

Evidencias/Apuntes

ISC-8BM Sistemas Expertos y Aprendizaje Automático

❖ JUAN PABLO MARTINEZ ALVAREZ 161080140
❖ EDUARDO LARA GOMEZ 161080147

ÍNDICE

ÍNDICE	1
CAPÍTULO 1 INTRODUCCIÓN	3
1.1 QUÉ ES IA	3
1.2 Los fundamentos de la inteligencia artificial	5
1.3 Historia de la inteligencia artificial	7
1.4 EL ESTADO DEL ARTE	9
1.5 riesgos y beneficios de la ia	10
Lecture 1: Overview Stanford CS221: AI (Autumn 2019)	13
Lec 01: Introduction to AI	13
BÚSQUEDA HEURÍSTICA	22
BE FIRST SEARCH MÉTODOS DFS Y BFS	22
PRINCIPAL ALGORITMO A*	23
EJEMPLO DE DFS EN PYTHON	24
EJEMPLOS DE LOS MÉTODOS HEURÍSTICOS ANTERIORES	28
¿Por qué razonar probabilísticamente?	33
Representar creencias sobre propuestas	33
¿Qué son las redes bayesianas?	36
Probabilístico	37
Gráfico	37
Nodos	38
Discreto	38
Continuo	38
Enlaces	39
Aprendizaje estructural	39
Selección de características	40
Gráfico acíclico dirigido (DAG)	40
Ciclos dirigidos	40
Notación	40
Probabilidad	41

Probabilidad conjunta	41
La probabilidad condicional	41
Probabilidad marginal	41
Distribuciones	43
Aprendizaje de parámetros	44
Aprender en línea	45
Evidencia	45
Instanciación	46
Probabilidad conjunta de una red bayesiana	47
Ley distributiva	48
Teorema de Bayes	48
¿Son las redes bayesianas bayesianas?	48
Inferencia	49
Inferencia exacta	49
Inferencia aproximada	49
Algoritmos	50
Consultas	50
Análisis	50
Redes dinámicas bayesianas	50
Gráficos de decisión	51
Aprendizaje basado en árboles de decisión	51
Regresión Lineal	53
Máquinas de Vector Soporte (Support Vector Machines, SVMs)	55

CAPÍTULO 1 INTRODUCCIÓN

Algunas de las partes más importantes del contexto general del libro se definen en el capítulo 1 enlistare algunos párrafos que pude traducir con ideas , apuntes y comentarios citados en esta parte del documento

PÁRRAFO 2 PÁGINA 29

El campo de la inteligencia artificial, o IA, se ocupa no sólo de comprender, sino también de construir entidades inteligentes, máquinas que pueden calcular cómo actuar de manera efectiva y segura en una amplia variedad de situaciones novedosas.

PÁRRAFO 3 PÁGINA 29

. El experto en inteligencia artificial Kai-Fu Lee predice que su impacto será "más que nada en la historia de la humanidad".

1.1 QUÉ ES IA

- Algunos consideran que la inteligencia es una propiedad de los procesos de pensamiento y el razonamiento internos, mientras que otros se centran en el comportamiento inteligente,

Definición de aprendizaje automático

El aprendizaje automático es un subcampo de la IA que estudia la capacidad de mejorar el rendimiento en función de la experiencia. Algunos sistemas de inteligencia artificial utilizan métodos de aprendizaje automático para lograr la competencia, pero otros no.

- **El libro menciona 4 enfoques del racionalismo**

Sistemas que piensan como humanos	Sistemas que piensan racionalmente
«El nuevo y excitante esfuerzo de hacer que los computadores piensen... máquinas con mentes, en el más amplio sentido literal». (Haugeland, 1985) «[La automatización de] actividades que vinculamos con procesos de pensamiento humano, actividades como la toma de decisiones, resolución de problemas, aprendizaje...» (Bellman, 1978)	«El estudio de las facultades mentales mediante el uso de modelos computacionales». (Charniak y McDermott, 1985) «El estudio de los cálculos que hacen posible percibir, razonar y actuar». (Winston, 1992)
Sistemas que actúan como humanos	Sistemas que actúan racionalmente
«El arte de desarrollar máquinas con capacidad para realizar funciones que cuando son realizadas por personas requieren de inteligencia». (Kurzweil, 1990) «El estudio de cómo lograr que los computadores realicen tareas que, por el momento, los humanos hacen mejor». (Rich y Knight, 1991)	«La Inteligencia Computacional es el estudio del diseño de agentes inteligentes». (Poole <i>et al.</i> , 1998) «IA... está relacionada con conductas inteligentes en artefactos». (Nilsson, 1998)

1.1.1 Actuar humanamente: el enfoque de la prueba de Turing

La Prueba de Turing, propuesta por Alan Turing (1950), se diseñó para proporcionar una definición operacional y satisfactoria de inteligencia. En vez de proporcionar una lista larga y quizá controvertida de cualidades necesarias para obtener inteligencia artificialmente, él sugirió una prueba basada en la incapacidad de diferenciar entre entidades inteligentes indiscutibles y seres humanos

- **Procesamiento de lenguaje natural** que le permita comunicarse satisfactoriamente en inglés.
- **Representación del conocimiento** para almacenar lo que se conoce o siente.
- **Razonamiento automático** para utilizar la información almacenada para responder a preguntas y extraer nuevas conclusiones.
- **Aprendizaje automático** para adaptarse a nuevas circunstancias y para detectar y extrapolar patrones.

1.1.2 Pensar como humano el enfoque del modelo cognitivo

La auténtica ciencia cognitiva se fundamenta necesariamente en la investigación experimental en humanos y animales, y en esta obra se asume que el lector sólo tiene acceso a un computador para experimentar.

1.1.3 Pensamiento racional: el enfoque de las «leyes del pensamiento»

El filósofo griego Aristóteles fue uno de los primeros en intentar codificar la «manera correcta de pensar», es decir, un proceso de razonamiento irrefutable. Sus silogismos son esquemas de estructuras de argumentación mediante las que siempre se llega a conclusiones correctas si se parte de premisas correctas

1.1.4 Actuar de forma racional: el enfoque del agente racional

- **DIFERENCIAS PRINCIPALES ENTRE AGENTES**

Un agente es algo que razona (agente viene del latín agere, hacer). Pero de los agentes informáticos se espera que tengan otros atributos que los distingan de los «programas» convencionales, como que estén dotados de controles autónomos, que perciban su entorno, que persistan durante un período de tiempo prolongado, que se adapten a los cambios, y que sean capaces de alcanzar objetivos diferentes. Un agente racional es aquel que actúa con la intención de alcanzar el mejor resultado o, cuando hay incertidumbre, el mejor resultado esperado.

1.2 Los fundamentos de la inteligencia artificial

Aquí los años más importantes

Filosofía (desde el año 428 a.C. hasta el presente) • ¿Se pueden utilizar reglas formales para extraer conclusiones válidas? • ¿Cómo se genera la inteligencia mental a partir de un cerebro físico? • ¿De dónde viene el conocimiento? • ¿Cómo se pasa del conocimiento a la acción?

- **Aristóteles (384-322 a.C.) fue el primero en formular un conjunto preciso de leyes**

que gobernaban la parte racional de la inteligencia

- **Ramón Lull (d. 1315)** tuvo la idea de que el razonamiento útil se podría obtener por medios artificiales. Sus «ideas» aparecen representadas en la portada de este manuscrito.
- **Thomas Hobbes (1588- 1679)** propuso que el razonamiento era como la computación numérica, de forma que «nosotros sumamos y restamos silenciosamente en nuestros pensamientos». La automatización de la computación en sí misma estaba en marcha; alrededor de 1500,
- **Leonardo da Vinci (1452-1519)** diseñó, aunque no construyó, una calculadora mecánica; construcciones recientes han mostrado que su diseño era funcional.
- **Wilhelm Schickard (1592-1635)**, aunque la Pascalina, construida en 1642 por Blaise Pascal (1623-1662), sea más famosa. Pascal escribió que «la máquina aritmética produce efectos que parecen más similares a los pensamientos que a las acciones animales». **Gottfried Wilhelm Leibniz (1646-1716)** construyó un dispositivo mecánico con el objetivo de llevar a cabo operaciones sobre conceptos en lugar de sobre números, pero su campo de acción era muy limitado

Matemáticas (aproximadamente desde el año 800 al presente)

MUY IMPORTANTE

El concepto de lógica formal se remonta a los filósofos de la antigua Grecia (véase el Capítulo 7), pero su desarrollo matemático comenzó realmente con el trabajo de George Boole (1815-1864) que definió la lógica proposicional o Booleana (Boole, 1847). En 1879, Gottlob Frege (1848-1925) extendió la lógica de Boole para incluir objetos y relaciones, y creó la lógica de primer orden que se utiliza hoy como el sistema más básico de representación de conocimiento. Alfred Tarski (1902-1983) introdujo una teoría de referencia que enseña cómo relacionar objetos de una lógica con objetos del mundo real

Se piensa que el primer algoritmo no trivial es el algoritmo Euclídeo para el cálculo del máximo común divisor.

Alan Turing (1912-1954) a tratar de caracterizar exactamente aquellas funciones que sí eran susceptibles de ser caracterizadas.

ECONOMÍA Y NEUROCIENCIA

La Neurociencia es el estudio del sistema neurológico, y en especial del cerebro. La forma exacta en la que en un cerebro se genera el pensamiento es uno de los grandes misterios de la ciencia

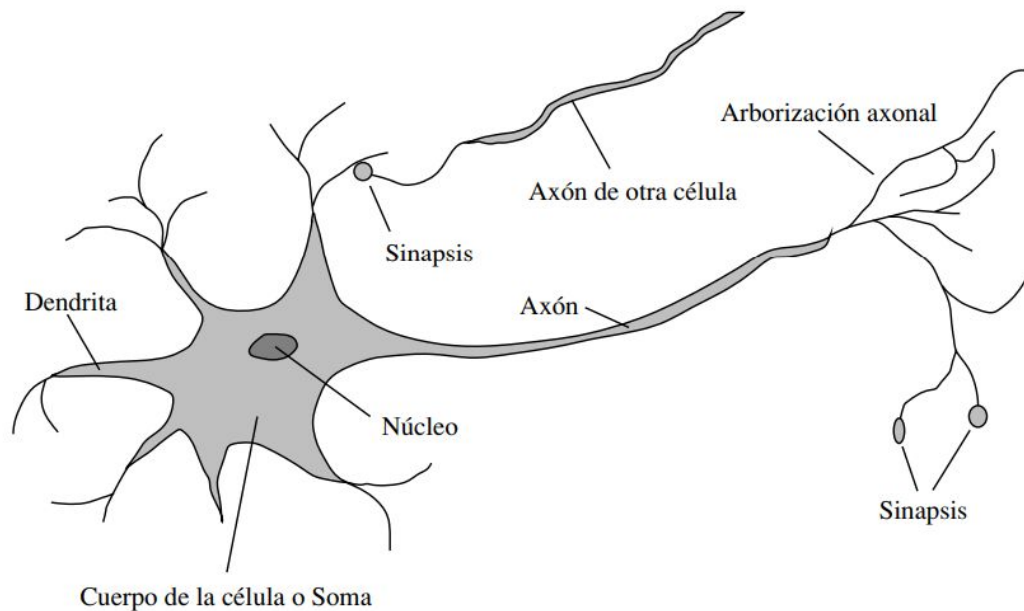


DIAGRAMA PARTES DE UNA CÉLULA NERVIOSA O NEURONA

TEMAS A TRATAR PREGUNTAR LA RELACIÓN DE LA ECONOMIA PSICOLOGIA LINGUISTICA SUPUESTAMENTE EL LIBRO MARCA QUE SON LAS BASES DE LA INT ARTIFICIAL Y LA CONSTRUCCIÓN DE UNA RED NEURONAL ARTIFICIAL

1.3 Historia de la inteligencia artificial

Génesis de la inteligencia artificial (1943-1955)

Warren McCulloch y Walter Pitts (1943) han sido reconocidos como los autores del primer trabajo de IA. Partieron de tres fuentes: conocimientos sobre la fisiología básica y funcionamiento de las neuronas en el cerebro, el análisis formal de la lógica proposicional de Russell y Whitehead y la teoría de la computación de Turing. Propusieron un modelo constituido por neuronas artificiales, en el que cada una de ellas se caracterizaba por estar «activada» o «desactivada»; la «activación» se daba como respuesta a la estimulación producida por una cantidad suficiente de neuronas vecinas.

