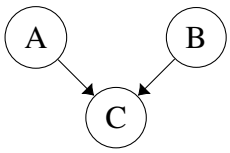
a) $P(-e,-s,-m,-b)$

b) $P(+b)$, la probabilidad de que los océanos lleguen a ebullición.

c) $P(+m \mid +b)$, la probabilidad de que se produzca el Apocalipsis Maya dado que los océanos están hirviendo.

(Sol. en pág. 30)

Ejercicio 2.3 Dada la siguiente red bayesiana:



A	P(A)
+a	1/4
-a	3/4

B	P(B)
+b	3/4
-b	1/4

A	B	C	P(C|A,B)
+a	+b	+c	1
+a	+b	-c	0
+a	-b	+c	0
+a	-b	-c	1
-a	+b	+c	1/2
-a	+b	-c	1/2
-a	-b	+c	0
-a	-b	-c	1

a) Escribe la fórmula de la distribución conjunta que se induce de la red bayesiana.

b) Calcula $P(+c)$.

c) Calcula $P(+a, +b)$.

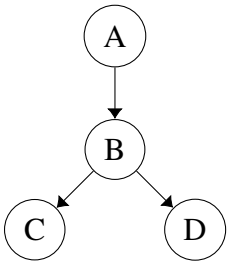
d) Calcula $P(+a, +b \mid +c)$.

(Sol. en pág. 31)

Ejercicio 2.4 Considera la siguiente red bayesiana para las variables A, B, C y D:

A	P(A)
+a	0.8
-a	0.2

B	C	P(C | B)
+b	+c	0.8
+b	-c	0.2
-b	+c	0.5
-b	-c	0.5



A	B	P(B | A)
+a	+b	0.8
+a	-b	0.2
-a	+b	0.5
-a	-b	0.5

B	D	P(D | B)
+b	+d	0.7
+b	-d	0.3
-b	+d	0.1
-b	-d	0.9

Obtener los siguientes valores:

a) P(+a)

b) P(+a, -b, -c, +d)

c) P(+a | -b, -c, +d)

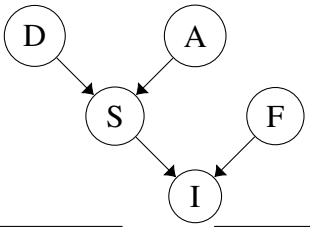
(Sol. en pág. 31)

Ejercicio 2.5 Dada la siguiente red bayesiana lleva a cabo el algoritmo de eliminación de variables con el orden S, A, F para obtener P(F|+i,-d).

D	P(D)
+d	0.1
-d	0.9

A	P(A)
+a	0.4
-a	0.6

F	P(F)
+f	0.4
-f	0.6



D	A	S	P(S|A,D)
+d	+a	+s	0.01
+d	+a	-s	0.99
+d	-a	+s	0.2
-d	-a	-s	0.8
-d	+a	+s	0.25
-d	+a	-s	0.75
-d	-a	+s	0.7
-d	-a	-s	0.3

S	F	I	P(I|S,F)
+s	+f	+i	0.8
+s	+f	-i	0.2
+s	-f	+i	0.7
-s	-f	-i	0.3
-s	+f	+i	0.6
-s	+f	-i	0.4
-s	-f	+i	0.3
-s	-f	-i	0.7

a) Calcula P(F|+i,-d) aplicando el algoritmo de eliminación de variables con el siguiente orden: S, A, F.

b) Calcula P(F|+i,-d) aplicando el algoritmo de enumeración.

c) Calcula P(S|+i) aplicando el algoritmo de enumeración.

(Sol. en pág. 32)

3. Aprendizaje automático

Ejercicio 3.1 Se tiene la siguiente información sobre fraude bancario. Se pide predecir mediante un modelo Naive Bayes si un cliente X con las siguientes características realiza F:Fraude o no basado en los datos de la tabla:

- E: Edad: ≤ 30.
- I Ingresos: Medios.
- T: Nº Tarjetas: 4 o más.
- N: Trans. Nocturnas: si.

ID	E	I	T	N	Fraude
1	≤ 30	Altos	3 o menos	Si	Si
2	≤ 30	Altos	3 o menos	No	Si
3	[31,40]	Altos	3 o menos	Si	No
4	> 40	Medios	3 o menos	Si	No
5	> 40	Bajos	4 o más	Si	No
6	> 40	Bajos	4 o más	No	Si
7	[31,40]	Bajos	4 o más	No	No
8	≤ 30	Medios	3 o menos	Si	Si
9	≤ 30	Bajos	4 o más	Si	No
10	> 40	Medios	4 o más	Si	No
11	≤ 30	Medios	4 o más	No	No
12	[31,40]	Medios	3 o menos	No	No
13	[31,40]	Altos	4 o más	Si	No
14	> 40	Medios	3 o menos	No	Si

(Sol. en pág. 35)

Ejercicio 3.2 Una empresa recoge información sobre el uso de su aplicación por parte de los usuarios para así poder detectar si alguien les intenta suplantar la identidad. La información que recogen es del tipo: tiempo que tarda en escribir la contraseña, tiempo que tarda en realizar una operación, número de veces que se confunde y pulsa en "atrás", tiempo total que pasa en la aplicación, horas de principal actividad, etc. Con todas estas mediciones, van sacando medias y, una vez tienen un buen registro fiable sobre la información del usuario, cada vez que se vuelve a meter comparan las mediciones de esa sesión con las previas. Si hay mucha diferencia, salta una alerta de posible suplantación de identidad y se verifica. La empresa ha modelado esta situación mediante un modelo Naive Bayes en el que la variable respuesta es S:Suplantación (si o no) y las variables explicativas son:

- C: Contraseña: mide la diferencia entre el tiempo medio que el usuario tarda en escribir la contraseña y el tiempo que ha tardado en la sesión actual. Valores: menos, usual, más.
- O: Operación: mide la diferencia entre el tiempo medio que el usuario tarda en

realizar una operación y el tiempo actual. Valores: usual, anómalo.

- A: Atrás: diferencia con la media de pulsaciones en atrás. Valores: ≤ 4, > 4.
- T: Tiempo: diferencia con la media del tiempo que pasa en la aplicación. Valores: usual, anómalo.
- H: Horas: diferencia con las principales horas de actividad. Valores: diurna, usual, nocturna.

En base a esto, la empresa ha proporcionado la siguiente tabla de datos:

ID	C	O	A	T	H	S
1	Usual	Anómalo	≤ 4	Anómalo	Diurna	No
2	Más	Anómalo	≤ 4	Anómalo	Nocturna	Si
3	Usual	Usual	> 4	Usual	Diurna	No
4	Usual	Usual	≤ 4	Usual	Diurna	No
5	Usual	Usual	≤ 4	Usual	Usual	No
6	Más	Usual	> 4	Usual	Usual	No
7	Usual	Usual	≤ 4	Anómalo	Nocturna	Si
8	Menos	Usual	≤ 4	Usual	Nocturna	No
9	Usual	Usual	≤ 4	Usual	Usual	No
10	Menos	Anómalo	> 4	Anómalo	Usual	Si
11	Usual	Usual	> 4	Usual	Usual	No
12	Usual	Usual	≤ 4	Anómalo	Usual	No
13	Usual	Anómalo	≤ 4	Usual	Usual	No
14	Más	Usual	> 4	Usual	Nocturna	Si
15	Menos	Anómalo	≤ 4	Anómalo	Nocturna	Si

¿Cómo clasificaría el modelo a un individuo X con las siguientes características?

- $C =$ Menos
- $O =$ Usual
- $A = \leq 4$
- $T =$ Anómalo
- $H =$ Nocturna

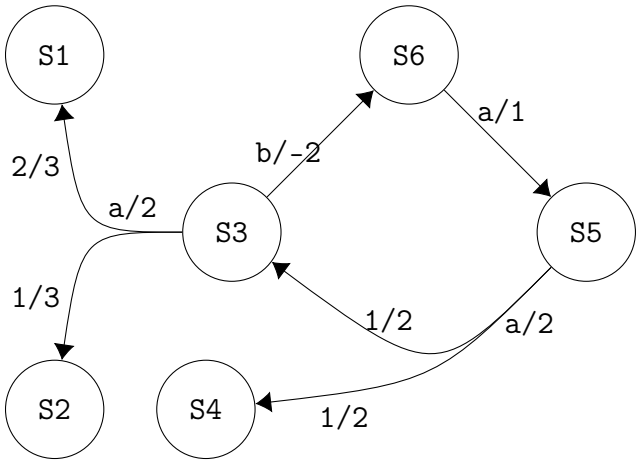
(Sol. en pág. 36)

Ejercicio 3.3 Construya el árbol de decisión de acuerdo al algoritmo ID3 para los siguientes datos donde se donde se concluye dar o no un crédito respecto a un cliente bancario de acuerdo a ciertos atributos (nivel de ingresos, tipo de contrato, etc.).

Cliente	Moroso	Antigüedad	Ingresos	Trabajo Fijo	Conceder
1	si	>5	600–1200	si	no
2	no	<1	600–1200	si	si
3	si	1–5	>1200	si	no
4	no	>5	>1200	no	si
5	no	<1	>1200	si	si
6	si	1–5	600–1200	si	no
7	no	1–5	>1200	si	si
8	no	<1	< 600	si	no
9	no	>5	600–1200	no	no
10	si	1–5	< 600	no	no

(Sol. en pág. 38)

Ejercicio 3.4 Considera el siguiente grafo correspondiente a un problema modelado como Proceso de decisión de Markov. Los nodos representan los grafos y las flechas las acciones (asociados con sus nombres, recompensas y probabilidades de transición).



a) Calcule los valores de la función Q* (función estado-acción-valor) para todas las acciones de este problema. Nota: NO es necesario aplicar un algoritmo para calcular los acciones (aunque también es una opción válida).

(Sol. en pág. 40)

Ejercicio 3.5 Considera el siguiente escenario. Una persona se encuentra delante de una máquina tragaperras y debe decidir si juega o no. Si decide jugar, paga un euro como precio del juego. La máquina tiene tres botones: A, B y COBRAR. El juego consiste en sumar números a un contador que inicialmente está en 0, con el objetivo de conseguir una suma exactamente igual a 3. Para aumentar el contador, el jugador puede presionar los botones A o B: (i) Al presionar el botón A, la máquina siempre

suma 2 al contador. (ii) Al presionar el botón B, la máquina suma aleatoriamente 1 o 2 al contador, cada valor con probabilidad 0.5.

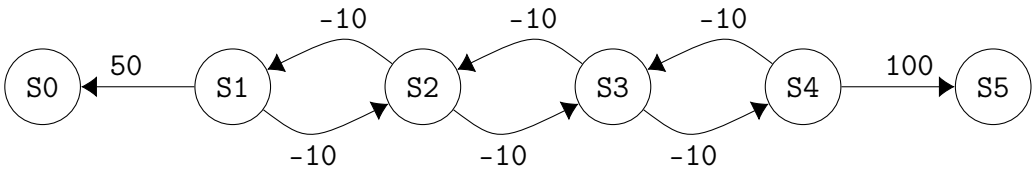
El jugador puede presionar los botones A o B mientras el contador sea menor que 3. Si el contador alcanza o supera 3, los botones A y B dejan de funcionar y el juego termina. Si al terminar el valor del contador es exactamente 3, se activa el botón COBRAR. Al presionarlo, la máquina devuelve 2 euros de ganancia al jugador. Si el contador supera 3, el botón COBRAR no se activa y el juego termina sin ganancias.

a) Modeliza este problema como un Proceso de Decisión de Markov (MDP). Especifica los diferentes estados, incluyendo también el estado inicial en el que el el que el jugador decide si juega o no juega con la máquina), las acciones con sus recompensas y respectivas transiciones. NOTA: Puedes usar un grafo o un conjunto de tablas para especificar el modelo.

b) Suponiendo que la constante de descuento γ tenga el valor 1, determina los valores de la política óptima (π^*) así como los valores de las funciones valor-estado (V*) y valor-estado-acción (Q*) para todos los estados y pares estado- acción y con respecto a la política óptima. NOTA: para determinar estos valores NO es necesario aplicar el algoritmo visto en clase. ¿Es razonable jugar a este juego?