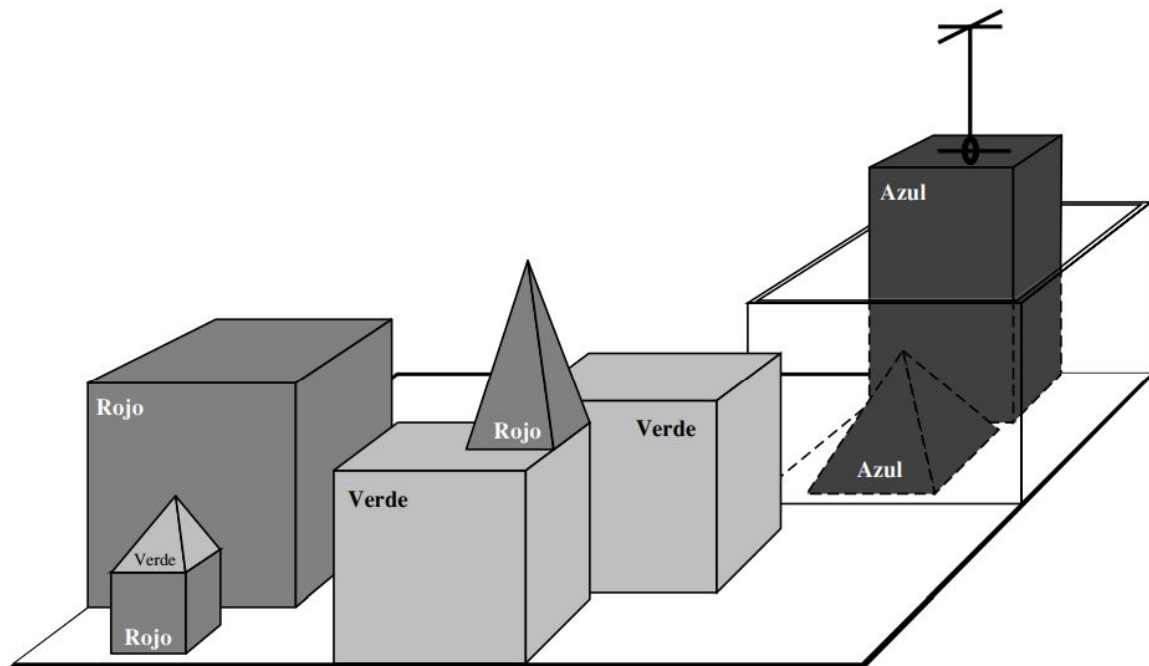
Nacimiento de la inteligencia artificial (1956)

Princeton acogió a otras de las figuras señeras de la IA, John McCarthy. Posteriormente a su graduación, McCarthy se trasladó al Dartmouth College, que se erigiría en el lugar del nacimiento oficial de este campo. McCarthy convenció a Minsky, Claude Shannon y Nathaniel Rochester para que le ayudarían a aumentar el interés de los investigadores americanos en la teoría de autómatas, las redes neuronales y el estudio de la inteligencia. Organizaron un taller con una duración de dos meses en Dartmouth en el verano de 1956. Hubo diez asistentes en total, entre los que se incluían Trenchard More de Princeton, Arthur Samuel de IBM, y Ray Solomonoff y Oliver Selfridge del MIT.

La primera respuesta es que la IA desde el primer momento abarcó la idea de duplicar facultades humanas como la creatividad, la auto-mejora y el uso del lenguaje Ninguno de los otros campos tenían en cuenta esos temas. La segunda respuesta está relacionada con la metodología.

IMPORTANTE DIAGRAMA PRIMERO DE REDES NEURONALES

El trabajo realizado por McCulloch y Pitts con redes neuronales hizo florecer esta área. El trabajo de Winograd y Cowan (1963) mostró cómo un gran número de elementos podría representar un concepto individual de forma colectiva, lo cual llevaba consigo un aumento proporcional en robustez y paralelismo. Los métodos de aprendizaje de Hebb se reforzaron con las aportaciones de Bernie Widrow (Widrow y Hoff, 1960; Widrow, 1962), quien llamó adalines a sus redes, y por Frank Rosenblatt (1962) con sus perceptrones



Sistemas basados en el conocimiento: ¿clave del poder? (1969-1979)

El programa DENDRAL (Buchanan et al. 1969) constituye uno de los primeros ejemplos de este enfoque. Fue diseñado en Stanford, donde Ed Feigenbaum (discípulo de Herbert Simon), Bruce Buchanan (filósofo convertido en informático) y Joshua Lederberg (genetista ganador del Premio Nobel) colaboraron en la solución del problema de inferir una estructura molecular a partir de la información proporcionada por un espectrómetro de masas.

La IA se convierte en una industria (desde 1980 hasta el presente)

El primer sistema experto comercial que tuvo éxito, R1, inició su actividad en Digital Equipment Corporation (McDermott, 1982).

Hasta la fecha muchos dispositivos ocupan la IA

IA se convierte en una ciencia (desde 1987 hasta el presente)

En los primeros años de la IA parecía perfectamente posible que las nuevas formas de la computación simbólica, por ejemplo, los marcos y las redes semánticas, hicieran que la mayor parte de la teoría clásica pasará a ser obsoleta. Esto llevó a la IA a una especie de aislamiento, que la separó del resto de las ciencias de la computación. En la actualidad se está abandonando este aislamiento. Existe la creencia de que el aprendizaje automático no se debe separar de la teoría de la información, que el razonamiento incierto no se debe separar de los modelos estocásticos, de que la búsqueda no se debe aislar de la optimización clásica y el control, y de que el razonamiento automático no se debe separar de los métodos formales y del análisis estático.

1.4 EL ESTADO DEL ARTE

AQUÍ RESUMIMOS LOS IMPORTANTES AVANCES Y CAPACIDADES DE LA IA EN LA HISTORIA

1. **Planificación autónoma:** a un centenar de millones de millas de la Tierra, el programa de la NASA Agente Remoto se convirtió en el primer programa de planificación autónoma a bordo que controlaba la planificación de las operaciones de una nave espacial desde abordó (Jonsson et al., 2000).

2. **Juegos:** Deep Blue de IBM fue el primer sistema que derrotó a un campeón mundial en una partida de ajedrez cuando superó a Garry Kasparov por un resultado de 3.5 a 2.5 en una partida de exhibición (Goodman y Keene, 1997). Kasparov dijo que había percibido un «nuevo tipo de inteligencia» al otro lado del tablero.
3. **Control autónomo:** el sistema de visión por computador ALVINN fue entrenado para dirigir un coche de forma que siguiese una línea. Se instaló en una furgoneta controlada por computador en el NAVLAB de UCM y se utilizó para dirigir al vehículo por Estados Unidos. Durante 2.850 millas controló la dirección del vehículo en el 98 por ciento del trayecto
4. **Diagnosis:** los programas de diagnóstico médico basados en el análisis probabilista han llegado a alcanzar niveles similares a los de médicos expertos en algunas áreas de la medicina. Heckerman (1991) describe un caso en el que un destacado experto en la patología de los nodos linfáticos se mofó del diagnóstico generado por un programa en un caso especialmente difícil.
5. **Planificación logística:** durante la crisis del Golfo Pérsico de 1991, las fuerzas de Estados Unidos desarrollaron la herramienta Dynamic Analysis and Replanning Tool **32 INTELIGENCIA ARTIFICIAL. UN ENFOQUE MODERNO (DART)** (Cross y Walker, 1994), para automatizar la planificación y organización logística del transporte. Lo que incluía hasta 50.000 vehículos, carga y personal a la vez, teniendo en cuenta puntos de partida, destinos, rutas y la resolución de conflictos entre otros parámetros
6. **Robótica:** muchos cirujanos utilizan hoy en día asistentes robot en operaciones de microcirugía. HipNav (DiGioia et al., 1996) es un sistema que utiliza técnicas de visión por computador para crear un modelo tridimensional

1.5 riesgos y beneficios de la ia

Mas bien seria como un resumen a continuación añadiremos unos fragmentos importantes basados en los riesgos de la ia el libro menciona algunos de los siguientes

- **Automatiza los procesos**
La Inteligencia artificial permite que robots desarrollen tareas repetitivas, rutinarias y de optimización de procesos de una manera automática y sin intervención humana.
- **Potencia las tareas creativas**
La IA libera a las personas de tareas rutinarias y repetitivas y permite que estas puedan destinar más tiempo a desarrollar funciones creativas.
- **Aporta precisión**
La aplicación de la IA es capaz de aportar una precisión mayor que el ser humano,

por ejemplo en entornos industriales, las máquinas pueden llegar a tomar decisiones que antes sin la IA se tomaban de manera manual o monitorizada.

- **Reduce el error humano**
La IA reduce los fallos provocados por las limitaciones del ser humano. En algunas cadenas de producción la IA se utiliza para detectar mediante sensores de infrarrojos, pequeñas fisuras o defectos en piezas que son indetectables por el ojo humano.
- **Reduce los tiempos empleados en análisis de datos**
Permite que el análisis y la explotación de los datos derivados de producción se puedan llegar a efectuar en tiempo real.
- **Mantenimiento predictivo**
Permite realizar un mantenimiento del equipamiento industrial basado en los tiempos y condiciones de funcionamiento de los mismos, permitiendo incrementar su rendimiento y ciclo de vida.
- **Mejora en la toma de decisiones tanto a nivel de producción como de negocio**
Al disponer de mayor información de una manera estructurada, permite a cada uno de los responsables tomar decisiones de una manera más rápida y eficiente.
- **Control y optimización de procesos productivos y líneas de producción**
A través de la IA se consiguen procesos más eficientes, libres de errores, obteniendo mayor control sobre las líneas de producción en la empresa.
- **Aumento de la productividad y calidad en la producción**
La IA no sólo incrementa la productividad a nivel de maquinaria, sino que también hace que incremente la productividad de los trabajadores y la calidad del trabajo que realizan. El poder gozar de mayor información, les permite tener una visión más focalizada de su trabajo y tomar mejores decisiones.

Desventajas de la inteligencia artificial (AI), riesgos y barreras

Disponibilidad de datos

A menudo, los datos se presentan de manera aislada en las empresas o son inconsistentes y de baja calidad, con lo que presenta un desafío importante para las empresas que buscan crear valor a partir de la IA a escala. Para poder superar esta barrera, será de vital importancia trazar una estrategia clara desde el principio para poder extraer los datos de IA de una manera organizada y consistente.

Falta de profesionales cualificados

Otro obstáculo que se suele dar a nivel empresarial para la adopción de IA es la escasez de perfiles con habilidades y experiencia en este tipo de implementaciones. Es crucial en estos casos contar con profesionales que ya hayan trabajado en proyectos de la misma envergadura.

Cada uno tiene una visión distinta de lo que es la IA. Es importante responder a las dos preguntas siguientes: ¿Está interesado en el razonamiento y el comportamiento? ¿Desea modelar seres humanos o trabajar a partir de un ideal estándar?

En este libro se adopta el criterio de que la inteligencia tiene que ver principalmente con las acciones racionales. Desde un punto de vista ideal, un agente inteligente es aquel que emprende la mejor acción posible ante una situación dada. Se estudiará el problema de la construcción de agentes que sean inteligentes en este sentido.

- Los filósofos (desde el año 400 a.C.) facilitaron el poder imaginar la IA, al concebir la idea de que la mente es de alguna manera como una máquina que funciona a partir del conocimiento codificado en un lenguaje interno, y al considerar que el pensamiento servía para seleccionar la acción a llevar a cabo.
- Las matemáticas proporcionaron las herramientas para manipular tanto las aseveraciones de certeza lógicas, como las inciertas de tipo probabilista.

Lecture 1: Overview | Stanford CS221: AI (Autumn 2019)

El nacimiento de la IA como concepto empezó en el año 1956 y fue avanzando poco a poco durante sus primeros años, anteriormente ya se usaban redes neuronales en 1943 como por ejemplo para identificar escritura a mano y los programas propiamente referenciados como IA tenían importantes limitaciones tecnológicas propias de la época. Cabe destacar que los sistemas expertos nacieron en 1969 No fue sino hasta nuestra década que la IA tuvo un “BOOM” en cuanto a su explotación y exploración. Lo cual es evidente gracias al sin fin de menciones que tiene la inteligencia artificial en la cultura popular e innumerables aplicaciones que se han implementado en diversas áreas.

Cabe recalcar que la IA está sumamente relacionada con la filosofía y lógica humana un ejemplo muy claro lo tenemos con uno de las categorías dentro de la inteligencia artificial, me refiero a los **agentes de IA** estos resaltan por querer asemejar el pensamiento humano de la manera más cercana posible simulando procesos del pensamiento como la percepción e identificación del entorno, comunicación, razonamiento, aprendizaje, etc. Todos estos conceptos se deben pensar más allá de un aspecto técnico y plantearlos de manera más filosófica. La principal ventaja de los agentes sobre los humanos que se menciona durante la clase es que los humanos pueden realizar diversas tareas, pero entregando pocos ejemplos y, en el caso de las máquinas, trabajan bajo tareas muy específicas, pero pueden manejar un montón de ejemplos.

La otra categoría son las **herramientas de IA** estas no buscan imitar el complejo pensamiento humano, su objetivo se enfoca en ayudar a los seres humanos con las diversas tareas que empleamos y estas abarcan un sinfín de áreas, desde la industria tecnológica hasta cosas tan cotidianas como comandos de voz o teclados que predicen tu escritura, las áreas involucradas también son extensas, pasando por la estadística, probabilidad, optimización, actividades comerciales, entretenimiento, etc. El alcance que tiene esta tecnología es incalculable y nadie sabe que se va a desarrollar mañana.

Existe una estructura que se utiliza a la hora de querer generar una IA para resolver tareas, el cual consiste en 3 paradigmas:

1. Modelar: Se refiere a tomar el ejemplo del mundo real y transcribirlo en un modelo gráfico de estados para poder analizarlo de manera más técnica

2. Inferir: De manera coloquial se refiere a hacer preguntas, predecir cómo se va a comportar el modelo en diversas situaciones y qué caminos va a tomar

3. Aprendizaje: Básicamente es el esqueleto del modelo sin los parámetros que se obtuvieron al momento de inferir, pero una vez que se ingresa la “data” se crean los parámetros que se plantearon al momento de inferir

Dentro de la IA el concepto de **Machine learning** esencialmente se refiere a la forma de “enseñanza” que se aplica al momento de ingresar “data” a un modelo. Dentro de este concepto se utiliza una categorización que va desde la inteligencia de bajo nivel hasta el alto nivel.

1. Reflejo: es información que se toma y analiza de una manera muy rápida con, relativamente, pocos datos de entrada y arroja un resultado.

2. Basado en estados: Es cuando se utilizan los famosos modelos de IA representado con círculos y flechas (estados y caminos respectivamente) Son utilizados en varios casos como:

a. Problemas de búsqueda, ósea entornos con estados muy controlados.

b. Proceso Markov de decisiones, estos involucran aleatoriedad respecto a los datos que entran en el sistema por parte del entorno.

c. Problemas de enfrentamiento, también presentan aleatoriedad, pero esta vez no solo involucran los datos de entrada si no que están diseñados para competir contra un adversario

3. Variables: Son muy similares a los basados en estados, pero estos permiten el uso de restricciones para satisfacer las necesidades de la tarea a completar también son implementadas cuando se hace uso de redes bayesianas, el ejemplo que dan en la clase es el rastreo de carros mediante sensores.

4. Lógica: Se refiere a la capacidad de una IA de recibir información y poder responder preguntas que se hagan basado en lo que “aprendió” de una manera lógica. Un ejemplo muy directo de esta categoría serían los chatbots.

Lec 01: Introduction to AI

A diferencia del primer vídeo, esta clase hace mucho énfasis a la parte “romántica” de la inteligencia artificial.

La IA, junto con su conceptualización, desarrollo y metas empiezan con los sueños y la imaginación. La IA apunta a un sinfín de metas tales como la imitación de las habilidades humanas, creación de autómatas, generar habilidades artísticas, etc. Existen diversos ejemplos en la historia y el docente incluye hasta las armaduras medievales como ejemplo de autómatas. Este ejemplo aplica mucho el alcance de la definición de autómatas e IA ya que también se hace un hincapié en el extenso historial de la fascinación que tiene el ser humano con los robots, IA y autómatas, por ejemplo, en el terreno de la ciencia ficción.

Preguntas relevantes a la hora de la conceptualización de la IA:

1. ¿Qué es un comportamiento inteligente?
2. ¿Qué es tener una mente?
3. ¿Cómo los humanos logran tener comportamientos inteligentes?

Nuevamente se toca el tema de la lógica y filosofía ¿por qué son tan importantes estas preguntas? Las ciencias cognitivas están extremadamente relacionadas con la inteligencia artificial. Ya que no solo se busca que una máquina se perciba como inteligente o simplemente imitar los comportamientos humanos, se busca que se mimetice con la humanidad, pasando desapercibida e irreconocible entre una máquina y un humano. Así que no solo debe de responder peticiones formuladas por los humanos, también es importante el **cómo** se plantean estos cuestionamientos.

Pero ¿en qué nos basamos para declarar si una máquina tiene inteligencia artificial? La respuesta a esa pregunta se tiene desde hace mucho, para lograr ese objetivo se utiliza la **prueba de Turing**, consiste en un interrogador humano que será enfrentado a otro humano y a una máquina, si el entrevistador no es capaz de distinguir quien es quien la prueba da positivo a una inteligencia artificial funcional.

Existen distintas dimensiones en la inteligencia artificial:

1. Pensar como un humano: Modelo cognitivo humano

- 2. Pensar racionalmente: Proceso de inferencia formalizado
- 3. Actuar racionalmente: Hacer lo correcto
- 4. Actuar como un humano: Exhibir comportamiento humano

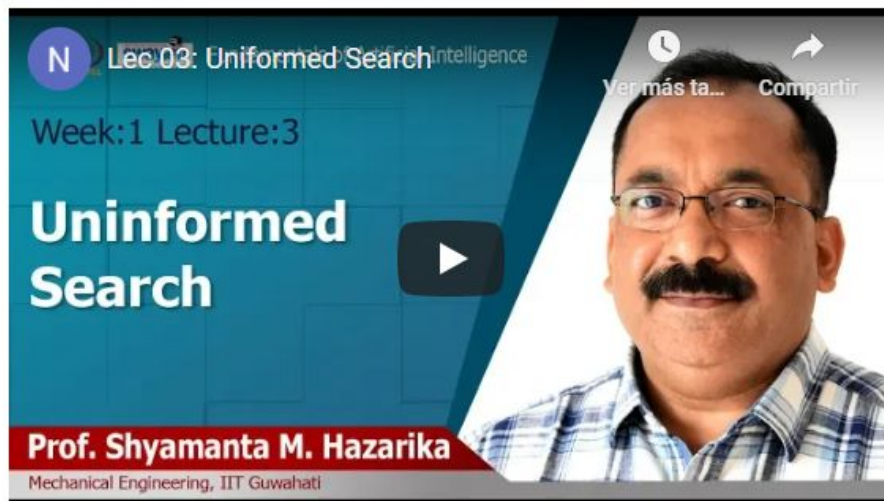
Existen diferentes obstáculos al momento de querer asemejar la lógica en una máquina

- 1. No todo comportamiento inteligente es a raíz de una deliberación lógica
- 2. ¿Cuál es el propósito de pensar? ¿Qué pensamientos debería de tener?
- 3. El conocimiento informal no es preciso, por lo tanto, la dificultad para crear un modelo es incierta
- 4. La teoría y la práctica son difíciles de poner juntas

Finalmente cabe recalcar que existen 2 conceptos importantes dentro de la inteligencia artificial

- 1. IA débil: Máquinas que actúan de manera inteligente o racional
- 2. IA Fuerte: Construcción de “personas” mediante programación, pero se considera que ese hito aún está muy lejano de su obtención

APUNTES Actividades semana 5 (Oct 19-23, 2020) Tarea

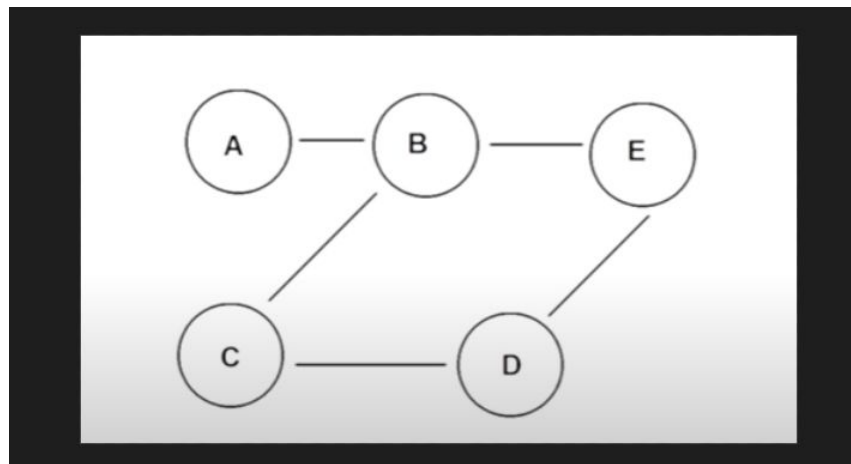


En relación a los videos del curso propuesto el primer video explica la formación de los grafos como tesis principal de la inteligencia artificial , asi como los metodos de busqueda , las principales características de los sistemas de producción , una explicación a manera de ejemplo y algunos ejemplos de métodos de búsqueda en

arboles binarios el profesor explicó uno que pudimos hacer a manera de ejemplo como complementación del aprendizaje en Python

El cual es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo con pesos en cada arista.

Como podemos ver tenemos un grafo con 5 nodos y cada nodo tiene un camino con cargas el cual se explicará en el código



Como podemos ver a continuación en el código se implementa la importación de una librería la cual facilita la implementación de los grafos .

Se declara un grafo y a manera de vector se pasa como parametro sus cargas y sus nodos y hacia qué dirección va a apuntar cada uno

Finalmente se hace la primer pregunta la cual es como ir de a -> e

```
import networkx as nx

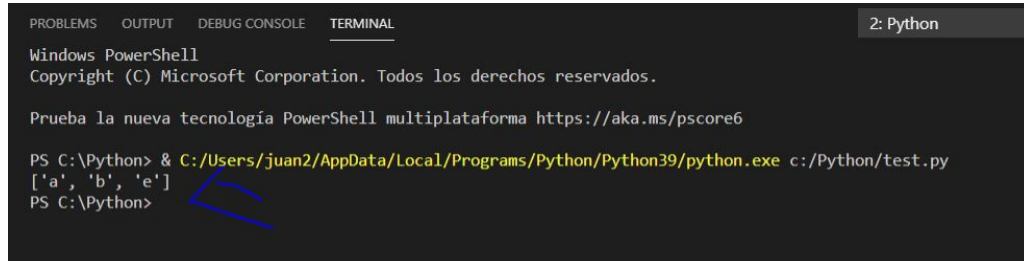
G=nx.Graph()

e=[('a','b',1), ('b','c',2), ('a','c',3), ('c','d',4), ('d','e',2), ('b','e',1)]

G.add_weighted_edges_from(e)

print(nx.dijkstra_path(G,'a','e'))
```

Si compilamos el programa la respuesta es la siguiente:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 2: Python
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Prueba la nueva tecnología PowerShell multiplataforma https://aka.ms/pscore6