(Sol. en pág. 40)

Ejercicio 3.6 Dada la siguiente figura:



Si hay una flecha entre dos estados Si y Sj significa que se puede ejecutar un acción que hace transitar el agente, de manera determinista, del estado si al estado Sj , recibiendo la correspondiente recompensa. Los estados S0 y S5 son estados terminales.

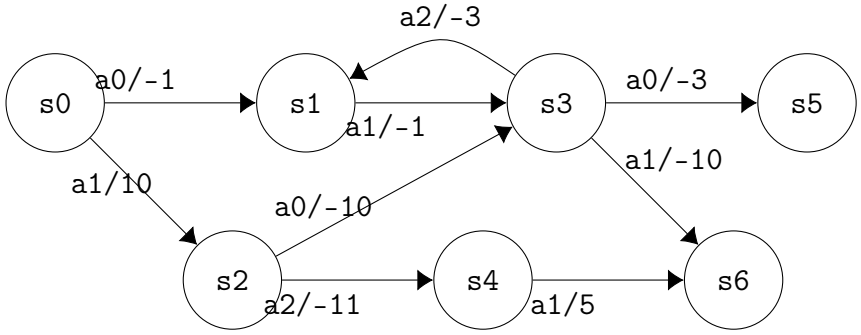
a) Calcule los valores V*(s) para todos los estados, suponiendo que $\gamma = 1$. Calcule también los valores de $Q*(s,a)$ para todas las parejas estado-acción, y la política óptima en este entorno.

b) Calcule los valores V*(s) para todos los estados, suponiendo que $\gamma = 0,8$. Calcule también los valores de $Q*(s,a)$ para todas las parejas estado-acción, y la política óptima en este caso.

c) ¿Qué diferencia hay entre la opción a) y la b)? Argumente su respuesta.

(Sol. en pág. 41)

Ejercicio 3.7 El siguiente grafo representa un MDP determinista. Aplicando el algoritmo de iteración de valores para la resolución de MDPs, presenta en una tabla la evolución de los valores de Q^* y determina los valores finales de V^* y π^*. Realiza el algoritmo hasta que se hayan obtenido los valores finales de Q^*.

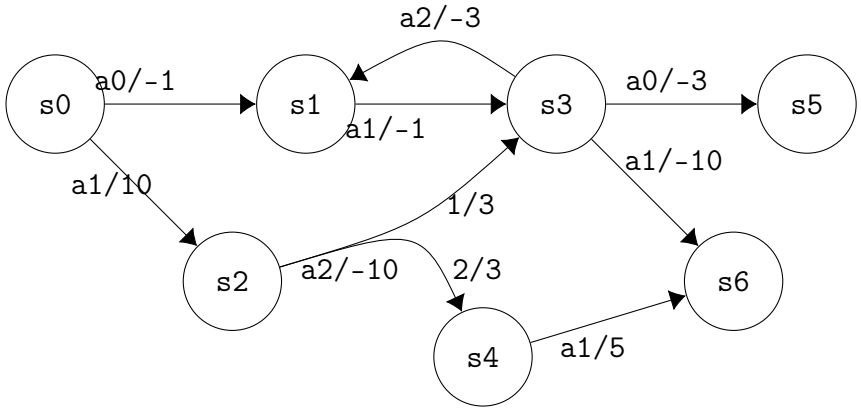


Los parámetros a utilizar son los siguientes:

- Valores de Q^* inicializados a 0.
- Orden de ejecución en cada iteración: estados de s_0 a s_6 y, para cada estado, acciones de a_0 a a_2.
- $\gamma = 1$.

(Sol. en pág. 41)

Ejercicio 3.8 El siguiente grafo representa un MDP NO determinista. Aplicando el algoritmo de iteración de valores para la resolución de MDPs, presenta en una tabla la evolución de los valores de Q^* y determina los valores finales de V^* y π^*. Realiza el algoritmo hasta que se hayan obtenido los valores finales de Q^*. Los parámetros a utilizar son los siguientes:



- Valores de Q^* inicializados a 0.
- Orden de ejecución en cada iteración: estados de s_0 a s_6 y, para cada estado, acciones de a_0 a a_2.
- $\gamma = 1$. Nota: En este grafo existe una transición NO determinista (acción a_2 en s_2).

(Sol. en pág. 42)

Ejercicio 3.9 ¿Cuáles de las siguientes afirmaciones acerca del algoritmo Q-learning son ciertas?

a) Para garantizar la convergencia de los valores Q a los valores Q^* óptimos, el entorno tiene que ser determinista.

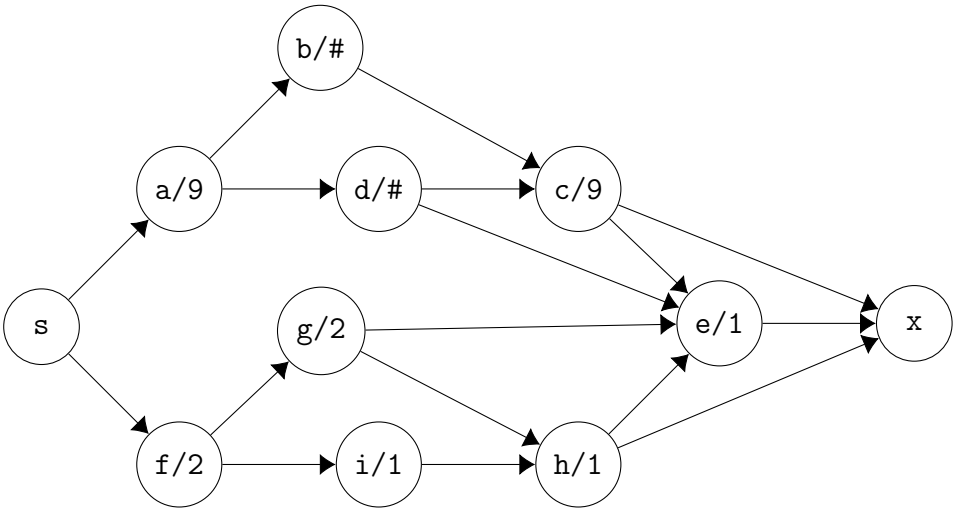
b) Para garantizar la convergencia de los valores Q a los valores Q^* óptimos, el entorno tiene que ser estacionario.

c) Los algoritmos de aprendizaje por refuerzo aprenden la política óptima $\pi^*(s)$ usando un conjunto de parejas $\langle s, \pi^*(s)\rangle$.

d) Los algoritmos de aprendizaje por refuerzo en su versión básica necesitan almacenar un valor $Q(s,a)$ para toda pareja estado-acción $\langle s,a\rangle$.

(Sol. en pág. 42)

Ejercicio 3.10 En el Laberinto de Creta, Teseo debe encontrar la salida antes de ser alcanzado por el Minotauro. El laberinto está representado por el grafo dirigido que se muestra en la figura, donde Teseo parte del estado inicial s y debe alcanzar el estado final x.



Teseo puede desplazarse de un punto de cruce a otro únicamente siguiendo la dirección de las flechas. No puede retroceder ni volver por el camino del que proviene. Cuando Teseo entra en un nuevo punto de cruce, obtiene una recompensa según el símbolo asociado al nodo:

- 9: Teseo obtiene 9 monedas.
- 2: Teseo obtiene 2 monedas.
- 1: Teseo obtiene 1 moneda.
- #: Teseo pierde una vida.

En el estado x Teseo no obtiene recompensa y el juego termina. Para poder comparar monedas y vidas, Teseo considera que cada vida perdida equivale a 10 monedas. Por tanto, perder una vida se modela como una recompensa de -10. Los estados se identifican mediante las letras minúsculas de la figura. Las acciones disponibles en cada estado se denominan a_1 y a_2, de modo que el camino situado más arriba corresponde a la acción a_1 y el otro (si existe) corresponde a la acción a_2. Por ejemplo, en el estado d, tomar el camino hacia c corresponde a ejecutar la acción a_1, mientras que tomar el camino hacia e corresponde a ejecutar la acción a_2.

Se supone que Teseo no conoce el laberinto a priori y desea aprender el camino más lucrativo a lo largo de sucesivas partidas. El camino más lucrativo es aquel que maximiza la recompensa total acumulada.

Aplique el algoritmo de Q-learning a este problema con los siguientes parámetros:

- Los valores de la función $Q(s,a)$ se inicializan a 0.
- El factor de descuento es $\gamma = 1$.
- La tasa de aprendizaje es $\alpha = 1$.
- Teseo sigue una política greedy (avara).
- En caso de empate entre acciones, elige siempre la acción superior (a_1).

a) Ejecute el algoritmo de Q-learning para una primera instancia del juego, indicando la secuencia de estados recorridos y la evolución de los valores de la función Q tras cada transición.

b) Utilizando los valores aprendidos, ejecute el algoritmo para una segunda y una tercera instancia del juego. Indique nuevamente la secuencia de estados y la evolución de la función Q.

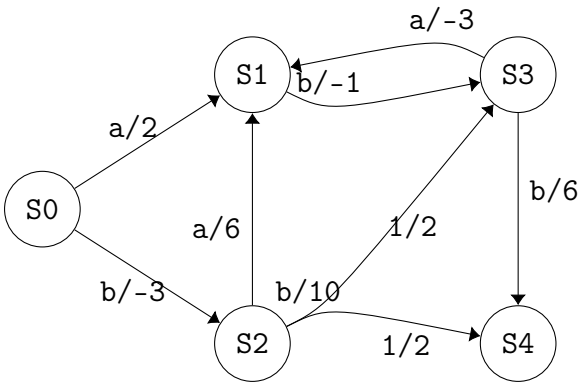
c) Indique si Teseo aprendería el camino óptimo si el proceso se repitiese indefinidamente. Justifique su respuesta, i.e., argumenta por qué encuentra (o no) el camino óptimo.

(Sol. en pág. 42)

Ejercicio 3.11 Aplica el algoritmo de aprendizaje *Q-learning* al problema representado en el siguiente grafo. Presenta la evolución de los valores $Q(s,a)$ en una tabla tras ejecutar tres episodios.

En los episodios 1 y 2, el agente selecciona siempre la acción con mayor valor Q (política voraz), eligiendo la acción b en caso de empate. En el episodio 3, el agente actúa igual que antes, salvo en el estado inicial s_0, donde selecciona la acción b para explorar.

Respecto a la acción b en el estado s_2, se asume que la primera vez que se ejecuta, el agente transita a s_3, la segunda vez a s_4, la tercera de nuevo a s_3, y así sucesivamente, alternando entre s_3 y s_4.