PS C:\Python> & C:/Users/juan2/AppData/Local/Programs/Python/Python39/python.exe c:/Python/test.py
['a', 'b', 'e']
PS C:\Python>
```

El siguiente video explica la búsqueda uniforme y la resolución de un puzzle o juego sencillo , así como la relación del aprendizaje entre la heurística y un agente inteligente

Los métodos de búsqueda heurística disponen de alguna información sobre la proximidad de cada estado a un estado objetivo, lo que permite explorar en primer lugar los caminos más prometedores

Como complementación el profesor explica sobre la heurística pero la explica en relación a un juego de puzzle en los casos más prácticos de la formación laboral se tuvo que buscar en qué casos se aplica la heurística

- Un problema puede **no** contar con **solución exacta** debido a ambigüedades inherentes en el problema o datos disponibles.
 - Diagnóstico médico
 - Visión
- Un problema puede tener solución exacta, **pero el costo computacional por encontrarla puede ser prohibitivo.**
 - Problema del agente viajero
 - Problema de coloración
 - Juego de ajedrez

POR EJEMPLO:

- Veamos ejemplos de heurísticas para algunos problemas concretos. Para el problema del 8-puzzle tenemos la siguientes heurísticas:

- a) La basada en la distancia Manhattan (o distancia taxi). Se asocia a cada casilla un número que es la suma de las distancias horizontal y vertical a su posición en el tablero objetivo (esto es, la suma de diferencias de sus coordenadas x e y). La función heurística es la suma de las distancias de cada una de las casillas (excluyendo la que se encuentra vacía).

	2	3
1	8	4
7	6	5

Fig. 1

$H(E_i)=2$ (1 de la casilla 1 más 1 de la casilla 8)



Cabe destacar un punto importante dentro de los videos, la **heurística**, viene del griego que significa encontrar o descubrir. Una función heurística toma estados y retorna valores numéricos para cumplir con distintos propósitos.

Dentro de la IA, una función heurística estima que tan cerca está un estado a ser una meta y la principal ganancia del uso de las mismas es la reducción, a menudo sorprendente, del espacio de los estados.

Existen distintos enfoques que se utilizan en la heurística:

- Relajar un problema. Ósea, agregar acciones que no estaban en el modelo original y las cuales no necesitan una búsqueda para obtener la solución
- Número de piezas perdidas.
- Distancia Manhattan total

Estas heurísticas son las que se explican en el video pero existen muchísimas más y varían en cuanto su efectividad ya que dependen totalmente del problema al resolver. Algunos puntos que debemos tomar en cuenta a la hora de escoger una heurística adecuada son:

- Números de nodos a examinar
- La eficiencia al correr la propia heurística
- La cantidad de información que se maneja en el modelo. Ya que eso puede significar más tiempo de computación

También podemos clasificar la heurística en:

1. Admisible

- a. Nunca sobreestima el costo necesario para alcanzar una meta y jamás regresa un valor más grande que el costo real del camino y la meta más cercana de cualquier nodo

2. Inadmisible

- a. Siempre sobrestima el costo y rompe la optimización ya que no consigue el camino más corto por sobrestimar el valor

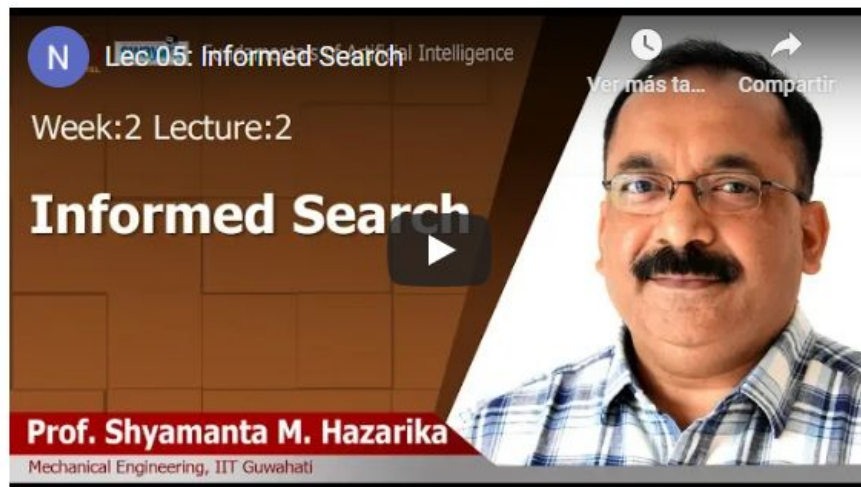
3. Compuesta

- a. El máximo de dos heurísticas admisibles también es admisible, esto toma más tiempo para computar, pero es más precisa

4. Consistente

- a. Contienen restricciones más estrictas. la heurística estimada nunca puede ser más grande que el costo individual de cada arco que se usa para llegar a la meta, ósea que el valor nunca disminuye en su camino hacia la meta

BÚSQUEDA HEURÍSTICA



Los métodos de búsqueda heurística disponen de alguna información sobre la proximidad de cada estado a un estado objetivo, lo que permite explorar en primer lugar los caminos más prometedores.

Son características de los métodos heurísticos:

- No garantizan que se encuentre una solución, aunque existan soluciones
- Si encuentran una solución, no se asegura que ésta tenga las mejores propiedades (que sea de longitud mínima o de coste óptimo).
- En algunas ocasiones (que, en general, no se podrán determinar a priori), encontrarán una solución (aceptablemente buena) en un tiempo razonable

BE FIRST SEARCH MÉTODOS DFS Y BFS

<https://medium.com/@dpthegrey/best-first-search-heuristic-search-technique-717f0b761559>

https://en.wikipedia.org/wiki/Best-first_search

<https://www.geeksforgeeks.org/best-first-search-informed-search/>

https://es.wikipedia.org/wiki/B%C3%BAsqueda_en_profundidad

https://es.wikipedia.org/wiki/B%C3%BAsqueda_en_anchura

PRINCIPAL ALGORITMO A*

El algoritmo A* ^[24] es un algoritmo de búsqueda que puede ser empleado para el cálculo de caminos mínimos en una red. Se trata de un algoritmo heurístico, ya que una de sus principales características es que hará uso de una función de evaluación heurística, mediante la cual etiquetar los diferentes nodos de la red y que servirá para determinar la probabilidad de dichos nodos de pertenecer al camino óptimo.

Esta función de evaluación que etiquetará los nodos de la red estará compuesta a su vez por otras dos funciones. Una de ellas indicará la distancia actual desde el nodo origen hasta el nodo a etiquetar, y la otra expresa la distancia estimada desde este nodo a etiquetar hasta el nodo destino hasta el que se pretende encontrar un camino mínimo. Es decir, si se pretende encontrar el camino más corto desde el nodo origen s, hasta el nodo destino t, un nodo intermedio de la red n tendría la siguiente función de evaluación f(n) como etiqueta:

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

Donde:

-g(n) indica la distancia del camino desde el nodo origen s al n.

-h(n) expresa la distancia estimada desde el nodo n hasta el nodo destino t.

h(n) se trata de una función heurística, expresa la idea de cuán lejos aún se está de alcanzar el nodo destino, y de su correcta elección dependerá en gran medida el rendimiento del algoritmo A* al aplicarlo en una red. Así, en el caso de que esta función heurística nunca sobreestime el valor de la distancia real entre el nodo y el destino, se dice que es admisible, y está garantizada la solución óptima. Por el contrario, en el caso en que la función no sea admisible no se puede garantizar el hallazgo de la solución óptima para el problema del camino más corto.

A la función de evaluación f que caracteriza a un nodo y que sirve para etiquetarlo, también se la conoce como mérito de ese nodo, y expresa la probabilidad del nodo de estar en el camino más corto. Cuanto menor sea el mérito de un nodo, es decir, cuanto menor sea el valor de su función de evaluación, más probable será que el camino más corto atraviere ese nodo. En este mérito de los diferentes todos se basará el funcionamiento del algoritmo A*. El algoritmo irá explorando nodos de la red y sus sucesores (aquellos nodos con lo que les une algún enlace) basándose en su valor de mérito. Es decir, mantendrá una lista ordenada por valor creciente de mérito de los nodos que pueden ser explorados, y de ahí seleccionará el de menor valor, que será el primero de la lista. El algoritmo empezará analizando el nodo que se toma como origen para el problema del camino más corto. Calculará su mérito, y a continuación pasará a explorar sus nodos sucesores, es decir, los nodos con los que esté unido por un enlace este nodo fuente. Para estos nodos se calculará su mérito, y se seleccionará aquél de ellos que presente un menor valor, que será el que se someterá

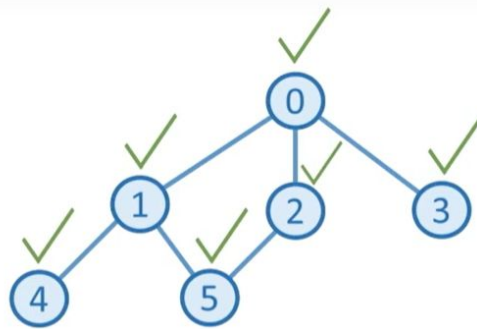
a análisis. Ahora el algoritmo continuará su ejecución explorando los nodos vecinos de ese nodo con mérito menor, y así sucesivamente, hasta llegar al caso donde el nodo a analizar sea el nodo destino del problema, momento en que el algoritmo termina y ya se dispone de la solución. Para llevar a cabo este funcionamiento será necesario un registro donde el algoritmo irá guardando el conjunto de nodos que han sido explorados con sus valores correspondientes de mérito, y de donde se irán eliminando aquellos que sean seleccionados para ser analizados.

Algoritmo A*

- 1.- Establecer el nodo s como origen. Hacer $f(s)=0$, y $f(i)=\infty$ para todos los nodos i diferentes del nodo s. Iniciar el conjunto Q vacío.
- 2.- Calcular el valor de $f(s)$ y mover el nodo s al conjunto Q .
- 3.- Seleccionar el nodo i del conjunto Q que presente menor valor de la función $f(i)$ y eliminarlo del conjunto Q.
- 4.- Analizar los nodos vecinos j de i. Para cada enlace (i, j) con coste c_{ij} hacer:
 - 4.1.-Calcular: $f(j)=g(i)+c_{ij}+h(j)$
 - 4.2.-Si $f(j) < f(j)$,
 - 4.2.1.-Actualizar la etiqueta de j y su nuevo valor será: $f(j)= g(i)+c_{ij}+h(i)$
 - 4.2.2.-Insertar el nodo j en Q
 - 4.3.-Si $f(j) \geq f(j)$
 - 4.3.1.-Dejar la etiqueta de j como está, con su valor $f(j)$
- 5.- Si Q está vacío el algoritmo se termina. Si no está vacío, volver al paso 3.

EJEMPLO DE DFS EN PYTHON

Una Búsqueda en profundidad (en inglés DFS o *Depth First Search*) es un algoritmo de búsqueda no informada utilizado para recorrer todos los nodos de un grafo o árbol (teoría de grafos) de manera ordenada, pero no uniforme. Su funcionamiento consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto. Cuando ya no quedan más nodos que visitar en dicho camino, regresa (**Backtracking**), de modo que repite el mismo proceso con cada uno de los hermanos del nodo ya procesado.



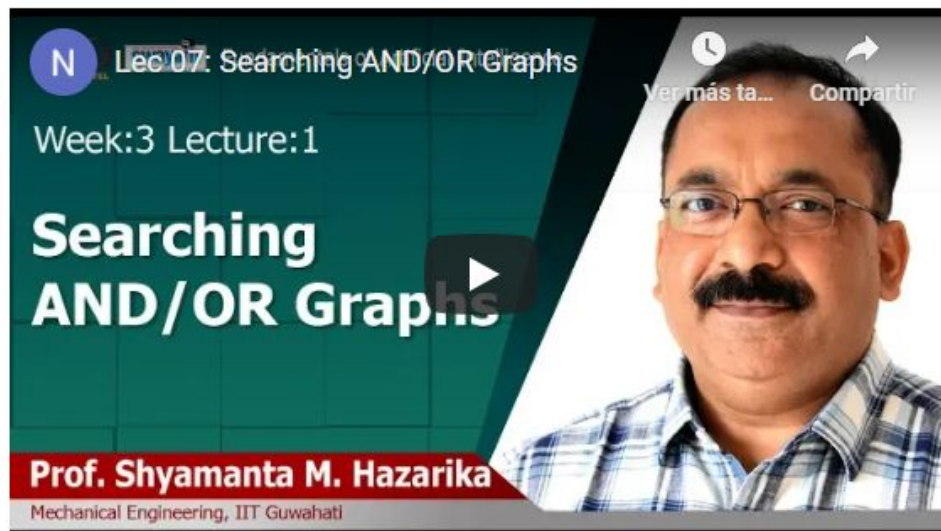
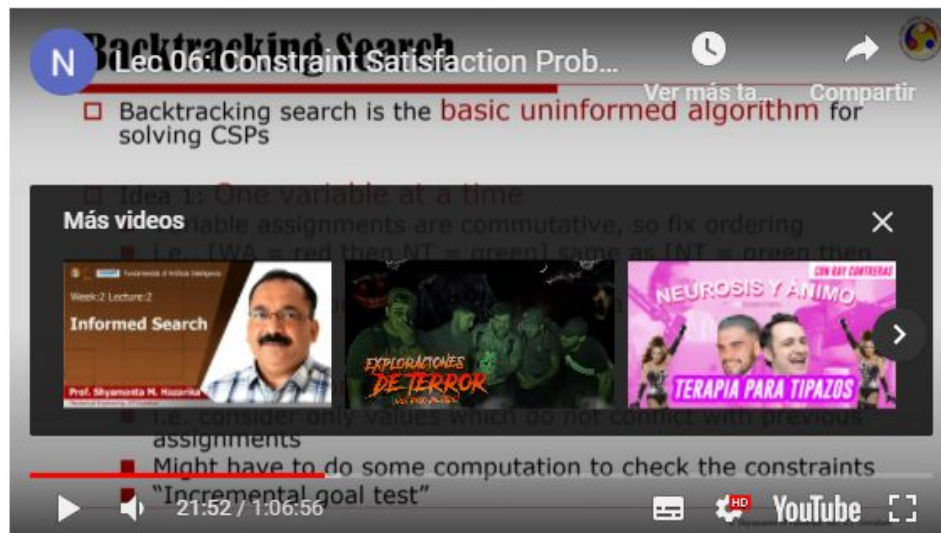
	0	1	2	3	4	5
0	0	1	1	1	0	0
1	1	0	0	0	1	1
2	1	0	0	0	0	1
3	1	0	0	0	0	0
4	0	1	0	0	0	0
5	0	1	1	0	0	0

Stack

Step 12:

--	--	--	--	--	--

Popped: 4.2.5.1.3.0



https://www.uaeh.edu.mx/investigacion/productos/6524/analisis_de_algoritmos_de_busqueda_en_espacio_de_estados.pdf

Tipo de búsqueda ciega:

- 1. Exhaustiva**
- 2. Aleatoria (Genera-y-Prueba)**
- 3. Por Profundidad (Depth-First)**
- 4. Por Amplitud (Breath-First)**
- 5. Costo Uniforme**
- 6. Limitada por Profundidad**
- 7. Profundización Iterativa**
- 8. Bidireccional**

Tipo de búsqueda heurística:

- 1. Búsqueda Tacaña (Greedy Search)**
- 2. Búsqueda A***
- 3. Templado Simulado**
- 4. Búsqueda Tabú**
- 5. Búsqueda Basada en Restricciones**

EJEMPLOS DE LOS MÉTODOS HEURÍSTICOS ANTERIORES

ALGORITMO A*

```
ABIERTOS := [INICIAL] //inicialización
CERRADOS := []
f'(INICIAL) := h'(INICIAL)
repetir
    si ABIERTOS = [] entonces FALLO
    si no // quedan nodos
        extraer MEJORNODO de ABIERTOS con f' mínima
        // cola de prioridad
        mover MEJORNODO de ABIERTOS a CERRADOS
        si MEJOR NODO contiene estado_objetivo entonces
            SOLUCION_ENCONTRADA := TRUE
        sí no
            generar SUCESTORES de MEJORNODO
            para cada SUCESOR hacer TRATAR_SUCESOR
hasta SOLUCION_ENCONTRADA o FALLO
```

ALGORITMO DFS

```
DFS_Visitar(nodo  $u$ , int  $tiempo$ )  
    estado[ $u$ ]  $\leftarrow$  VISITADO  
    tiempo  $\leftarrow$  tiempo + 1  
    d[ $u$ ]  $\leftarrow$  tiempo  
    PARA CADA  $v \in$  Vecinos[ $u$ ] HACER  
        SI estado[ $v$ ] = NO_VISITADO ENTONCES  
            padre[ $v$ ]  $\leftarrow$   $u$   
            DFS_Visitar( $v$ , tiempo)  
    estado[ $u$ ]  $\leftarrow$  TERMINADO  
    tiempo  $\leftarrow$  tiempo + 1  
    f[ $u$ ]  $\leftarrow$  tiempo
```

ALGORITMO BFS

```
BFS(grafo G, nodo_fuente s)

{

    // recorremos todos los vértices del grafo inicializándolos a NO_VISITADO,

    // distancia INFINITA y padre de cada nodo NULL

    for u ∈ V[G] do

    {

        estado[u] = NO_VISITADO;

        distancia[u] = INFINITO; /* distancia infinita si el nodo no es alcanzable */

        padre[u] = NULL;

    }

    estado[s] = VISITADO;

    distancia[s] = 0;

    padre[s] = NULL;

    CrearCola(Q); /* nos aseguramos que la cola está vacía */

    Encolar(Q, s);

    while !vacía(Q) do

    {

        // extraemos el nodo u de la cola Q y exploramos todos sus nodos adyacentes

        u = extraer(Q);

        for v ∈ adyacencia[u] do

        {

            if estado[v] == NO_VISITADO then

            {

                estado[v] = VISITADO;

                distancia[v] = distancia[u] + 1;

                padre[v] = u;

                Encolar(Q, v);

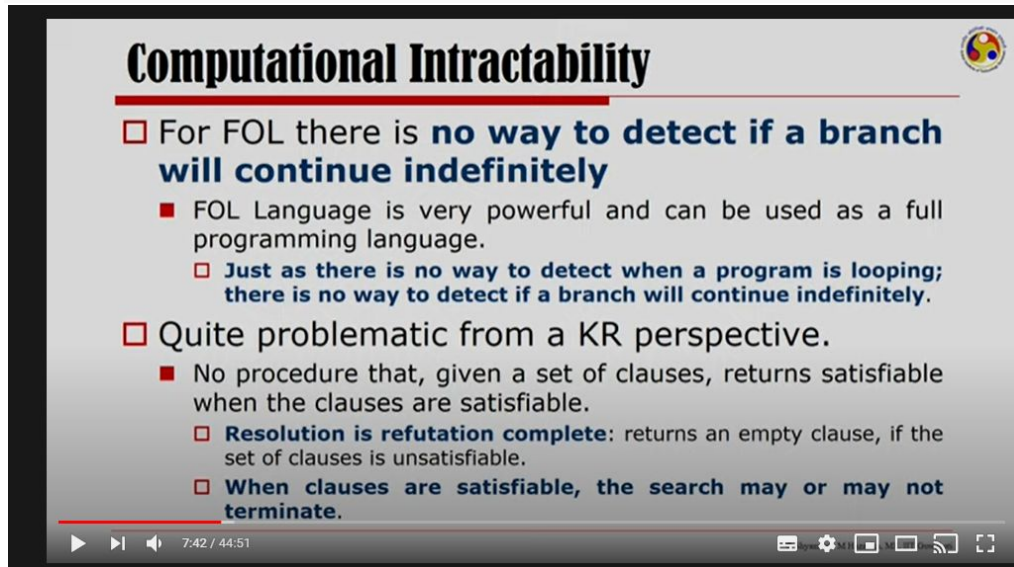
            }

        }

    }

}
```

APUNTES SEMANA 12 VIDEOS



Computational Intractability

- For FOL there is **no way to detect if a branch will continue indefinitely**
 - FOL Language is very powerful and can be used as a full programming language.
 - **Just as there is no way to detect when a program is looping; there is no way to detect if a branch will continue indefinitely.**
- Quite problematic from a KR perspective.
 - No procedure that, given a set of clauses, returns satisfiable when the clauses are satisfiable.
 - **Resolution is refutation complete:** returns an empty clause, if the set of clauses is unsatisfiable.
 - **When clauses are satisfiable, the search may or may not terminate.**

7:42 / 44:51

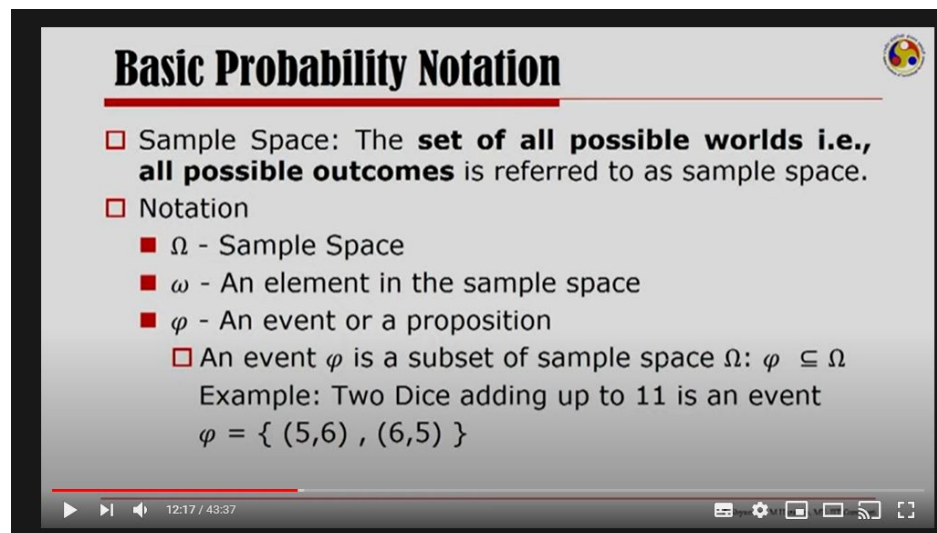
En inteligencia artificial , un sistema de razonamiento procedimental (PRS) es un marco para construir sistemas de razonamiento en tiempo real que pueden realizar tareas complejas en entornos dinámicos. Se basa en la noción de un agente racional o un agente inteligente que utiliza el modelo de software creencia-deseo-intención .

Una aplicación de usuario se define predominantemente y se proporciona a un sistema PRS un conjunto de áreas de conocimiento . Cada área de conocimiento es una pieza de conocimiento procedimental que especifica cómo hacer algo, por ejemplo, cómo navegar por un pasillo o cómo planificar una ruta (en contraste con las arquitecturas robóticas donde el programador solo proporciona un modelo de los estados de la mundo son y cómo las acciones primitivas del agente les afectan). Dicho programa, junto con un intérprete PRS , se utiliza para controlar al agente.

El intérprete es responsable de mantener creencias sobre el estado mundial, elegir qué objetivos intentar alcanzar a continuación y elegir qué área de conocimiento aplicar en la situación actual. La forma exacta en que se realizan estas operaciones puede depender de las áreas de conocimiento de metanivel específicas del dominio . A diferencia de los sistemas de planificación de IA tradicionales que generan un plan completo al principio y replanifican si suceden cosas inesperadas, PRS intercala la planificación y la realización de acciones en el mundo. En cualquier momento, es posible que el sistema solo tenga un plan parcialmente especificado para el futuro.

El PRS se basa en el BDI o marco de creencias-deseo-intención para agentes inteligentes. Las creencias consisten en lo que el agente cree que es cierto sobre el estado actual del mundo, los deseos consisten en las metas del agente y las intenciones consisten en los planes actuales del agente para lograr esas metas. Además, cada uno de estos tres componentes suele estar representado explícitamente en algún lugar de la memoria del agente PRS en tiempo de ejecución, lo que contrasta con los sistemas puramente reactivos, como la arquitectura de subsunción .

VIDEO 2



Poole, D. y Mackworth, A. (2010). Razonamiento bajo incertidumbre. En Inteligencia artificial: fundamentos de los agentes computacionales (págs. 219-280). Cambridge: Cambridge University Press. doi: 10.1017 / CBO9780511794797.007

–Pierre Simon de Laplace [1812]

Todo el tiempo, los agentes se ven obligados a tomar decisiones basadas en información incompleta. Incluso cuando un agente detecta el mundo para obtener más información, rara vez descubre el estado exacto del mundo. Un robot no sabe exactamente dónde está un objeto. Un médico no sabe exactamente qué le pasa a un paciente. Un maestro no sabe exactamente lo que entiende un alumno. Cuando los agentes inteligentes deben tomar decisiones, deben utilizar cualquier información que tengan. Este capítulo considera el razonamiento bajo incertidumbre: determinar lo que es cierto en el mundo basado en observaciones del mundo. Esto se utiliza en el capítulo 9 como base para actuar en condiciones de incertidumbre, donde el agente debe tomar decisiones sobre qué acción tomar aunque no pueda predecir con precisión los resultados de sus acciones. Este capítulo comienza con probabilidad,

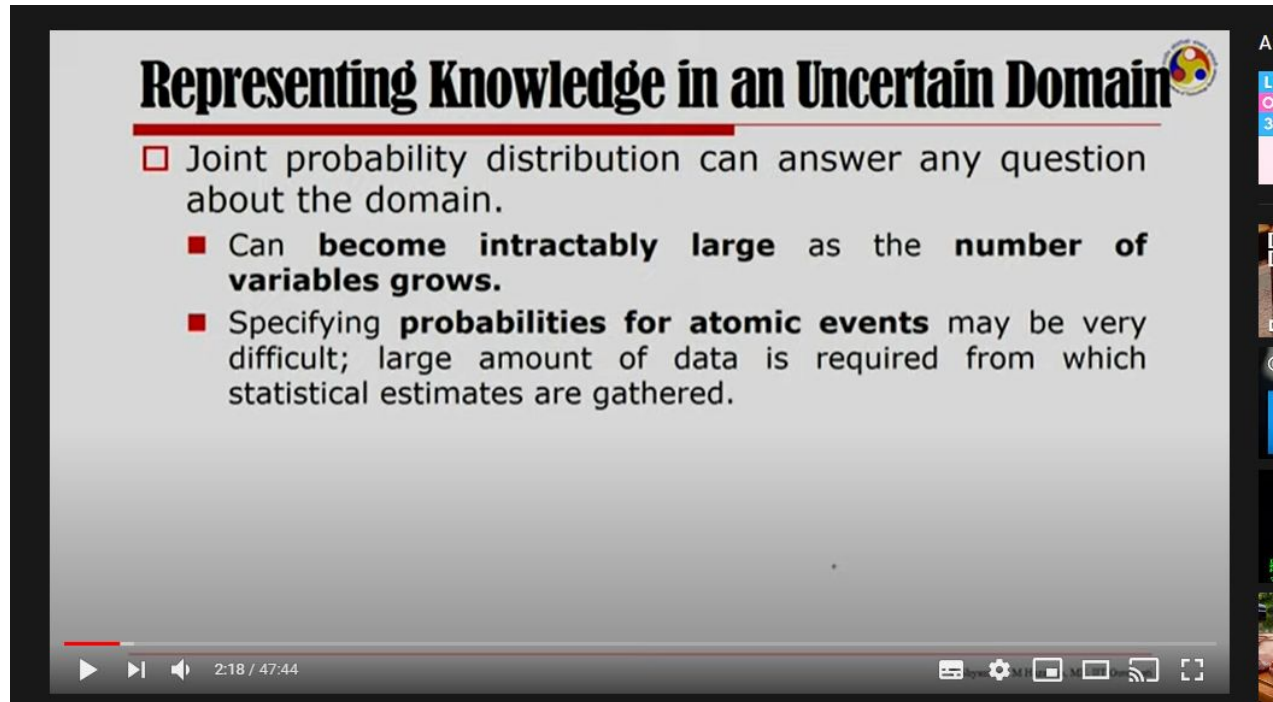
¿Por qué razonar probabilísticamente?

- En muchos dominios de problemas no es posible crear modelos completos y consistentes del mundo. Por lo tanto, los agentes (y las personas) deben actuar en mundos inciertos (que es el mundo real).
- Quiere que un agente tome decisiones racionales incluso cuando no hay suficiente información para demostrar que una acción funcionará.
- Algunas de las razones para razonar bajo incertidumbre:
 - **Verdadera incertidumbre** . Por ejemplo, lanzar una moneda.
 - **Ignorancia teórica** . No existe una teoría completa que se conozca sobre el dominio del problema. Por ejemplo, diagnóstico médico.
 - **Pereza** . El espacio de factores relevantes es muy grande y requeriría demasiado trabajo para enumerar el conjunto completo de antecedentes y consecuentes. Además, sería demasiado difícil usar las enormes reglas que resultaron.
 - **Ignorancia práctica** . No está seguro de un individuo en particular en el dominio porque no se ha recopilado toda la información necesaria para ese individuo.
- La teoría de la probabilidad servirá como lenguaje formal para representar y razonar con conocimiento incierto.

Representar creencias sobre propuestas

- En lugar de razonar sobre la verdad o falsedad de una proposición, razonar sobre la creencia de que una proposición o evento es verdadero o falso
- Para cada proposición o evento primitivo, agregue un **grado de creencia** a la oración
- Utilizar **la teoría de la probabilidad** como un medio formal para manipular los grados de creencia.

- Dada una proposición, A , asigne una probabilidad, $P(A)$, tal que $0 \leq P(A) \leq 1$, donde si A es verdadera, $P(A) = 1$, y si A es falsa, $P(A) = 0$. La proposición A debe ser verdadera o falsa, pero $P(A)$ resume nuestro grado de creencia en que A es verdadero / falso.
- Ejemplos
 - $P(\text{Clima} = \text{Soleado}) = 0.7$ significa que creemos que el clima será soleado con un 70% de certeza. En este caso, el tiempo es una variable aleatoria que puede tomar valores en un dominio como {soleado, lluvioso, nevado, nublado}.
 - $P(\text{Cavidad} = \text{Verdadero}) = 0.05$ significa que creemos que hay un 5% de probabilidad de que una persona tenga una caries. Cavity es una variable aleatoria booleana ya que puede tomar posibles valores *Verdadero* y *Falso*.
 - Ejemplo: $P(A = a \wedge B = b) = P(A = a, B = b) = 0.2$, donde $A = \text{My_Mood}$, $a = \text{happy}$, $B = \text{Weather}$, $b = \text{rainy}$, significa que hay 20 % de probabilidad de que cuando llueve mi estado de ánimo sea feliz.
- Obtención e interpretación de probabilidades
Existen varios sentidos en los que se pueden obtener e interpretar probabilidades, entre ellos los siguientes:
 - **Interpretación frecuentista**
La probabilidad es una propiedad de una población de eventos similares. Por ejemplo, si el conjunto $S = P \cup N$, y la intersección $P \cap N$ es el conjunto vacío, entonces la probabilidad de que un objeto esté en el conjunto P es $|P| / |S|$. Por lo tanto, en esta interpretación las probabilidades provienen de experimentos y de la determinación de la población asociada con una proposición dada.
 - **Interpretación subjetivista**
Un grado subjetivo de creencia en una proposición o en la ocurrencia de un evento. Por ejemplo, la probabilidad de que apruebe el examen final en función de su propia evaluación subjetiva de la cantidad de estudios que ha realizado y su comprensión del material. Por tanto, en esta interpretación, las probabilidades caracterizan las creencias del agente.
- Supondremos que en un dominio de problema dado, el programador y el experto identifican todas las variables proposicionales relevantes que se necesitan para razonar sobre el dominio. Cada uno de estos se representará como una **variable aleatoria**, es decir, una variable que puede tomar valores de un conjunto de valores mutuamente excluyentes y exhaustivos llamado **espacio muestral** o **partición** de la variable aleatoria. Por lo general, esto significa un espacio muestral { *Verdadero*, *Falso* }. Por ejemplo, la proposición *Cavity* tiene valores posibles *Verdadero* y *Falso* indicando si un paciente determinado tiene una caries o no. Una variable aleatoria que tiene Verdadero y Falso como valores posibles se denomina **variable aleatoria booleana**.
De manera más general, las proposiciones pueden incluir el predicado de igualdad con variables aleatorias y los posibles valores que pueden tener. Por ejemplo, podríamos tener una variable aleatoria *Color* con posibles valores *rojo*, *verde*, *azul* y *otros*. Entonces $P(\text{Color} = \text{rojo})$ indica la probabilidad de que el color de un objeto dado sea rojo. De manera similar, para las variables aleatorias booleanas podemos preguntar $P(A = \text{Verdadero})$, que se abrevia como $P(A)$, y $P(A = \text{Falso})$, que se abrevia como $P(\neg A)$.