Utiliza los parámetros:

- Los valores de $Q(s,a)$ se inicializan en 0.
- El estado inicial en los tres episodios es s_0.
- El factor de descuento es $\gamma = 1$, y la tasa de aprendizaje es $\alpha = 0{,}1$.

(Sol. en pág. 44)

A. Soluciones

Solución del ejercicio 1.1 Que actúen de forma racional en un entorno.

Solución del ejercicio 1.2 Encontrar una solución al problema de las Torres de Hanoi.

Solución del ejercicio 1.3

- La búsqueda en anchura es óptima y completa siempre y cuando el coste de los operadores sea constante.
- Tanto la complejidad en tiempo como la complejidad en espacio de la búsqueda en anchura se pueden expresar en función del número de nodos expandidos

Solución del ejercicio 1.4

a)
```
E_Inicial ← [0,0]
E_Objetivo ← [X,2]

FUNCION vecinos([P,G])
    lista_vecinos ← []

    % 1. Llenar Tres
    SI P < 3 ENTONCES
        lista_vecinos ← [ <[3, G], 1> | lista_vecinos ]
    % 2. Llenar Cuatro
    SI G < 4 ENTONCES
        lista_vecinos ← [ <[P, 4], 1> | lista_vecinos ]
    % 3. Vaciar Tres
    SI P > 0 ENTONCES
        lista_vecinos ← [ <[0, G], 1> | lista_vecinos ]
    % 4. Vaciar Cuatro
    SI G > 0 ENTONCES
        lista_vecinos ← [ <[P, 0], 1> | lista_vecinos ]
    % 5a. Vaciar Cuatro en Tres
    SI G > 0 Y P + G =< 3 ENTONCES
        lista_vecinos ← [ <[P + G, 0], 1> | lista_vecinos ]
    % 5b. Llenar Tres con Cuatro
    SI G > 0 Y P + G > 3 ENTONCES
        lista_vecinos ← [ <[3, G - (3 - P)], 1> | lista_vecinos ]
    % 6a. Vaciar Tres en Cuatro
    SI P > 0 Y P + G =< 4 ENTONCES
        lista_vecinos ← [ <[0, P + G], 1> | lista_vecinos ]
    % 6b. Llenar Cuatro con Tres
    SI P > 0 Y P + G > 4 ENTONCES
        lista_vecinos ← [ <[P - (4 - G), 4], 1> | lista_vecinos ]

    DEVOLVER lista_vecinos
```

b) El agente no tiene conocimiento a priori.

```
vecinos([0,0]) → [ [3,0],[0,4] ]

vecinos([3,0]) → [ [3,4],[0,0],[0,3] ]

vecinos([1,4]) → [ [0,4],[1,0],[3,2],[3,4] ]
```

c)
```
solucion → [
            [0,0],
            [0,4],   % Llenar Cuatro
            [3,1],   % Llenar Tres con Cuatro (5b)
            [0,1],   % Vaciar Tres
            [1,0],   % Vaciar Cuatro en Tres (5a)
            [1,4],   % Llenar Cuatro
            [3,2]    % Llenar Tres con Cuatro (5b)
           ]
```

Solución del ejercicio 1.7

a) Los costes óptimos reales hasta un estado meta son:

$$h(A) = 10, \quad h(B) = 7, \quad h(C) = 7, \quad h(D) = 3, \quad h(E) = 0, \quad h(F) = 0.$$

Una heurística es:

- optimista si $h^*(n) \leq h(n)$
- consistente si $h^*(n) \leq g(n,n') + h*(n')$ para todo arco

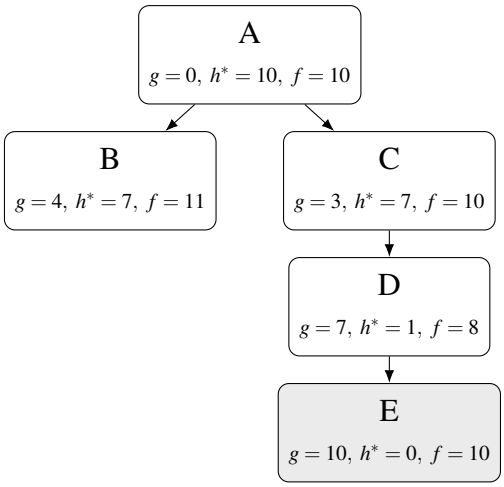
Una heurística optimista pero no consistente es, por ejemplo:

$$h^*(A) = 10, \quad h^*(B) = 7, \quad h^*(C) = 7, \quad h^*(D) = 1, \quad h^*(E) = 0, \quad h^*(F) = 0.$$

Esta heurística es optimista porque nunca sobreestima el coste real hasta una meta, pero no es consistente ya que para el arco $C \to D$ se cumple:

$$h^*(C) = 7 > 4 + h^*(D) = 5.$$

b) El algoritmo A* comienza en el nodo A y genera los nodos B y C. Dado que $f(C) = 3+7 = 10 < f(B) = 4+7 = 11$, se expande C. A continuación se genera D con $f(D) = 7+1 = 8$, que se expande antes que B. Desde D se alcanza el estado meta E con un coste total de 10.

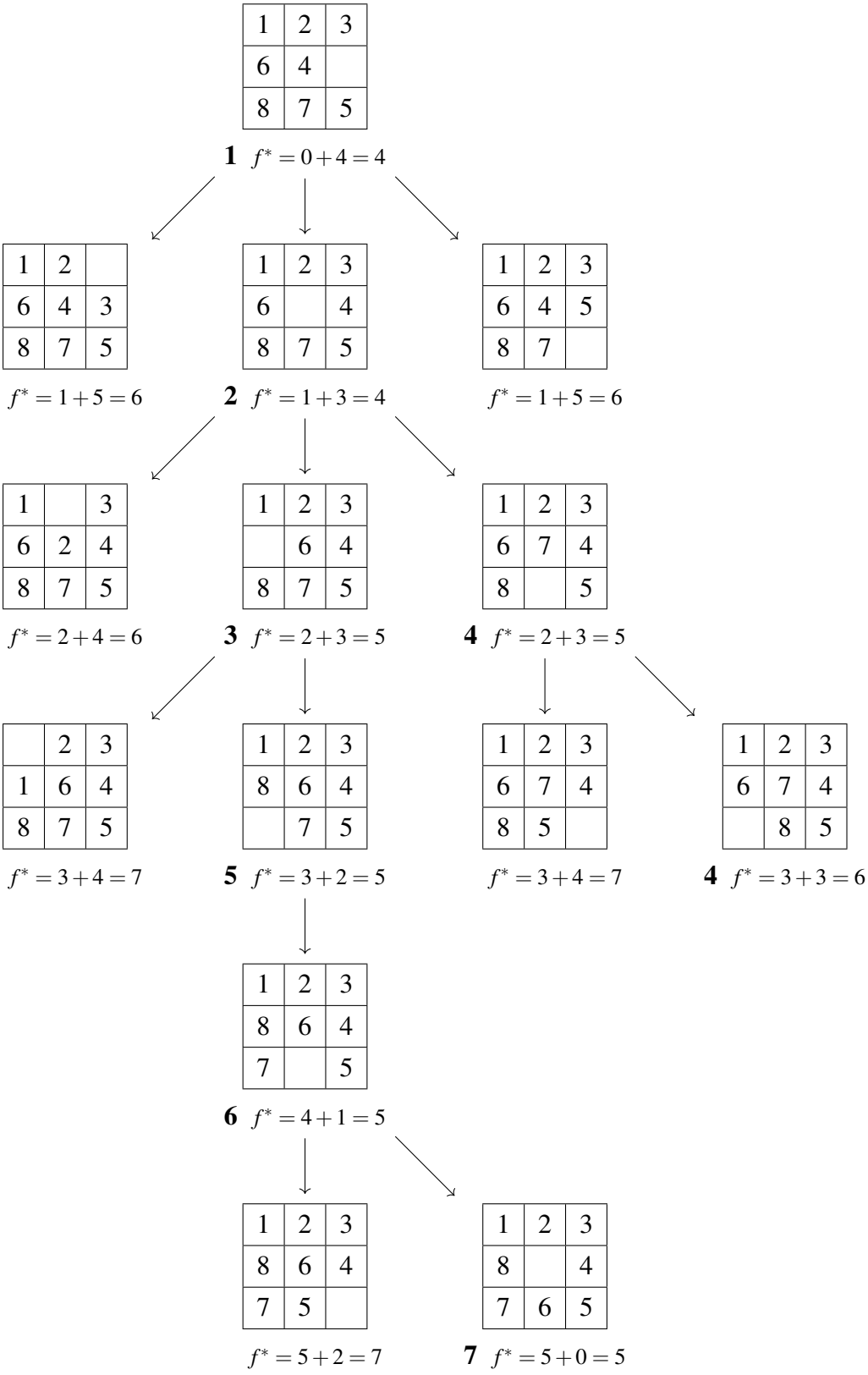


Por tanto, A* encuentra la solución $A \to C \to D \to E$ con coste 10, que es óptima.

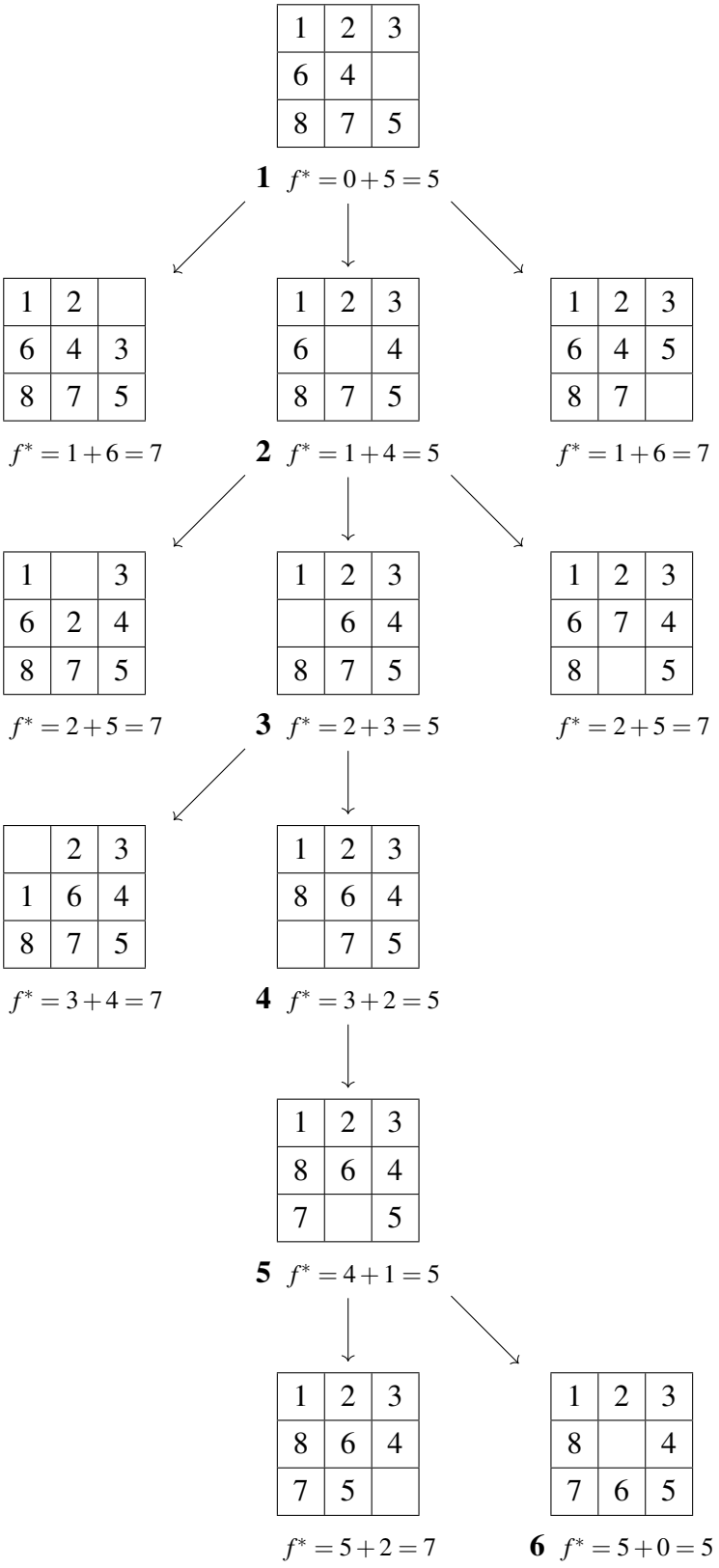
Aunque la heurística no es consistente, en este caso A* resulta ser óptimo al utilizar una heurística optimista.