Representing Knowledge in an Uncertain Domain

- Joint probability distribution can answer any question about the domain.
 - Can **become intractably large** as the **number of variables grows**.
 - Specifying **probabilities for atomic events** may be very difficult; large amount of data is required from which statistical estimates are gathered.

2:18 / 47:44

¿Qué son las redes bayesianas?

Las redes bayesianas son un tipo de **modelo gráfico probabilístico** que se puede utilizar para construir modelos a partir de datos y / o opiniones de expertos.

Se pueden utilizar para una amplia gama de tareas que incluyen predicción, detección de anomalías, diagnóstico, conocimiento automático, razonamiento, predicción de series de tiempo y toma de decisiones en condiciones de incertidumbre. La Figura 1 a continuación muestra estas capacidades en términos de las cuatro principales disciplinas **analíticas** , **analítica descriptiva** , **analítica de diagnóstico** , **analítica predictiva** y **analítica prescriptiva** .

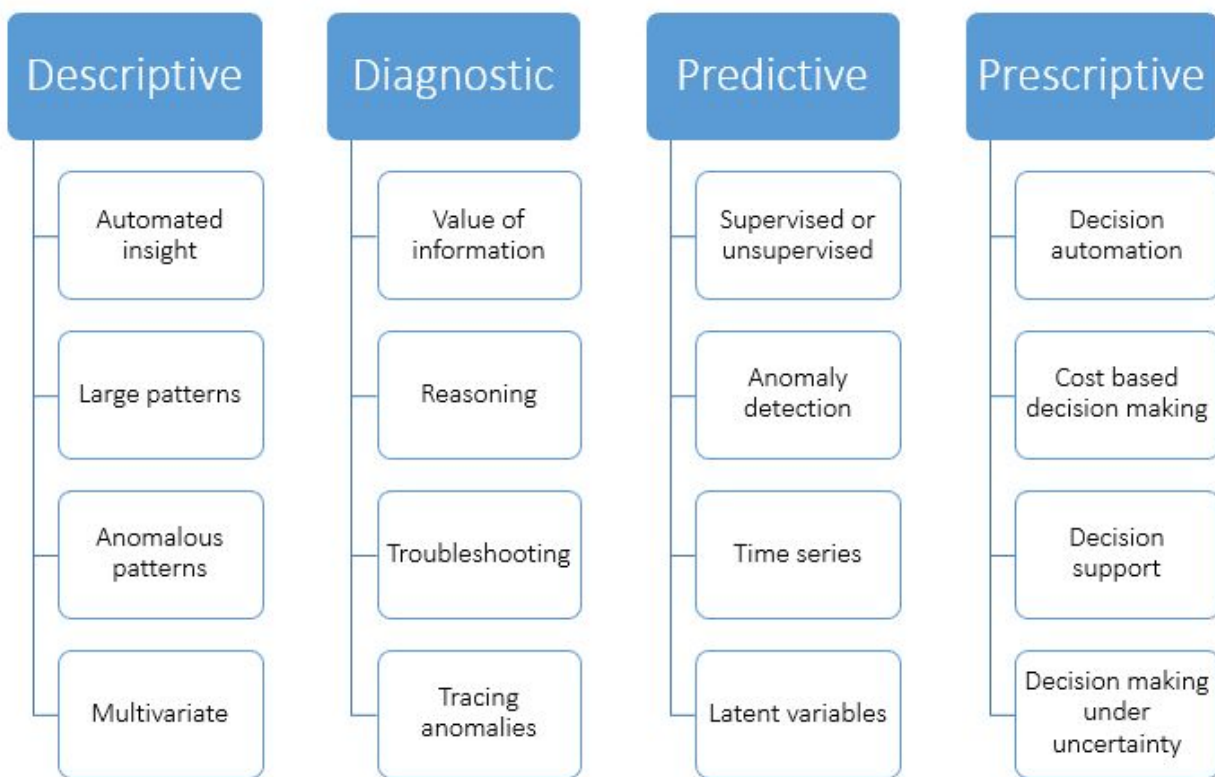


Figura 1 - Análisis descriptivo, diagnóstico, predictivo y prescriptivo con redes bayesianas

También se les conoce comúnmente como **redes de Bayes** , **redes de creencias** y, a veces, **redes causales** .

Probabilístico

Las redes bayesianas son probabilísticas porque se construyen a partir de distribuciones de probabilidad y también usan las leyes de probabilidad para la **predicción** y **detección de anomalías** , para el razonamiento y **diagnóstico** , la **toma de decisiones bajo incertidumbre** y la predicción de **series de tiempo** .

Gráfico

Las redes bayesianas se pueden representar gráficamente como se muestra en la *Figura 2* , que muestra la conocida *red de Asia* . Aunque visualizar la estructura de una red bayesiana es opcional, es una excelente manera de comprender un modelo.

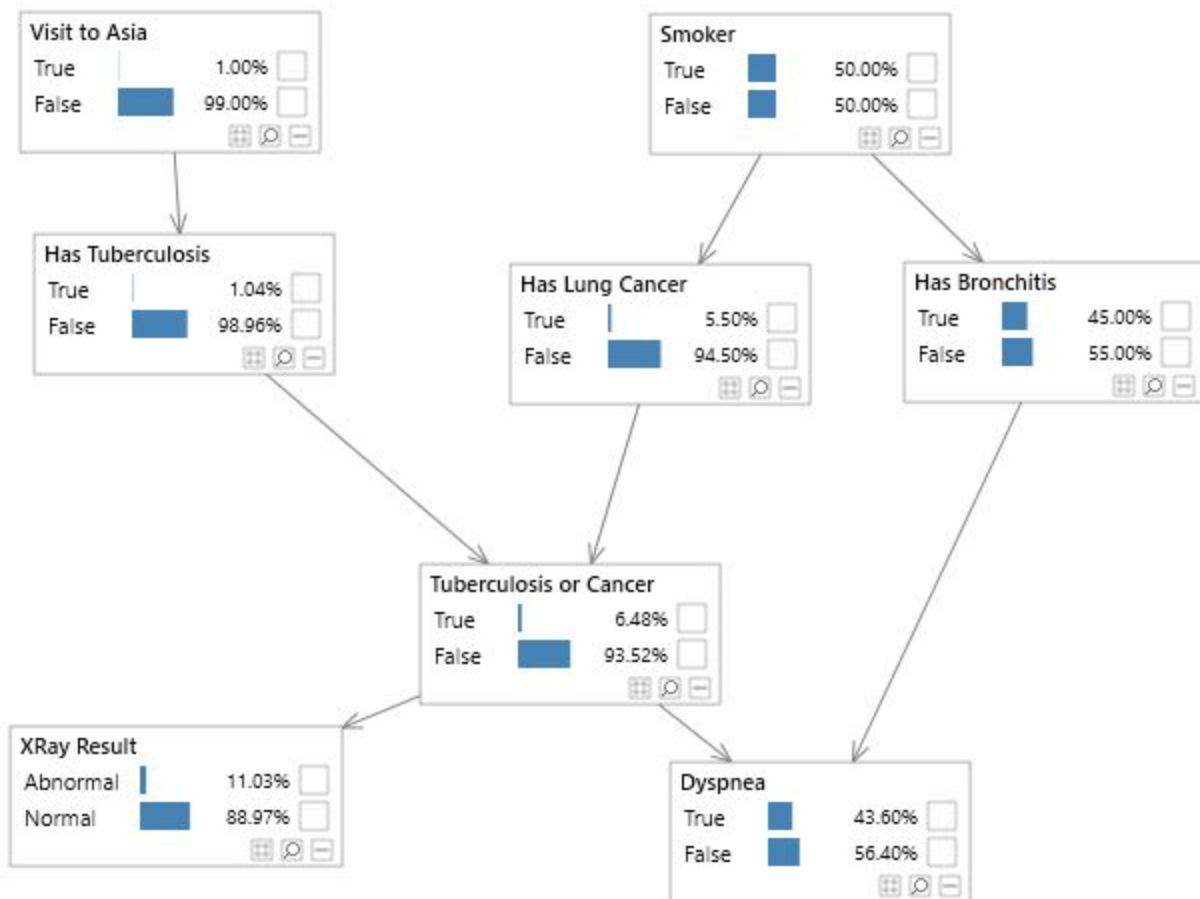


Figura 2: una red bayesiana simple, conocida como red de Asia.

Versión interactiva

Una red bayesiana es un gráfico que se compone de **nodos** y **enlaces** dirigidos entre ellos.

Nodos

En muchas redes bayesianas, cada nodo representa una **variable** como la altura, la edad o el sexo de una persona. Una variable puede ser discreta, como Sexo = {Mujer, Hombre} o puede ser continua, como la edad de alguien.

En Bayes Server, cada nodo puede contener múltiples variables. Llamamos nodos con más de una **variable** **nodos multivariantes**.

Los nodos y enlaces forman la estructura de la red bayesiana, y la llamamos **especificación estructural**.

Bayes Server admite variables tanto discretas como continuas.

Discreto

Una variable discreta es aquella con un conjunto de estados mutuamente excluyentes como Género = {Mujer, Hombre}.

Continuo

Bayes Server admite variables continuas con **distribuciones lineales gaussianas condicionales** (CLG). Esto simplemente significa que las distribuciones continuas pueden depender unas de otras (son multivariadas) y también pueden depender de una o más variables discretas.

Aunque los gaussianos pueden parecer restrictivos al principio, de hecho, las distribuciones CLG pueden modelar relaciones complejas no lineales (incluso jerárquicas) en los datos. Bayes Server también admite **variables latentes** que pueden modelar relaciones ocultas (extracción automática de características, similar a las capas ocultas en una red neuronal profunda).

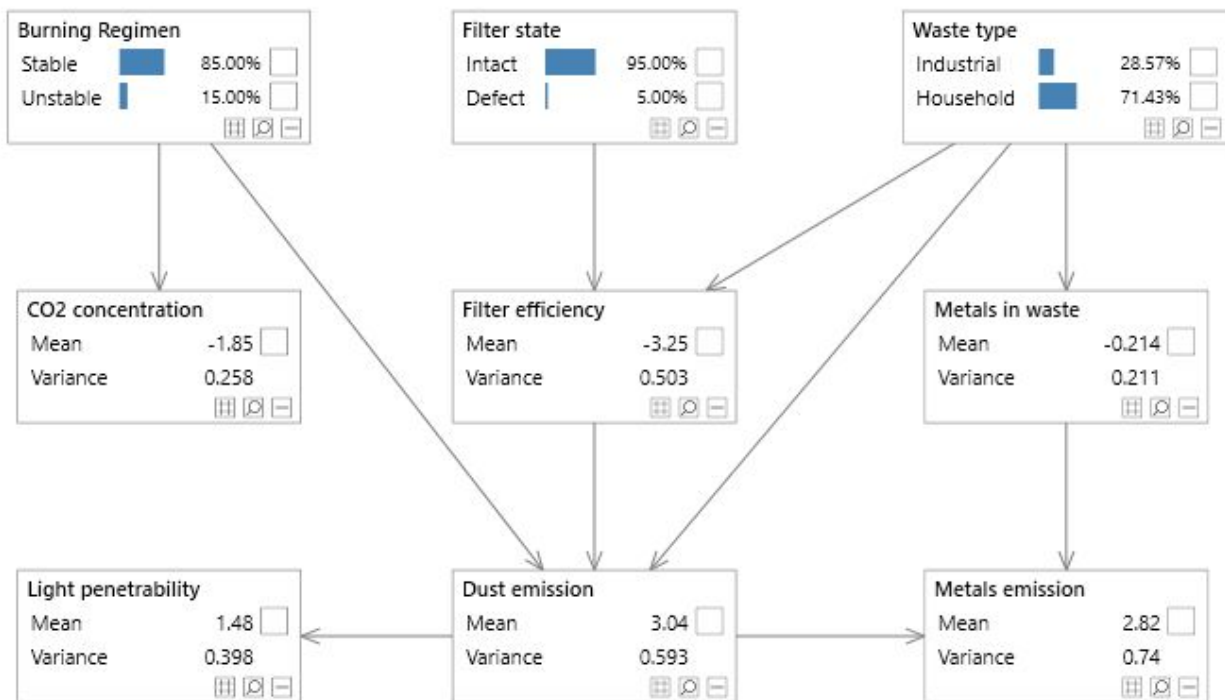


Figura 3 - Una red bayesiana simple con variables tanto discretas como continuas, conocida como la red Waste.

[Versión interactiva](#)

Enlaces

Se agregan enlaces entre los nodos para indicar que un nodo influye directamente en el otro. Cuando no existe un enlace entre dos nodos, esto no significa que sean completamente independientes, ya que pueden estar conectados a través de otros nodos. Sin embargo, pueden volverse dependientes o independientes dependiendo de la evidencia que se establezca en otros nodos.

Aunque los enlaces en una red bayesiana están dirigidos, la información puede fluir en ambos sentidos (de acuerdo con reglas estrictas que se describen más adelante).

Aprendizaje estructural

Bayes Server incluye un algoritmo de [aprendizaje estructural](#) para redes bayesianas, que puede determinar automáticamente los enlaces necesarios a partir de los datos.

Tenga en cuenta que el aprendizaje estructural a menudo no es necesario, ya que hay muchas estructuras bien conocidas que pueden resolver muchos problemas.

Selección de características

Bayes Server admite un algoritmo de **selección de** funciones que puede ayudar a determinar qué variables tienen más probabilidades de influir en otra. Esto puede resultar útil para determinar la estructura de un modelo.

Otra técnica útil es hacer uso de **variables latentes** para extraer características automáticamente como parte del modelo.

Gráfico acíclico dirigido (DAG)

Una red bayesiana es un tipo de gráfico llamado gráfico **acíclico dirigido** o **DAG**. Un Dag es un gráfico con enlaces dirigidos y uno que no contiene ciclos dirigidos.

Ciclos dirigidos

Un ciclo dirigido en un gráfico es una ruta que comienza y termina en el mismo nodo donde la ruta tomada solo puede ser a lo largo de la dirección de los enlaces.

Notación

En este punto, es útil introducir alguna notación matemática simple para variables y distribuciones de probabilidad.

Las variables se representan con letras mayúsculas (A, B, C) y sus valores con letras minúsculas (a, b, c). Si $A = a$ decimos que A ha sido instanciado.

Un conjunto de variables se denota con una letra mayúscula en negrita (\mathbf{X}) y una instanciación particular con una letra minúscula en negrita (\mathbf{x}). Por ejemplo, si \mathbf{X} representa las variables A, B, C, entonces \mathbf{x} es la instanciación a, b, c. El número de variables en \mathbf{X} se denota $|\mathbf{X}|$. El número de estados posibles de una variable discreta A se denota $|A|$.

La notación $pa(X)$ se usa para referirse a los padres de X en un gráfico. Por ejemplo, en la **Figura 2**, $pa(\text{disnea}) = (\text{tuberculosis o cáncer, tiene bronquitis})$.

Usamos $P(A)$ para denotar la probabilidad de A.

Usamos $P(A, B)$ para denotar la probabilidad conjunta de A y B.

Usamos $P(A | B)$ para denotar la probabilidad condicional de A dado B.

Probabilidad

$P(A)$ se usa para denotar la probabilidad de A. Por ejemplo, si A es discreto con estados {Verdadero, Falso}, entonces $P(A)$ podría ser igual a $[0.2, 0.8]$. Es decir, 20% de probabilidad de ser cierto, 80% de probabilidad de ser falso.

Probabilidad conjunta

Una probabilidad conjunta se refiere a la probabilidad de que ocurran juntas más de una variable, como la probabilidad de A y B, denominada $P(A, B)$.

A continuación se muestra un ejemplo de distribución de probabilidad conjunta para las variables **Raining** ad **Windy**. Por ejemplo, la probabilidad de que haga viento y no llueva es de 0,16 (o 16%).

Para las variables discretas, las entradas de probabilidad conjunta suman uno.

Raining	Windy = False	Windy = True
False	0.64	0.16
True	0.1	0.1

Si dos variables son independientes (es decir, no relacionadas), entonces $P(A, B) = P(A) P(B)$.

La probabilidad condicional

La probabilidad condicional es la probabilidad de una variable (o conjunto de variables) dada otra variable (o conjunto de variables), denotado $P(A | B)$.

Por ejemplo, la probabilidad de que Windy sea True, dado que Raining es True, podría ser igual al 50%.

Esto se denotará $P(\text{ventoso} = \text{verdadero} | \text{lloviendo} = \text{verdadero}) = 50\%$.

Probabilidad marginal

Una probabilidad marginal es una distribución formada al calcular el subconjunto de una distribución de probabilidad más grande.

Si tenemos una distribución conjunta $P(\text{lloviendo, ventoso})$ y alguien nos pregunta cuál es la probabilidad de que llueva, necesitamos $P(\text{lloviendo})$, no $P(\text{lloviendo, ventoso})$. Para calcular $P(\text{lloviendo})$, simplemente podemos sumar todos los valores de Lluvia = Falso y Lluvia = Verdadero, como se muestra a continuación.

$P(\text{Raining, Windy})$

Raining	Windy = False	Windy = True	Sum
False	0.64	0.16	0.8
True	0.1	0.1	0.2



$P(\text{Raining})$

Raining = False	Raining = True
0.8	0.2

Este proceso se llama **marginación**.

Cuando consultamos un nodo en una red bayesiana, el resultado a menudo se denomina **marginal**.

Para las variables discretas sumamos, mientras que para las continuas integramos.

Se cree que el término marginal surgió porque las tablas de probabilidad conjunta escritas en libros mayores se sumaron a lo largo de filas o columnas, y el resultado se escribió en los márgenes del libro mayor.

A continuación se muestra un ejemplo más complicado que implica la marginación de más de una variable.

P(A,B,C,D)

B	C	D	A = True	A = False
True	True	True	0.0036	0.0054
True	True	False	0.0098	0.0252
True	False	True	0.0024	0.0486
True	False	False	0.0042	0.1008
False	True	True	0.0256	0.0864
False	True	False	0.0432	0.1728
False	False	True	0.0064	0.2016
False	False	False	0.0048	0.2592



P(A,C)

C	A = True	A = False
True	0.0822	0.2898
False	0.0178	0.6102

Distribuciones

Una vez que se ha definido la estructura (es decir, nodos y enlaces), una red bayesiana requiere que se asigne una distribución de probabilidad a cada nodo.

Tenga en cuenta que es un poco más complicado para los nodos de series de tiempo y los nodos ruidosos, ya que normalmente requieren múltiples distribuciones.

Cada nodo X en una red bayesiana requiere una distribución de probabilidad $P(X | pa(X))$.

Tenga en cuenta que si un nodo X no tiene padres, $pa(X)$ está vacío y la distribución requerida es solo $P(X)$, a veces denominada anterior.

Ésta es la probabilidad de sí mismo dados sus nodos padres. Así, por ejemplo, en la figura 2, el nodo Disnea tiene dos padres (Tuberculosis o Cáncer, Tiene Bronquitis), y por lo tanto requiere la distribución de probabilidad $P(\text{Disnea} \mid \text{Tuberculosis o Cáncer}, \text{Tiene Bronquitis})$, un ejemplo de lo cual se muestra en la tabla 1. Este tipo de distribución de probabilidad se conoce como distribución de probabilidad condicional, y para las variables discretas, cada fila sumará 1.

La dirección de un enlace en una red bayesiana por sí sola no restringe el flujo de información de un nodo a otro o viceversa, sin embargo, cambia las distribuciones de probabilidad requeridas, ya que, como se describió anteriormente, la distribución de un nodo está condicionada a sus padres.

Tiene bronquitis	Tuberculosis o cáncer	Disnea = Verdadero	Disnea = Falso
Cierto	Cierto	0,9	0,1
Cierto	Falso	0,8	0,2
Falso	Cierto	0,7	0,3
Falso	Falso	0,1	0,9

Tabla 1 - $P(\text{disnea} \mid \text{tuberculosis o cáncer}, \text{tiene bronquitis})$

Las distribuciones en una red bayesiana se pueden [aprender a partir de los datos](#) o se pueden especificar manualmente mediante la opinión de un experto.

Aprendizaje de parámetros

Hay varias formas de determinar las distribuciones necesarias.

- Aprenda de los datos
- Especificarlos manualmente (solicítelos) utilizando expertos.

- Una mezcla de ambos.

Bayes Server incluye un algoritmo de **aprendizaje de parámetros** extremadamente flexible . Las características incluyen:

- Los datos faltantes son totalmente compatibles
- Soporte para variables latentes tanto discretas como continuas
- Los registros se pueden ponderar (por ejemplo, 1000 o 0,2)
- Algunos nodos se pueden aprender mientras que otros no
- Los anteriores son compatibles
- Aprendizaje multiproceso y / o distribuido.

Consulte el tema de ayuda de [aprendizaje de parámetros](#) para obtener más información.

Aprender en línea

El aprendizaje en línea (también conocido como adaptación) con redes bayesianas, permite al usuario o al desarrollador de API actualizar las distribuciones en una red bayesiana cada registro a la vez. Esto utiliza un enfoque completamente bayesiano.

A menudo, se utiliza un enfoque de aprendizaje por lotes en datos históricos periódicamente, y se utiliza un algoritmo en línea para mantener el modelo actualizado entre el aprendizaje por lotes.

Para obtener más información, consulte [Aprendizaje en línea](#) .

Evidencia

Las cosas que sabemos (evidencia) se pueden configurar en cada nodo / variable en una red bayesiana. Por ejemplo, si sabemos que alguien es fumador, podemos establecer el estado del nodo Fumador en Verdadero. Del mismo modo, si una red contuviera variables continuas, podríamos establecer pruebas como Edad = 37,5.

Usamos **e** para denotar evidencia establecida sobre una o más variables.

Cuando la evidencia se establece en una distribución de probabilidad, podemos reducir el número de variables en la distribución, ya que ciertas variables tienen valores conocidos y, por lo tanto, ya no son variables. Este proceso se denomina **instanciación** .

Bayes Server también admite otras técnicas relacionadas con la evidencia:

- [Evidencia retractada](#)
- [Evidencia blanda / virtual](#)

Instanciación

La siguiente figura muestra un ejemplo de instanciar una variable en una distribución de probabilidad discreta.

$P(A = \text{False}, B, C, D)$

B	C	D	A = True	A = False
True	True	True	0.0036	0.0
True	True	False	0.0098	0.0
True	False	True	0.0024	0.0
True	False	False	0.0042	0.0
False	True	True	0.0256	0.0
False	True	False	0.0432	0.0
False	False	True	0.0064	0.0
False	False	False	0.0048	0.0



$P(B, C, D)$

C	D	B=True	B=False
True	True	0.0036	0.0256
True	False	0.0098	0.0432
False	True	0.0024	0.0064
False	False	0.0042	0.0048

Tenga en cuenta que, si es necesario, podemos instanciar más de una variable a la vez, por ejemplo, $P(A = \text{Falso}, B = \text{Falso}, C, D) \Rightarrow P(C, D)$.

Tenga en cuenta que cuando se crea una instancia de una distribución de probabilidad, ya no es estrictamente una distribución de probabilidad y, por lo tanto, a menudo se la denomina **probabilidad** denotada con el símbolo griego $\hat{\Pi}_i$.

Probabilidad conjunta de una red bayesiana

Si $\mathbf{U} = \{A_1, \dots, A_n\}$ es el universo de variables (todas las variables) en una red bayesiana, y $pa(A_i)$ son los padres de A_i , entonces la distribución de probabilidad conjunta $P(\mathbf{U})$ es simplemente el producto de todas las distribuciones de probabilidad (previas y condicionales) en la red, como se muestra en la siguiente ecuación.

Esta ecuación se conoce como la regla de la cadena.

$$P(\mathbf{X}, \mathbf{e}) = \sum_{\mathbf{U} \setminus \mathbf{X}} P(\mathbf{U}, \mathbf{e}) = \sum_{\mathbf{U} \setminus \mathbf{X}} \prod_i P(U_i | pa(U_i)) \mathbf{e}$$

A partir de la distribución conjunta sobre \mathbf{U} podemos a su vez calcular cualquier consulta que nos interese (con o sin conjunto de pruebas).

Por ejemplo, si \mathbf{U} contiene variables $\{A, B, C, D, E\}$, podemos calcular cualquiera de las siguientes:

- $P(A)$ o $P(B)$, etc ...
- $P(A, B)$
- $P(A | B)$
- $P(A, B | C, D)$
- $P(A | C = \text{falso})$
- ...

El problema es que la distribución de probabilidad conjunta, particularmente sobre variables discretas, puede ser muy grande.

Considere una red con 30 variables discretas binarias. Binario simplemente significa que una variable tiene 2 estados (por ejemplo, Verdadero y Falso). La probabilidad conjunta requeriría 2^{30} entradas, que es un número muy grande. Esto no solo requeriría una gran cantidad de memoria, sino que también las consultas serían lentas.

Las redes bayesianas son una representación factorizada de la articulación completa. (Esto solo significa que muchos de los valores en la articulación completa se pueden calcular a partir de distribuciones más pequeñas). Esta propiedad utilizada junto con la **ley distributiva** permite a las redes bayesianas consultar redes con miles de nodos.

Ley distributiva

La **ley distributiva** simplemente significa que si queremos marginar la variable A podemos realizar los cálculos en el subconjunto de distribuciones que contienen A

$$\text{if } A \notin X, A \in Y, \text{ then } \sum_A \phi_X \phi_Y = \phi_X \sum_A \phi_Y$$

La ley distributiva tiene implicaciones de largo alcance para la consulta eficiente de redes bayesianas y sustenta gran parte de su poder.

Teorema de Bayes

A partir de los axiomas de probabilidad, es fácil derivar el Teorema de Bayes de la siguiente manera:

$$P(A, B) = P(A | B) P(B) = P(B | A) P(A) \Rightarrow P(A | B) = P(B | A) P(A) / P(\text{SEGUNDO})$$

El teorema de Bayes nos permite actualizar nuestra creencia en una distribución Q (sobre una o más variables), a la luz de nueva evidencia e . $P(Q | e) = P(e | Q) P(Q) / P(e)$

El término $P(Q)$ se llama la probabilidad a priori o marginal de Q , y $P(Q | e)$ se llama la probabilidad posterior de Q .

El término $P(e)$ es la **probabilidad de evidencia**, y es simplemente un factor de normalización de modo que la probabilidad resultante sume 1. El término $P(e | Q)$ a veces se denomina probabilidad de Q dado e , denotado $L(Q | e)$. Esto se debe a que, dado que sabemos e , $P(e | Q)$ es una medida de la probabilidad de que Q haya causado la evidencia.

¿Son las redes bayesianas bayesianas?

Si y no. Hacen uso del teorema de Bayes durante la inferencia y, por lo general, utilizan a priori durante el aprendizaje de parámetros por lotes. Sin embargo, no suelen utilizar un tratamiento bayesiano completo en el sentido **estadístico bayesiano** (es decir, hiper parámetros y aprendizaje caso por caso).

El asunto se confunde aún más, ya que las redes bayesianas suelen utilizar un enfoque bayesiano completo para el [aprendizaje en línea](#).

Inferencia

La inferencia es el proceso de calcular una distribución de probabilidad de interés, por ejemplo, $P(A | B = \text{Verdadero})$ o $P(A, B | C, D = \text{Verdadero})$. Los términos inferencia y consultas se utilizan indistintamente. Los siguientes términos son todas formas de inferencia que diferenciarán ligeramente la semántica.