Solución del ejercicio 1.9

a)


1 2 3
6 4
8 7 5
1 $f^* = 0+4 = 4$
1 2
6 4 3
8 7 5
$f^* = 1+5 = 6$
1 2 3
6 4
8 7 5
2 $f^* = 1+3 = 4$
1 2 3
6 4 5
8 7
$f^* = 1+5 = 6$
1 3
6 2 4
8 7 5
$f^* = 2+4 = 6$
1 2 3
6 4
8 7 5
3 $f^* = 2+3 = 5$
1 2 3
6 7 4
8 5
4 $f^* = 2+3 = 5$
2 3
1 6 4
8 7 5
$f^* = 3+4 = 7$
1 2 3
8 6 4
7 5
5 $f^* = 3+2 = 5$
1 2 3
6 7 4
8 5
$f^* = 3+4 = 7$
1 2 3
6 7 4
8 5
4 $f^* = 3+3 = 6$
1 2 3
8 6 4
7 5
6 $f^* = 4+1 = 5$
1 2 3
8 6 4
7 5
$f^* = 5+2 = 7$
1 2 3
8 4
7 6 5
7 $f^* = 5+0 = 5$

b)


1 2 3
6 4
8 7 5
1 $f^* = 0+5 = 5$
1 2
6 4 3
8 7 5
$f^* = 1+6 = 7$
1 2 3
6 4
8 7 5
2 $f^* = 1+4 = 5$
1 2 3
6 4 5
8 7
$f^* = 1+6 = 7$
1 3
6 2 4
8 7 5
$f^* = 2+5 = 7$
1 2 3
6 4
8 7 5
3 $f^* = 2+3 = 5$
1 2 3
6 7 4
8 5
$f^* = 2+5 = 7$
2 3
1 6 4
8 7 5
$f^* = 3+4 = 7$
1 2 3
8 6 4
7 5
4 $f^* = 3+2 = 5$
1 2 3
8 6 4
7 5
5 $f^* = 4+1 = 5$
1 2 3
8 6 4
7 5
$f^* = 5+2 = 7$
1 2 3
8 4
7 6 5
6 $f^* = 5+0 = 5$

Solución del ejercicio 1.10

- Juego de las Damas.
- Tres-en-Raya.

Solución del ejercicio 1.11

a) Falso.

b) Verdadero.

c) Verdadero.

d) Falso.

e) Verdadero.

Solución del ejercicio 1.12

- ...puede mejorar la velocidad con la que se encuentra una solución (i.e. se elige una jugada más rápidamente que el Minimax simple).
- ...produce siempre la misma solución (i.e. se elige la misma jugada que el Minimax simple).

Solución del ejercicio 1.13

- Los nodos azar representan un evento aleatorio.
- Si un nodo azar tiene k vecinos equiprobables entonces, al evaluar el árbol de juego, se pueden sumar las evaluaciones de los vecinos del nodo azar y dividir dicha suma por k.

$$\text{Valor(nodo azar)} = \sum_{i=1}^{k} p_i \cdot \text{Valor}(v_i)$$

Si todos los vecinos son equiprobables: $p_i = \frac{1}{k}$, entonces:

$$\text{Valor(nodo azar)} = \frac{1}{k} \sum_{i=1}^{k} \text{Valor}(v_i)$$

Solución del ejercicio 1.16

a) Una posible representación es:

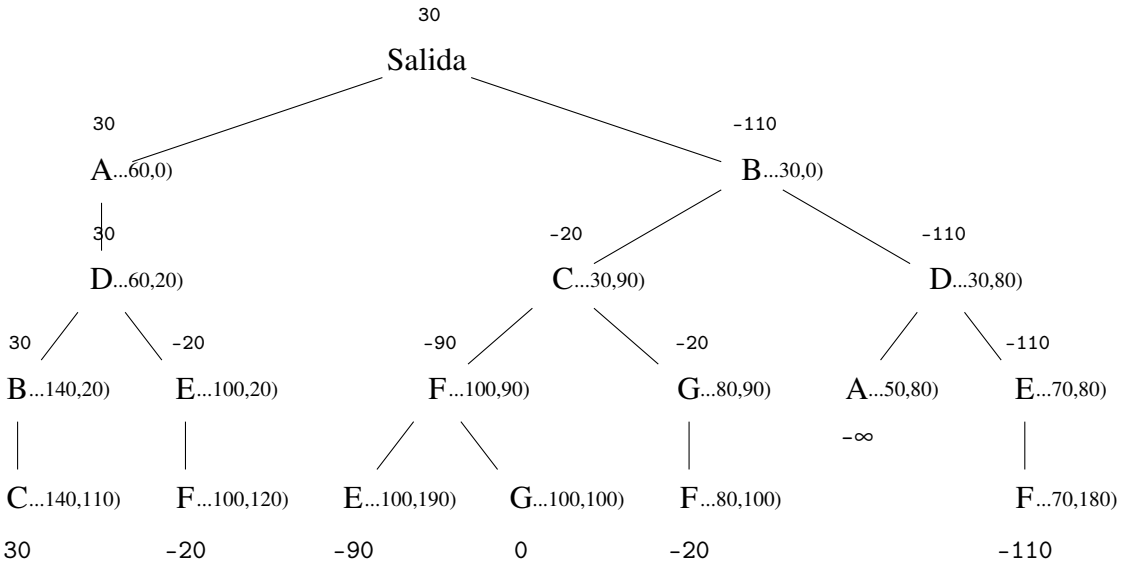
```
Estado = (ciudad_actual, ciudades_visitadas, km_agente, km_contrario)
```

pero se puede simplificar...

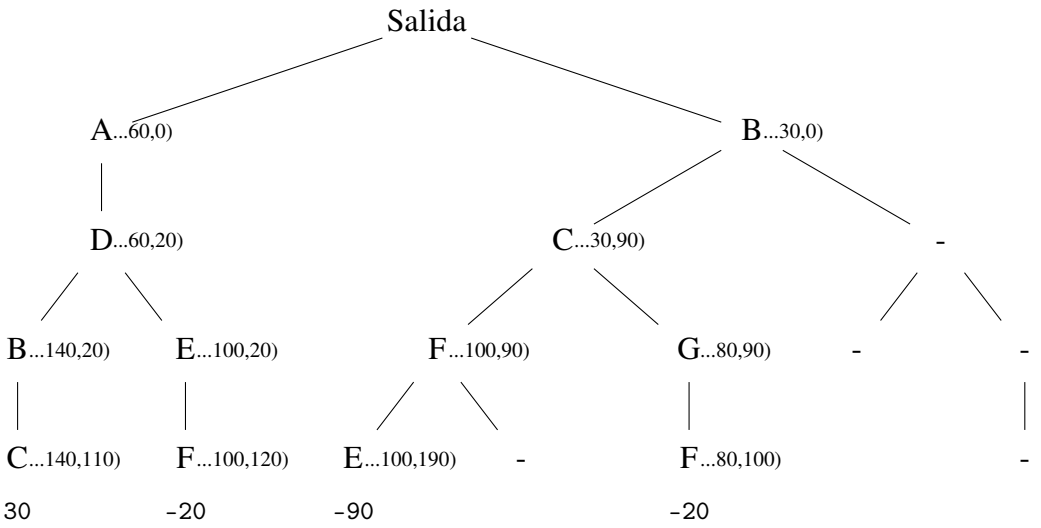
b)

$$e(c_A, cs_v, km_a, km_c) = \begin{cases} +\infty, & \text{si } vecinos(c_A) \subseteq cs_v \text{ y } km_a > km_c \\ -\infty, & \text{si } vecinos(c_A) \subseteq cs_v \text{ y } km_a < km_c \\ km_a - km_c, & \text{en caso contrario} \end{cases}$$

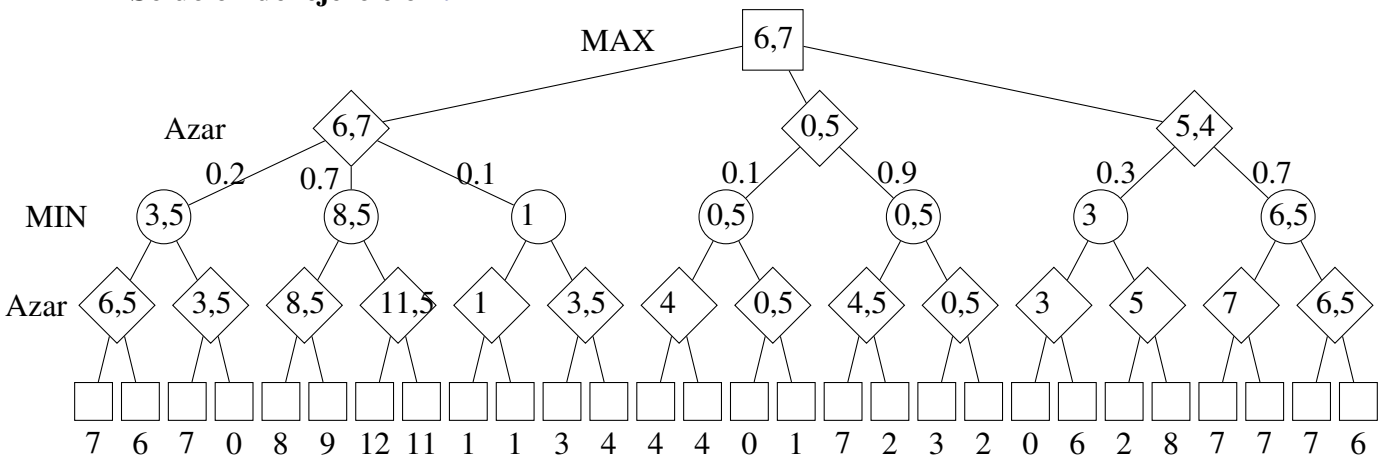
c)



d)



Solución del ejercicio 1.17



Solución del ejercicio 2.1

a) Solo faltan dos valores para completar la distribución conjunta, y como la suma total de probabilidades debe ser 1, calcular uno de ellos permite obtener automáticamente el otro. Para determinar estos valores, consideramos la **topología de la red**, que nos indica las dependencias e independencias condicionales, y aplicamos la **regla de la cadena**:

$$P(+i,+r,-m) = P(-m \mid +i,+r) \cdot P(+i) \cdot P(+r) = 1 \cdot 0{,}8 \cdot 0{,}4 = 0{,}32,$$
$$P(+i,-r,-m) = P(-m \mid +i,-r) \cdot P(+i) \cdot P(-r) = 1 \cdot 0{,}8 \cdot 0{,}6 = 0{,}48.$$

También podríamos haber obtenido $P(+i,-r,-m)$ restando del total 1 la suma de las demás probabilidades:

$$1 - 0 - 0{,}32 - 0 - 0{,}056 - 0{,}024 - 0{,}024 - 0{,}096 = 0{,}48.$$

I	R	M	P(I,M,R)
+i	+r	+m	0
+i	+r	-m	0.32
+i	-r	+m	0
+i	-r	-m	0.48
-i	+r	+m	0.056
-i	+r	-m	0.024
-i	-r	+m	0.024
-i	-r	-m	0.096

b) La probabilidad marginal de tener paperas se calcula sumando sobre todas las combinaciones de I y R:

$$\begin{aligned} P(+m) &= \sum_{i,r} P(+m,i,r) \\ &= P(+m,+i,+r) + P(+m,+i,-r) + P(+m,-i,+r) + P(+m,-i,-r) \\ &= 0 + 0 + 0{,}056 + 0{,}024 \\ &= 0{,}08 \end{aligned}$$

Solución del ejercicio 2.2

a) **Cálculo de** $P(-e,-s,-m,-b)$

Usamos la regla de la cadena, teniendo en cuenta la topología de la red:

$$P(E,S,M,B) = P(E)\cdot P(M)\cdot P(S \mid E,M)\cdot P(B \mid M)$$

Sustituyendo los valores para $(-e,-s,-m,-b)$:

$$\begin{aligned} P(-e,-s,-m,-b) &= P(-e)\cdot P(-m)\cdot P(-s \mid -e,-m)\cdot P(-b \mid -m) \\ &= 0{,}95\cdot 0{,}99\cdot 0{,}99\cdot 0{,}95 \\ &= 0{,}95\times 0{,}99\times 0{,}99\times 0{,}95 \\ &= 0{,}8845 \end{aligned}$$

b) Cálculo de $P(+b)$, la probabilidad de que los océanos lleguen a ebullición

Como B depende solo de M, usamos la suma marginal sobre M:

$$\begin{aligned} P(+b) &= P(+b \mid +m)P(+m) + P(+b \mid -m)P(-m) \\ &= 0{,}9\cdot 0{,}01 + 0{,}05\cdot 0{,}99 \\ &= 0{,}0495 \end{aligned}$$

c) Cálculo de $P(+m \mid +b)$ usando la regla de Bayes. Aplicando la definición de probabilidad condicional:

$$\begin{aligned} P(+m \mid +b) &= \frac{P(+b \mid +m)\,P(+m)}{P(+b)} \\ &= \frac{0{,}9\cdot 0{,}01}{0{,}0585} \\ &= 0{,}154 \end{aligned}$$

Ya conocíamos $P(+b \mid +m) = 0{,}9, \quad P(+m) = 0{,}01, \quad P(+b) \approx 0{,}0585$

Solución del ejercicio 2.3

a) La distribución conjunta inducida por la red bayesiana se escribe usando la regla de la cadena:

$$P(A,B,C) = P(C \mid A,B)\,P(A)\,P(B)$$

b) **Cálculo de** $P(C=T)$

Sumando sobre todas las combinaciones de A y B:

$$\begin{aligned} P(C=T) &= P(C=T \mid A=T,B=T)P(A=T)P(B=T) \\ &\quad + P(C=T \mid A=T,B=F)P(A=T)P(B=F) \\ &\quad + P(C=T \mid A=F,B=T)P(A=F)P(B=T) \\ &\quad + P(C=T \mid A=F,B=F)P(A=F)P(B=F) \\ &= 1\cdot\frac{1}{4}\cdot\frac{3}{4} + 0\cdot\frac{1}{4}\cdot\frac{1}{4} + \frac{1}{2}\cdot\frac{3}{4}\cdot\frac{3}{4} + 0\cdot\frac{3}{4}\cdot\frac{1}{4} \\ &= \frac{3}{16} + 0 + \frac{9}{32} + 0 = \frac{15}{32} \end{aligned}$$

c) **Cálculo de** $P(A=T,B=T)$

Como A y B son independientes:

$$P(A=T,B=T) = P(A=T)\,P(B=T) = \frac{1}{4}\cdot\frac{3}{4} = \frac{3}{16}$$

d) **Cálculo de** $P(A=T,B=T \mid C=T)$

Usando la probabilidad condicional:

$$P(A=T,B=T \mid C=T) = \frac{P(A=T,B=T,C=T)}{P(C=T)}$$

De la distribución conjunta:

$$P(A=T,B=T,C=T) = \frac{3}{16}, \quad P(C=T) = \frac{15}{32}$$

Por lo tanto:

$$P(A=T,B=T \mid C=T) = \frac{3/16}{15/32} = \frac{6}{15} = \frac{2}{5}$$

Solución del ejercicio 2.4

a) P(+a) = 0.8. Sólo hay que fijarse en los valores que nos dan, como A no tiene padres, conocemos la distribución de A.

b) Con la forma de la red para conocer las independencias entre variables y la regla de la cadena: P(+a, -b, -c, +d) = P(+a) · P(-b | +a) · P(-c | -b) · P(+d | -b) = 0.8 · 0.2 · 0.5 · 0.1 = 0.008.

c) Usando la probabilidad condicional y la probabilidad marginal para el denominador:

$$\begin{aligned}
\text{P(+a | -b,-c,+d)} &= \frac{\text{P(+a,-b,-c,+d)}}{\text{P(-b,-c,+d)}} = \\
&= \frac{\text{P(+a)P(-b|+a)P(-c|-b)P(+d|-b)}}{\sum_a \text{P(a,-cb,-c,+d)}} = \\
&= \frac{\text{P(+a)P(-b|+a)P(-c|-b)P(+d|-b)}}{\text{P(+a,-cb,-c,+d)} + \text{P(-a,-cb,-c,+d)}} = \\
&= \frac{0{,}008}{0{,}008 + 0{,}2 \cdot 0{,}5 \cdot 0{,}5 \cdot 0{,}1} = 0{,}615
\end{aligned}$$

Solución del ejercicio 2.5

a) **Eliminación de variables** (orden: S, A, F)

Queremos calcular:

$$P(F \mid +i, -d)$$

La factorización es:

$$P(D)P(A)P(F)P(S|A,D)P(I|S,F)$$

Con la evidencia $D = -d$, $I = +i$ los factores relevantes son:

$$P(A),\ P(F),\ P(S|A,-d),\ P(+i|S,F)$$

1. Eliminamos S

$$f_1(A,F) = \sum_S P(S|A,-d)P(+i|S,F)$$

	$+f$	$-f$
$+a$	0,65	0,40
$-a$	0,74	0,58

2. Eliminamos A

$$f_2(F) = \sum_A P(A) f_1(A,F)$$

$$f_2(+f) = 0{,}704 \qquad f_2(-f) = 0{,}508$$

3. Multiplicamos por $P(F)$

$$P(+f|+i,-d) \propto f_2(+f) \cdot P(+f) = 0{,}704 \cdot 0{,}4 = 0{,}2816$$
$$P(-f|+i,-d) \propto f_2(-f) \cdot P(-f) = 0{,}508 \cdot 0{,}6 = 0{,}3048$$

4. Normalizamos

$$Z = 0{,}2816 + 0{,}3048 = 0{,}5864$$

$$P(+f \mid +i,-d) = \frac{0{,}2816}{0{,}5864} \approx 0{,}48$$
$$P(-f \mid +i,-d) = \frac{0{,}3048}{0{,}5864} \approx 0{,}52$$

b) **Enumeración para** $P(F \mid +i,-d)$

Factorizamos:

$$P(D)P(A)P(F)P(S|A,D)P(I|S,F)$$

Con evidencia $D = -d$, $I = +i$:

$$P(F \mid +i,-d) = \frac{\sum_{A,S} P(-d)P(A)P(F)P(S|A,-d)P(+i|S,F)}{\sum_F \sum_{A,S} P(-d)P(A)P(F)P(S|A,-d)P(+i|S,F)}$$

Tabla de productos $P(A)P(S|A,-d)P(+i|S,F)$:

A	S	F	$P(A)P(S\|A,-d)P(+i\|S,F)$
$+a$	$+s$	$+f$	$0{,}4 \cdot 0{,}25 \cdot 0{,}8 = 0{,}08$
$+a$	$-s$	$+f$	$0{,}4 \cdot 0{,}75 \cdot 0{,}6 = 0{,}18$
$-a$	$+s$	$+f$	$0{,}6 \cdot 0{,}7 \cdot 0{,}8 = 0{,}336$
$-a$	$-s$	$+f$	$0{,}6 \cdot 0{,}3 \cdot 0{,}6 = 0{,}108$
$+a$	$+s$	$-f$	$0{,}4 \cdot 0{,}25 \cdot 0{,}7 = 0{,}07$
$+a$	$-s$	$-f$	$0{,}4 \cdot 0{,}75 \cdot 0{,}3 = 0{,}09$
$-a$	$+s$	$-f$	$0{,}6 \cdot 0{,}7 \cdot 0{,}7 = 0{,}294$
$-a$	$-s$	$-f$	$0{,}6 \cdot 0{,}3 \cdot 0{,}3 = 0{,}054$

Para $F = +f$:

$$\sum_{A,S} = 0{,}08 + 0{,}18 + 0{,}336 + 0{,}108 = 0{,}704$$

Para $F = -f$:

$$\sum_{A,S} = 0{,}07 + 0{,}09 + 0{,}294 + 0{,}054 = 0{,}508$$

Multiplicamos por $P(D = -d) = 0{,}9$ $\begin{cases} +f : 0{,}704 \cdot 0{,}9 = 0{,}6336 \\ -f : 0{,}508 \cdot 0{,}9 = 0{,}4572 \end{cases}$

Normalizamos:

$$P(F = +f \mid +i, -d) = \frac{0{,}6336}{0{,}6336 + 0{,}4572} \approx 0{,}58$$

$$P(F = -f \mid +i, -d) = \frac{0{,}457}{0{,}6336 + 0{,}4572} \approx 0{,}42$$

c) **Enumeración para** $P(S \mid +i)$

Factorizamos:

$$P(D)P(A)P(F)P(S|A,D)P(I|S,F)$$

Con evidencia $I = +i$:

$$P(S \mid +i) = \frac{\sum_{D,A,F} P(D)P(A)P(F)P(S|A,D)P(+i|S,F)}{\sum_S \sum_{D,A,F} P(D)P(A)P(F)P(S|A,D)P(+i|S,F)}$$

Tabla de productos $P(D)P(A)P(F)P(S|A,D)P(+i|S,F)$ para $S = +s$ y $S = -s$:

D	A	F	S	$P(D)P(A)P(F)P(S\|A,D)P(+i\|S,F)$
$+d$	$+a$	$+f$	$+s$	$0{,}1 \cdot 0{,}4 \cdot 0{,}4 \cdot 0{,}01 \cdot 0{,}8 = 0{,}000128$
$-d$	$+a$	$+f$	$+s$	$0{,}9 \cdot 0{,}4 \cdot 0{,}25 \cdot 0{,}4 \cdot 0{,}8 = 0{,}0288$
$+d$	$-a$	$+f$	$+s$	$0{,}1 \cdot 0{,}6 \cdot 0{,}2 \cdot 0{,}4 \cdot 0{,}8 = 0{,}00384$
$+d$	$+a$	$-f$	$+s$	$0{,}1 \cdot 0{,}4 \cdot 0{,}01 \cdot 0{,}6 \cdot 0{,}7 = 0{,}000168$
$-d$	$-a$	$+f$	$+s$	$0{,}9 \cdot 0{,}6 \cdot 0{,}7 \cdot 0{,}4 \cdot 0{,}8 = 0{,}12096$
$-d$	$+a$	$-f$	$+s$	$0{,}9 \cdot 0{,}4 \cdot 0{,}25 \cdot 0{,}6 \cdot 0{,}7 = 0{,}0378$
$+d$	$-a$	$-f$	$+s$	$0{,}1 \cdot 0{,}6 \cdot 0{,}2 \cdot 0{,}6 \cdot 0{,}7 = 0{,}01008$
$-d$	$-a$	$-f$	$+s$	$0{,}9 \cdot 0{,}6 \cdot 0{,}7 \cdot 0{,}6 \cdot 0{,}7 = 0{,}15876$

D	A	F	S	$P(D)P(A)P(F)P(S\|A,D)P(+i\|S,F)$
$+d$	$+a$	$+f$	$-s$	$0{,}1 \cdot 0{,}4 \cdot 0{,}99 \cdot 0{,}4 \cdot 0{,}6 = 0{,}009504$
$-d$	$+a$	$+f$	$-s$	$0{,}9 \cdot 0{,}4 \cdot 0{,}75 \cdot 0{,}4 \cdot 0{,}6 = 0{,}0648$
$+d$	$-a$	$+f$	$-s$	$0{,}1 \cdot 0{,}6 \cdot 0{,}8 \cdot 0{,}4 \cdot 0{,}6 = 0{,}01152$
$+d$	$+a$	$-f$	$-s$	$0{,}1 \cdot 0{,}4 \cdot 0{,}99 \cdot 0{,}6 \cdot 0{,}3 = 0{,}007128$
$-d$	$-a$	$+f$	$-s$	$0{,}9 \cdot 0{,}6 \cdot 0{,}3 \cdot 0{,}4 \cdot 0{,}6 = 0{,}03888$
$-d$	$+a$	$-f$	$-s$	$0{,}9 \cdot 0{,}4 \cdot 0{,}75 \cdot 0{,}6 \cdot 0{,}3 = 0{,}0486$
$+d$	$-a$	$-f$	$-s$	$0{,}1 \cdot 0{,}6 \cdot 0{,}8 \cdot 0{,}6 \cdot 0{,}3 = 0{,}00864$
$-d$	$-a$	$-f$	$-s$	$0{,}9 \cdot 0{,}6 \cdot 0{,}3 \cdot 0{,}6 \cdot 0{,}3 = 0{,}02916$

Sumatorio por S:

$$\sum_{D,A,F} P(D)P(A)P(F)P(+s|A,D)P(+i|+s,F) = 0{,}355496,$$

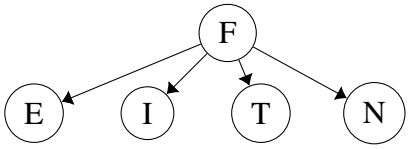
$$\sum_{D,A,F} P(D)P(A)P(F)P(-s|A,D)P(+i|-s,F) = 0{,}218232$$

Normalizamos:

$$P(S=+s \mid +i) = \frac{0{,}355496}{0{,}355496+0{,}218232} \approx 0{,}6196$$

$$P(S=-s \mid +i) = \frac{0{,}218232}{0{,}355496+0{,}218232} \approx 0{,}3804$$

Solución del ejercicio 3.1 Necesitamos calcular P(F=si|X) y P(F=no|X), en función de qué valor salga más alto diremos que el cliente X hace fraude o no. Como nos piden modelarlo mediante un modelo Naive Bayes sabemos que la red bayesiana subyacente es:



Es decir, las variables Edad, Ingresos, Nº Tarjetas y Transacciones nocturnas son condicionalmente independientes entre si dado Fraude. De este modo, podemos expresar la distribución conjunta completa como:

P(F,E,I,T,N) = P(F)P(E|F)P(I|F)P(T|F)P(N|F)

Nosotros estamos interesados en calcular $P(F = \text{si} \mid X)$ y $P(F = \text{no} \mid X)$. Gracias a la probabilidad condicional sabemos que:

$$P(F = \text{si} \mid X) = \frac{P(F = \text{si}, X)}{P(X)} = \alpha P(F = \text{si}, X)$$

$$= \alpha P(F = \text{si}, E \leq 30, I = \text{Medios}, T \geq 4, N = \text{si})$$

Aplicando Naive Bayes:

$$= \alpha P(F = \text{si})P(E \leq 30 \mid F = \text{si})P(I = \text{Medios} \mid F = \text{si})P(T \geq 4 \mid F = \text{si})P(N = \text{si} \mid F = \text{si})$$

Del mismo modo para $P(F = \text{no} \mid X)$:

$$= \alpha P(F = \text{no})P(E \leq 30 \mid F = \text{no})P(I = \text{Medios} \mid F = \text{no})P(T \geq 4 \mid F = \text{no})P(N = \text{si} \mid F = \text{no})$$

Ahora sólo falta calcular dichas probabilidades a partir de la tabla:

$$P(F = \text{si}) = \frac{5}{14} \qquad P(F = \text{no}) = \frac{9}{14}$$

$$P(E \leq 30 \mid F = \text{si}) = \frac{3}{5} \qquad P(E \leq 30 \mid F = \text{no}) = \frac{2}{9}$$

$$P(I = \text{Medios} \mid F = \text{si}) = \frac{2}{5} \qquad P(I = \text{Medios} \mid F = \text{no}) = \frac{4}{9}$$

$$P(T \geq 4 \mid F = \text{si}) = \frac{1}{5} \qquad P(T \geq 4 \mid F = \text{no}) = \frac{6}{9}$$

$$P(N = \text{si} \mid F = \text{si}) = \frac{2}{5} \qquad P(N = \text{si} \mid F = \text{no}) = \frac{6}{9}$$

Realizamos los cálculos:

$$P(F = \text{si} \mid X) = \alpha \left(\frac{5}{14} \cdot \frac{3}{5} \cdot \frac{2}{5} \cdot \frac{1}{5} \cdot \frac{2}{5} \right) = \alpha\, 0{,}0069$$

$$P(F = \text{no} \mid X) = \alpha \left(\frac{9}{14} \cdot \frac{2}{9} \cdot \frac{4}{9} \cdot \frac{6}{9} \cdot \frac{6}{9} \right) = \alpha\, 0{,}028$$

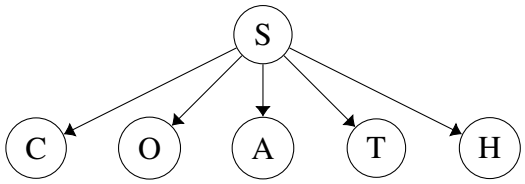
Para obtener la probabilidad final es necesario normalizar:

$$P(F = \text{si} \mid X) = \frac{0{,}0069}{0{,}0069 + 0{,}028} \approx 0{,}20$$

$$P(F = \text{no} \mid X) = \frac{0{,}028}{0{,}0069 + 0{,}028} \approx 0{,}80$$

Por tanto, el modelo predice que el cliente X NO realiza fraude.

Solución del ejercicio 3.2 Se pide calcular P(S=si|X) y P(S=no|X) para saber cuál sería la clasificación del modelo Naive Bayes. La red subyacente es:



La red subyacente asume que las variables C, O, A, T, H son condicionalmente independientes entre sí dado S. Por tanto, la distribución conjunta completa puede expresarse como:

$$P(S, C, O, A, T, H) = P(S)P(C \mid S)P(O \mid S)P(A \mid S)P(T \mid S)P(H \mid S)$$

Debemos calcular:

$$P(\text{S=si} \mid X) = \frac{P(\text{S=si}, X)}{P(X)} = \alpha P(\text{S=si}, X)$$

$$= \alpha P(\text{S=si}, C = \text{menos}, O = \text{usual}, A \leq 4, T = \text{anómalo}, H = \text{nocturna})$$

$$= \alpha P(\text{S=si}) P(C = \text{m} \mid \text{S=si}) P(O = \text{u} \mid \text{S=si}) P(A \leq 4 \mid \text{S=si}) P(T = \text{a} \mid \text{S=si}) P(H = \text{n} \mid \text{S=si})$$

Del mismo modo para $P(\text{S=no} \mid X)$:

$$= \alpha P(\text{S=no}) P(C = \text{m} \mid \text{S=no}) P(O = \text{u} \mid \text{S=no}) P(A \leq 4 \mid \text{S=no}) P(T = \text{a} \mid \text{S=no}) P(H = \text{n} \mid \text{S=no})$$

A partir de la tabla:

$$P(\text{S=si}) = \frac{5}{15} \qquad P(\text{S=no}) = \frac{10}{15}$$

$$P(C = \text{menos} \mid \text{S=si}) = \frac{2}{5} \qquad P(C = \text{menos} \mid \text{S=no}) = \frac{1}{10}$$

$$P(O = \text{usual} \mid \text{S=si}) = \frac{2}{5} \qquad P(O = \text{usual} \mid \text{S=no}) = \frac{8}{10}$$

$$P(A \leq 4 \mid \text{S=si}) = \frac{3}{5} \qquad P(A \leq 4 \mid \text{S=no}) = \frac{7}{10}$$

$$P(T = \text{anómalo} \mid \text{S=si}) = \frac{4}{5} \qquad P(T = \text{anómalo} \mid \text{S=no}) = \frac{2}{10}$$

$$P(H = \text{nocturna} \mid \text{S=si}) = \frac{4}{5} \qquad P(H = \text{nocturna} \mid \text{S=no}) = \frac{1}{10}$$

Multiplicamos:

$$P(\text{S=si} \mid X) = \alpha \left(\frac{5}{15} \cdot \frac{2}{5} \cdot \frac{2}{5} \cdot \frac{3}{5} \cdot \frac{4}{5} \cdot \frac{4}{5} \right) = \alpha\, 0{,}02048$$

$$P(\text{S=no} \mid X) = \alpha \left(\frac{10}{15} \cdot \frac{1}{10} \cdot \frac{8}{10} \cdot \frac{7}{10} \cdot \frac{2}{10} \cdot \frac{1}{10} \right) = \alpha\, 0{,}0007467$$

Normalizamos:

$$P(\text{S=si} \mid X) = \frac{0{,}02048}{0{,}02048 + 0{,}0007467} \approx 0{,}96$$

$$P(\text{S=no} \mid X) = \frac{0{,}0007467}{0{,}02048 + 0{,}0007467} \approx 0{,}04$$

En este caso, al individuo X le están suplantando la identidad con una probabilidad aproximada del 96 %.

Solución del ejercicio 3.3

a) Primero analizamos el nodo raíz. Dado que hay ejemplos pertenecientes a distintas clases, no es un nodo terminal y hay que encontrar el mejor atributo.

- **Entropía Global** Tenemos 10 ejemplos: 4 "si" y 6 "no". La entropía global es:

$$E_{\text{global}} = -\frac{6}{10}\log_2\frac{6}{10} - \frac{4}{10}\log_2\frac{4}{10} \approx 0{,}97$$

- **Ganancia por Atributo**
  Moroso:
  - Moroso = si: 4 casos, todos "no" $E_{\text{si}} = -1\cdot\log_2 1 - 0\cdot\log_2 0 = 0$
  - Moroso = no: 6 casos, 4 "si" y 2 "no" $E_{\text{no}} = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} \approx 0{,}92$

  Entropía media ponderada: $E_{\text{Moroso}} = \frac{4}{10}\cdot 0 + \frac{6}{10}\cdot 0{,}92 \approx 0{,}54$
  Ganancia: $G_{\text{Moroso}} = 0{,}97 - 0{,}54 \approx 0{,}43$

  Antigüedad:
  Se calculan los tres rangos de antigüedad:

$$E_{<1} \approx 0{,}92, \quad E_{1-5} \approx 0{,}81, \quad E_{>5} \approx 0{,}92$$

  Entropía ponderada: $E_{\text{Antigüedad}} = \frac{3}{10}\cdot 0{,}92 + \frac{4}{10}\cdot 0{,}81 + \frac{3}{10}\cdot 0{,}92 \approx 0{,}87$
  Ganancia: $G_{\text{Antigüedad}} = 0{,}97 - 0{,}87 \approx 0{,}10$

  Ingresos:

$$E_{<600} = 0, \quad E_{600-1200} \approx 0{,}81, \quad E_{>1200} \approx 0{,}81$$

$$E_{\text{Ingresos}} = \frac{2}{10}\cdot 0 + \frac{8}{10}\cdot 0{,}81 \approx 0{,}64$$

$$G_{\text{Ingresos}} = 0{,}97 - 0{,}64 \approx 0{,}33$$

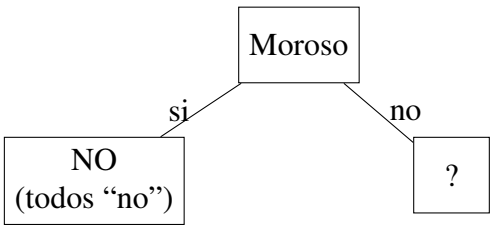
  Trabajo Fijo:

$$E_{\text{si}} \approx 0{,}98, \quad E_{\text{no}} \approx 0{,}92$$

$$E_{\text{TrabajoFijo}} = \frac{7}{10}\cdot 0{,}98 + \frac{3}{10}\cdot 0{,}92 \approx 0{,}97$$

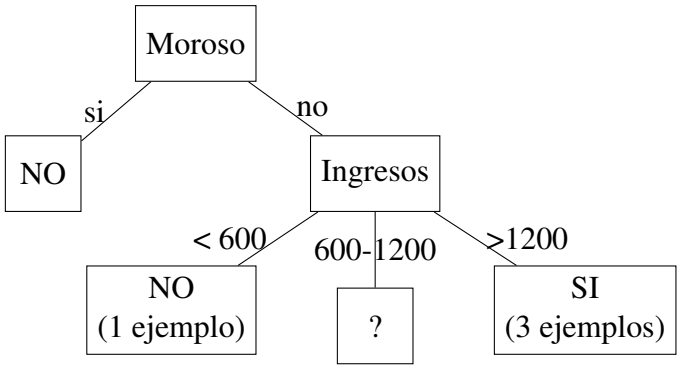
$$G_{\text{TrabajoFijo}} = 0{,}97 - 0{,}97 \approx 0$$

- **Elección del Nodo Raíz**
  Dado este resultado, el mejor atributo (el de mayor ganancia) es el atributo Moroso y por ello se elige como atributo para el nodo raíz del árbol. Respecto al nodo de los ejemplos con Moroso=Si, este nodo es etiquetado con "NO" ya que todos estos ejemplos son del tipo "no". Con ello, el árbol queda en este momento de la siguiente forma:



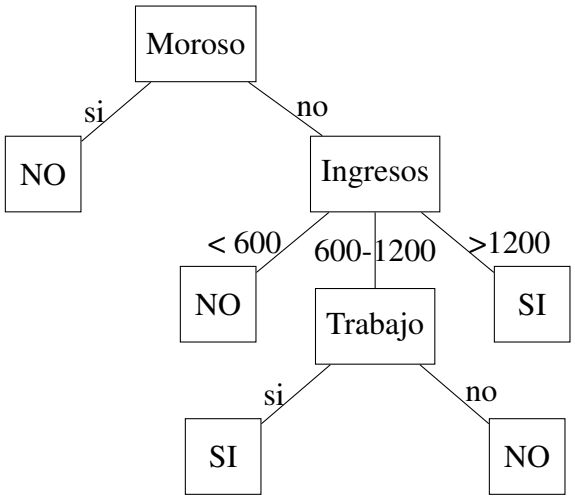
Para el nodo de los ejemplos con Moroso=no, se determina nuevamente el mejor atributo. En este caso sólo se tienen en cuanta los 6 ejemplos que llegan a este nodo, es decir, los que tienen el valor Moros=no.

- **Nivel 2: Nodo Moroso = no**
  Se calculan nuevamente entropías sobre los 6 ejemplos restantes: $E_{\text{global}} \approx 0{,}92$
  Ganancia Antigüedad: $G \approx 0{,}13$
  Ganancia Ingresos: $G \approx 0{,}59$
  Ganancia Trabajo Fijo: $G \approx 0{,}05$
  Se observa que el mejor atributo es Ingresos y se elige este atributo. Dos de las nuevas ramas del árbol se resuelven (terminan) porque todos sus ejemplos son de la misma clase y el árbol queda de la siguiente forma:



  Ahora queda encontrar el mejor atributo para el nodo que todavía no tiene etiqueta. A éste nodo sólo llegan dos ejemplo: el 2 y el 9.

- **Nivel 3: Nodo Ingresos = 600-1200**
  Se tiene un nodo con dos ejemplos: 2 y 9. $E_{\text{global}} = 1$
  Ganancia Trabajo Fijo: $G = 1$
  Se elige **Trabajo Fijo** como atributo y se etiquetan los nodos terminales.



  Finalmente cabe observar que en el último paso, si se hubiera analizado el atributo Antigüedad, se hubiera obtenido la misma ganancia. En este caso, ambos atributos

tienen la misma importancia, por lo que da igual cual de los dos elegir. Sin embargo, con el atributo Antigüedad, se hubiera tenido un nodo para el cual no existen ejemplos. En este caso se debería decidir que clase se devolvería en este nodo.

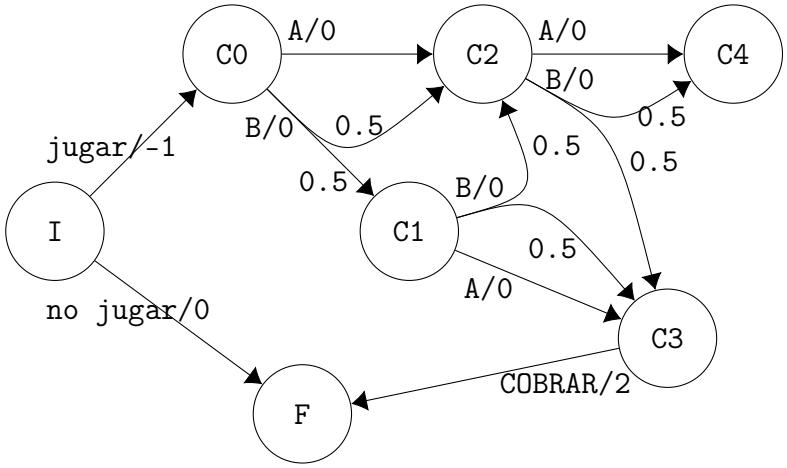
Solución del ejercicio 3.4

a) $Q*(S3,a) = 2$; $Q*(S3,b) = 2$; $Q*(S6,a) = 4$; y $Q*(S5,a) = 3$.

Solución del ejercicio 3.5

a) El siguiente grafo representa el modelo del escenario descrito:

- I: estado inicial en el que el jugador decide si quiere jugar o no.
- F: estado final.
- C_x con $x = 0,1,2,3,4$: estados intermedios que representan el valor actual del contador de la máquina.



b) Valores de V^*:

	I	F	C_0	C_1	C_2	C_3	C_4
V^*	0,5	0	1,5	2	1	2	0

Valores de $Q^*(s,a)$:

$Q^*(s,a)$	I	F	C_0	C_1	C_2	C_3	C_4
A	–	–	1	2	0	–	–
B	–	–	1,5	1,5	1	–	–
COBRAR	–	–	–	–	–	2	–
jugar	0,5	–	–	–	–	–	–
no jugar	0	–	–	–	–	–	–

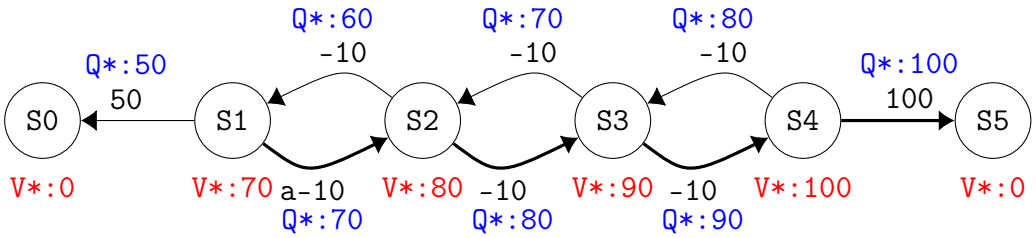
Política óptima π^*:

	I	F	C_0	C_1	C_2	C_3	C_4
π^*	jugar	–	B	A	B	COBRAR	–

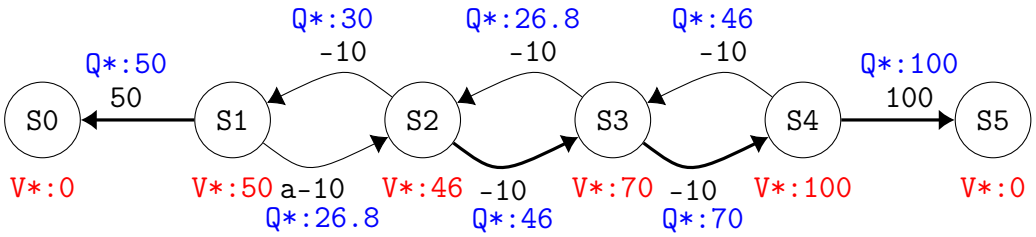
Es razonable jugar a esta máquina, porque la ganancia esperada en cada juego es de 0,5 euros, es decir, en promedio el jugador gana dinero con este juego.

Solución del ejercicio 3.6

a)



b)



c) En la opción b) el valor de γ, 0.8, es menor que el valor de γ en a), el agente da menos importancia a las recompensas futuras. Por tanto, en el estado S1 prefiere ir a la izquierda, en vez de ir hacía el estado S5.

Solución del ejercicio 3.7 En la siguiente tabla se presenta la evolución de los valores de Q*: Los valores finales se encuentran en la cuarta iteración. Los valores de V* y $\pi*$ son los siguientes:

	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_1,a_1)$	$Q(s_2,a_0)$	$Q(s_2,a_2)$	$Q(s_3,a_0)$	$Q(s_3,a_1)$	$Q(s_3,a_2)$	$Q(s_4,a_1)$
Q_1	-1	10	-1	-10	-11	-3	-10	-3	5
Q_2	-2	0	-4	-13	-6	-3	-10	-4	5
$Q_3 = Q^*$	-5	4	-4	-13	-6	-3	-10	-7	5

En la iteración 4 no hay cambios, por lo que $Q_3 = Q^*$.

	s_0	s_1	s_2	s_3	s_4	s_5	s_6
V^*	4	-4	-6	-3	5	0	0

	s_0	s_1	s_2	s_3	s_4
π^*	a_1	a_1	a_2	a_0	a_1

Estados s_5 y s_6 son terminales.

Solución del ejercicio 3.8 En la siguiente tabla se presenta la evolución de los valores de Q*: Los valores finales se encuentran en la cuarta iteración. Los valores de V* y $\pi*$ son los siguientes:

	$Q(s_0,a_0)$	$Q(s_0,a_1)$	$Q(s_1,a_1)$	$Q(s_2,a_2)$	$Q(s_3,a_0)$	$Q(s_3,a_1)$	$Q(s_3,a_2)$	$Q(s_4,a_1)$
Q_1	-1	10	-1	-10	-3	-10	-3	5
Q_2	-2	0	-4	-7.67	-3	-10	-4	5
$Q_3 = Q^*$	-5	2.33	-4	-7.67	-3	-10	-7	5

En la iteración 4 no hay cambios, por lo que $Q_3 = Q^*$.

	s_0	s_1	s_2	s_3	s_4	s_5	s_6
V^*	2.33	-4	-7.67	-3	5	0	0
π^*	a_1	a_1	a_2	a_0	a_1	-	-

Estados s_5 y s_6 son terminales.

Solución del ejercicio 3.9

- Para garantizar la convergencia de los valores Q a los valores Q^* óptimos, el entorno tiene que ser estacionario.
- Los algoritmos de aprendizaje por refuerzo en su versión básica necesitan almacenar un valor $Q(s,a)$ para toda pareja estado-acción $\langle s,a \rangle$.

Solución del ejercicio 3.10

a) Recordemos la regla de actualización de Q-learning (con $\gamma = 1$):

$$Q(s,a) \leftarrow r + \text{máx}_b\, Q(s',b).$$

Los valores iniciales son $Q(s,a) = 0 \quad \forall s,a$.

1. Primer episodio

Paso 1: estado s. Empate: $Q(s,a_1) = Q(s,a_2) = 0$. Se elige a_1: $s \rightarrow a$. Recompensa en a: $+9$.

$$Q(s,a_1) = 9 + \text{máx}(Q(a,a_1), Q(a,a_2)) = 9 + 0 = 9$$

Paso 2: estado a. Empate: $Q(a,a_1) = Q(a,a_2) = 0$. Se elige a_1: $a \to b$. Recompensa en b: -10.

$$Q(a,a_1) = -10 + \text{máx}(Q(b,a_1)) = -10 + 0 = -10$$

Paso 3: estado b. Única acción: $b \to c$. Recompensa en c: $+9$.

$$Q(b,a_1) = 9 + \text{máx}(Q(c,a_1), Q(c,a_2)) = 9 + 0 = 9$$

Paso 4: estado c. Empate: se elige a_1: $c \to x$. Recompensa en x: 0

$$Q(c,a_1) = 0 + \text{máx}(Q(c,_)) = 0 + 0 = 0$$

Trayectoria primera instancia: $s \to a \to b \to c \to x$.

	s	a	b	c	d	e	f	g	h	i
a_1	**9**	**−10**	**9**	**0**	0	0	0	0	0	0
a_2	0	0	−	0	0	−	0	0	0	−

b) 2. Segundo episodio

Paso 1: estado s. Se elige a_1 (9 >0): $s \to a$.

$$Q(s,a_1) = 9 + \text{máx}(-10,0) = 9$$

Paso 2: estado a. Ahora se elige a_2: $a \to d$ porque $Q(a,a_1) = -10, \quad Q(a,a_2) = 0$. Recompensa en d: -10

$$Q(a,a_2) = -10 + \text{máx}(Q(d,a_1), Q(d,a_2)) = -10 + 0 = -10$$

Paso 3: estado d. Empate: se elige a_1: $d \to c$. Recompensa en c: $+9$

$$Q(d,a_1) = 9 + \text{máx}(0,0) = 9 + 0 = 9$$

Paso 4: estado c. Empate: se elige a_1: $c \to x$. Recompensa en x: 0

$$Q(c,a_1) = 0 + \text{máx}(Q(c,_)) = 0 + 0 = 0$$

Trayectoria segunda instancia: $s \to a \to d \to c \to x$

	s	a	b	c	d	e	f	g	h	i
a_1	9	−10	9	0	**9**	0	0	0	0	0
a_2	0	**−10**	−	0	0	−	0	0	0	−

3. Tercer episodio

Paso 1: estado s. Se elige a_1 porque $Q(s,a_1) = 9, \quad Q(s,a_2) = 0$.

$$Q(s,a_1) = 9 + \text{máx}(-10,-10) = 9 - 10 = -1$$

Paso 2: estado a. Empate (-10,-10), se elige a_1.

$$Q(a,a_1) = -10 + \text{máx}(9) = -10 + 9 = -1$$

El resto permanece igual.

Trayectoria tercera instancia: $s \to a \to b \to c \to x$

	s	a	b	c	d	e	f	g	h	i
a_1	$\mathbf{-1}$	$\mathbf{-1}$	9	0	9	0	0	0	0	0
a_2	0	-10	–	0	0	–	0	0	0	–

c) Los caminos óptimos son: $s \to a \to b \to c \to e \to x$ y $s \to a \to d \to c \to e \to x$, con ganancia total: $9 - 10 + 9 + 1 = 9$. Sin embargo, con política *greedy pura* no hay exploración. Una vez que Teseo encuentre un camino razonablemente bueno (como el inferior): $s \to f \to g \to e \to x$ con recompensa: $2 + 2 + 1 = 5$ puede quedarse en él indefinidamente si los valores Q se estabilizan favoreciendo esa rama.

Para encontrar el camino óptimo Teseo debería emplear una política que explore más los posibles caminos como, por ejemplo, una política ε-greedy.

Solución del ejercicio 3.11

Recordemos la regla de actualización de Q-learning (con $\gamma = 1$):

$$Q(s,a) \leftarrow Q(s,a) + \alpha \cdot (r + \text{máx}_b\, Q(s',b) - Q(s,a)).$$

Los valores iniciales son $Q(s,a) = 0 \quad \forall s,a$.

1. Primer episodio

Paso 1: estado s_0. Empate: $Q(s_0,a) = Q(s_0,b) = 0$. Se elige b: $s_0 \to s_2$. Recompensa: -3.

$$Q(s_0,b) = 0 + 0,1 \cdot (-3 + \text{máx}(Q(s_2,a), Q(s_2,b)) - 0) = -0,3$$

Paso 2: estado s_2. Empate: $Q(s_2,a) = Q(s_2,b) = 0$. Se elige b: $s_2 \to s_3$. Recompensa: 10.

$$Q(s_2,b) = 0 + 0,1 \cdot (10 + \text{máx}(Q(s_3,a), Q(s_3,b)) - 0) = 1$$

Paso 3: estado s_3. Empate: $Q(s_3,a) = Q(s_3,b) = 0$. Se elige b: $s_3 \to s_4$. Recompensa: 6.

$$Q(s_3,b) = 0 + 0,1 \cdot (6 + \text{máx}(Q(s_4,a), Q(s_4,b)) - 0) = 0,6 \qquad [1]$$

[1] $\text{máx}(Q(s_4,a), Q(s_4,b))$ es 0 ya que no existen acciones disponibles en s4/

It.	s	$A(s)$	s'	r	$Q(s_0,a)$	$Q(s_0,b)$	$Q(s_1,b)$	$Q(s_2,a)$	$Q(s_2,b)$	$Q(s_3,a)$	$Q(s_3,b)$
	s_0				0	0	0	0	0	0	0
1	s_0	b	s_2	-3	0	-0,3	0	0	0	0	0
2	s_2	b	s_3	10	0	-0,3	0	0	1	0	0
3	s_3	b	s_4	6	0	-0,3	0	0	1	0	0,6
		Valores finales			0	-0,3	0	0	1	0	0,6

2. Segundo episodio

Paso 1: estado s_0. Mayor Q: $Q(s_0,a) = 0 > Q(s_0,b) = -0,3$. Se elige a: $s_0 \to s_1$. Recompensa: 2.

$$Q(s_0,a) = 0 + 0,1 \cdot (2 + \text{máx}(Q(s_1,b)) - 0) = 0,2$$

Paso 2: estado s_1. Única acción b: $s_1 \to s_3$. Recompensa: -1.

$$Q(s_1,b) = 0 + 0,1 \cdot (-1 + \text{máx}(Q(s_3,a), Q(s_3,b)) - 0) = -0,04$$

Paso 3: estado s_3. Mayor Q: $Q(s_3,b) = 0,6$. Se elige b: $s_3 \to s_4$. Recompensa: 6.

$$Q(s_3,b) = 0,6 + 0,1 \cdot (6 + \text{máx}(Q(s_4,a), Q(s_4,b)) - 0,6) = 1,14$$

It.	s	$A(s)$	s'	r	$Q(s_0,a)$	$Q(s_0,b)$	$Q(s_1,b)$	$Q(s_2,a)$	$Q(s_2,b)$	$Q(s_3,a)$	$Q(s_3,b)$
	s_0				0	-0,3	0	0	1	0	0,6
1	s_0	a	s_1	2	0,2	-0,3	0	0	1	0	0,6
2	s_1	b	s_3	-1	0,2	-0,3	-0,04	0	1	0	0,6
3	s_3	b	s_4	6	0,2	-0,3	-0,04	0	1	0	1,14
		Valores finales			0,2	-0,3	-0,04	0	1	0	1,14

3. Tercer episodio

Paso 1: estado s_0. Aunque $Q(s_0,a) = 0,2$ es mayor, el agente explora y elige b: $s_0 \to s_2$. Recompensa: -3.

$$Q(s_0,b) = -0,3 + 0,1 \cdot (-3 + \text{máx}(Q(s_2,a), Q(s_2,b)) - (-0,3)) = -0,47$$

Paso 2: estado s_2. Mayor Q: $Q(s_2,b) = 1$. Según el enunciado, $s_2 \to s_4$. Recompensa: 10.

$$Q(s_2,b) = 1 + 0,1 \cdot (10 + \text{máx}(Q(s_4,a), Q(s_4,b)) - 1) = 1,9$$

It.	s	$A(s)$	s'	r	$Q(s_0,a)$	$Q(s_0,b)$	$Q(s_1,b)$	$Q(s_2,a)$	$Q(s_2,b)$	$Q(s_3,a)$	$Q(s_3,b)$
	s_0				0,2	-0,3	-0,04	0	1	0	1,14
1	s_0	b	s_2	-3	0,2	-0,47	-0,04	0	1	0	1,14
2	s_2	b	s_4	10	0,2	-0,47	-0,04	0	1,9	0	1,14
		Valores finales			0,2	-0,47	-0,04	0	1,9	0	1,14