- Predicción: se centra en inferir resultados a partir de entradas.
- Diagnóstico: inferir entradas de salidas.
- Detección de anomalías supervisada: esencialmente lo mismo que la predicción
- Detección de anomalías no supervisada: la inferencia se utiliza para calcular el $P(e)$ o, más comúnmente, el registro ($P(e)$).
- Toma de decisiones bajo incertidumbre - optimización e inferencia combinadas. Consulte [Gráficos de decisión](#) para obtener más información.

Algunos ejemplos de inferencia en la práctica:

- Dados varios síntomas, ¿qué enfermedades son más probables?
- ¿Qué tan probable es que un componente falle, dado el estado actual del sistema?
- Dado el comportamiento reciente de 2 acciones, ¿cómo se comportará juntas durante los próximos 5 pasos de tiempo?

Es importante destacar que las redes bayesianas manejan los datos faltantes durante la inferencia (y también el aprendizaje), de una manera probabilística sólida.

Inferencia exacta

Inferencia exacta es el término utilizado cuando la inferencia se realiza exactamente (sujeto a errores de redondeo numérico estándar).

La inferencia exacta es aplicable a una amplia gama de problemas, pero puede que no sea posible cuando las combinaciones / rutas se agrandan.

A menudo es posible refactorizar una red bayesiana antes de recurrir a la inferencia aproximada o utilizar un enfoque híbrido.

Inferencia aproximada

- Clase más amplia de problemas
- Determinista / no determinista
- Sin garantía de respuesta correcta

Algoritmos

Existe una gran cantidad de algoritmos de inferencia exactos y aproximados para redes bayesianas.

Bayes Server admite inferencias tanto exactas como aproximadas con redes bayesianas, redes bayesianas dinámicas y gráficos de decisión. [Algoritmos de Bayes Server](#).

Los algoritmos exactos de Bayes Server han sido objeto de más de una década de investigación para hacerlos: * Muy rápidos * Numéricamente estables * Memoria eficiente

¡Estimamos que se han necesitado más de 100 algoritmos / optimizaciones de código para que esto suceda!

Consultas

Además de consultas complejas como $P(A | B)$, $P(A, B | C, D)$, Bayes Server también admite lo siguiente:

- [Probabilidad logarítmica](#)
- [Explicación más probable](#)
- [Conflicto](#)
- [Consulta de comparación](#) para comparar escenarios.

Análisis

Bayes Server también incluye una serie de técnicas de análisis que utilizan los potentes motores de inferencia para extraer información automatizada, realizar diagnósticos y analizar y ajustar los parámetros de la red bayesiana.

- [Conocimiento automatizado](#)
- [Ajuste de parámetros](#)
- [Sensibilidad a los parámetros](#)
- [Valor de la información](#)

Redes dinámicas bayesianas

Las redes dinámicas bayesianas (DBN) se utilizan para modelar series de tiempo y secuencias. Extienden el concepto de redes bayesianas estándar con el tiempo. En

Bayes Server, el tiempo ha sido una parte nativa de la plataforma desde el día 1, por lo que incluso puede construir distribuciones de probabilidad como $P(X[t = 0], X[t + 5], Y | Z[t = 2])$ (donde t es el tiempo).

Para obtener más información, consulte el tema de ayuda de la [red dinámica bayesiana](#).

Gráficos de decisión

Una vez que haya construido un modelo, a menudo el siguiente paso es tomar una decisión basada en ese modelo. Para tomar esas decisiones, a menudo hay costos involucrados. El problema de hacer esto manualmente es que puede haber muchas decisiones diferentes que tomar, costos que compensar entre sí y todo esto en un entorno incierto (es decir, no estamos seguros de ciertos estados).

Los gráficos de decisión son una extensión de las redes bayesianas que manejan la toma de decisiones en condiciones de incertidumbre.

Para obtener más información, consulte el tema de ayuda de [gráficos de decisión](#).

Aprendizaje basado en árboles de decisión

Algunas características principales de los árboles de decisión:

- Son una de las formas más usadas para la clasificación
- Es usado como un modelo de predicción
- Son más claros y fáciles de leer gracias a su modelo gráfico

Estos árboles son prácticamente iguales a los árboles binarios que ya conocemos. La parte interesante de estos es el cómo se van generando

Todo comienza con nuestro **dataset** (conjunto de datos) en donde tendremos toda la información disponible para el sistema.

Después tenemos **entradas o inputs** (objetos o situaciones) los cuales están definidos por unos rasgos o propiedades. Y también están las **salidas o outputs** las cuales se basan normalmente con valores booleanos pero puede llegar a utilizarse valores numéricos también.

Como ya sabemos a los nodos principales se les llama **raíz o padre** y los nodos que desembocan de los anteriores se les conoce como **hojas o hijos**.

Veamos el ejemplo que nos da la lección.

Aquí podemos observar que la problemática es que debería de decidir si esperaremos por una mesa en un restaurante.

Aquí las entradas son numeradas del 1 al 10 y están acompañadas de sus rasgos.

Learning decision trees

Decide whether to wait for a table at a restaurant.

Aim is to learn a definition for the goal predicate **WillWait**, where the definition is expressed as a decision tree.

Setting this up as a learning problem; we decide what properties or attributes are available to describe examples in the domain:

1. **Alternate**: is there an alternative restaurant nearby?
2. **Bar**: is there a comfortable bar area to wait in?
3. **Fri/Sat**: is today Friday or Saturday?
4. **Hungry**: are we hungry?
5. **Patrons**: number of people in the restaurant (None, Some, Full)
6. **Price**: price range (\$, \$\$, \$\$\$)
7. **Raining**: is it raining outside?
8. **Reservation**: have we made a reservation?
9. **Type**: kind of restaurant (French, Italian, Thai, Burger)
10. **WaitEstimate**: estimated waiting time (0-10, 10-30, 30-60, >60)

17

© Shyamanta M Hazarika, MEd, IIT Guwahati

Posteriormente se debe hacer una tabla que compare las entradas con sus posibles salidas, en este caso las salidas son si esperamos por nuestra mesa o no. En ocasiones estas tablas pueden tener **impurezas**, están son generadas por error humano o datos corruptos pero existen métodos para calcular que tantas impurezas podríamos generar y así aumentar la precisión de nuestros resultados.

Attribute-based Representations

- Examples described by **attribute values**
 - Boolean, discrete, continuous
 - E.g., situations where I will/won't wait for a table:
- Classification of examples is **positive (T)** or **negative (F)**

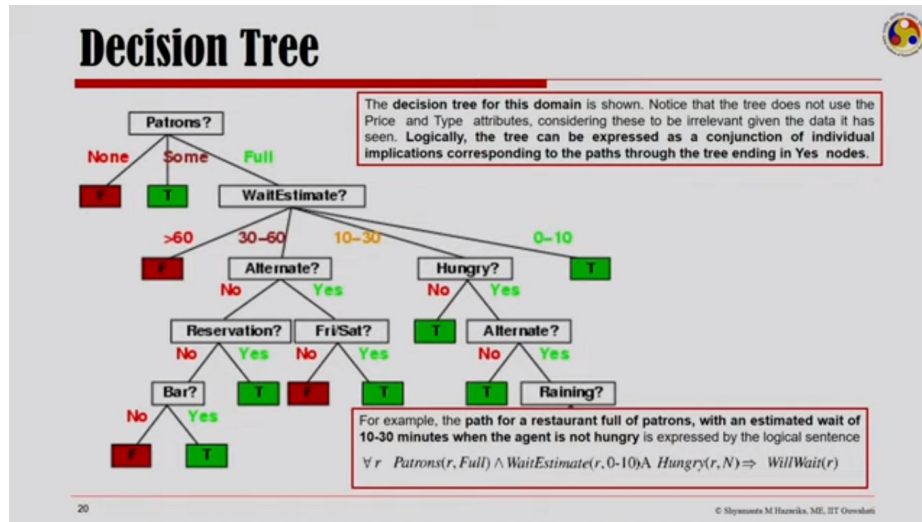
Example	Attributes										Target Wait
	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0-10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30-60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0-10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10-30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0-10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0-10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0-10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10-30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0-10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30-60	T

18

© Shyamanta M Hazarika, MEd, IIT Guwahati

Finalmente proseguiremos a desarrollar nuestro árbol.

Ahora es cuando el algoritmo debe crear los caminos posibles dentro de nuestro ejemplo basándose en las distintas entradas posibles y en el tipo de algoritmo que se esté utilizando. Si nos excedemos en crear caminos podría ser contraproducente, conocido como **overthinking** y en ese caso debemos eliminar algunos caminos, a esto se le conoce como **prunning**. Por ejemplo en el siguiente árbol no se muestra el atributo de precio o de tipo de restaurante ya que no son relevantes para saber si tendremos que esperar o no.



Una vez creado el árbol únicamente se volverá a recorrer las mismas instancias en usos futuros.

Regresión Lineal

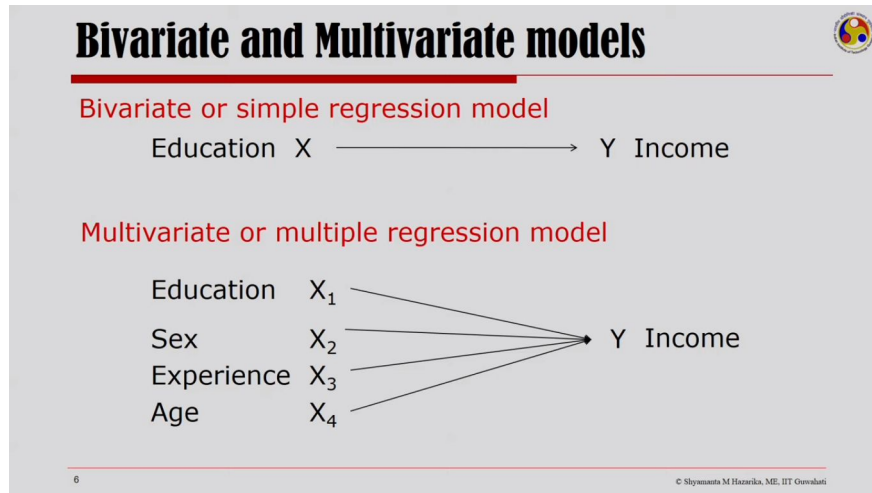
Pertenece al aprendizaje supervisado y su principal función es una forma estadística de relacionar las variables independientes y dependientes para lograr realizar una predicción. En otras palabras son las características que queremos usar para realizar la predicción

Las variables **independientes** son aquellas cuyo valor no cambian por otras variables ajenas, normalmente son denotadas por X.

Las variables **dependientes** son aquellas cuyo valor cambia en función de manipular las variables independientes, de ahí que se les conozcan como dependientes.

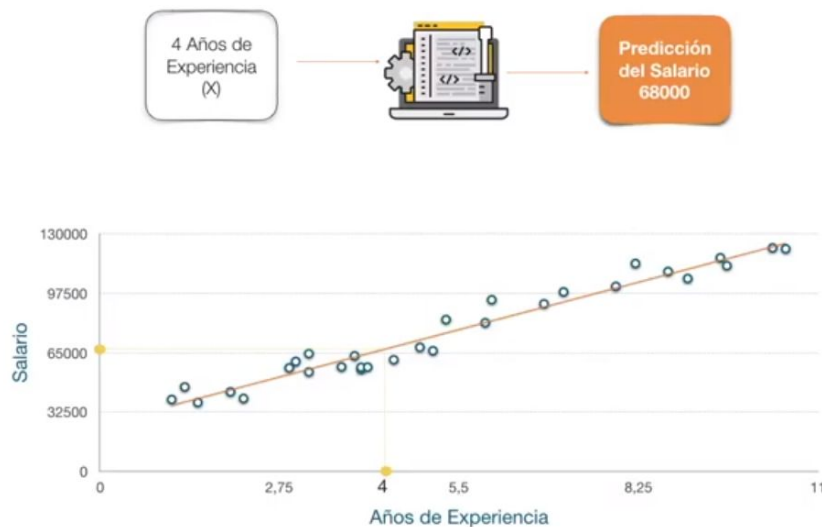
En otras palabras, son las características que estamos intentando predecir.

Tenemos dos tipos diferentes de regresión lineal, simple y múltiple. La múltiple tiene las mismas características que la simple con la única diferencia que se usarán distintas variables independientes para la predicción



Veamos un ejemplo sencillo.

Aprendizaje Supervisado: Lineal Regression

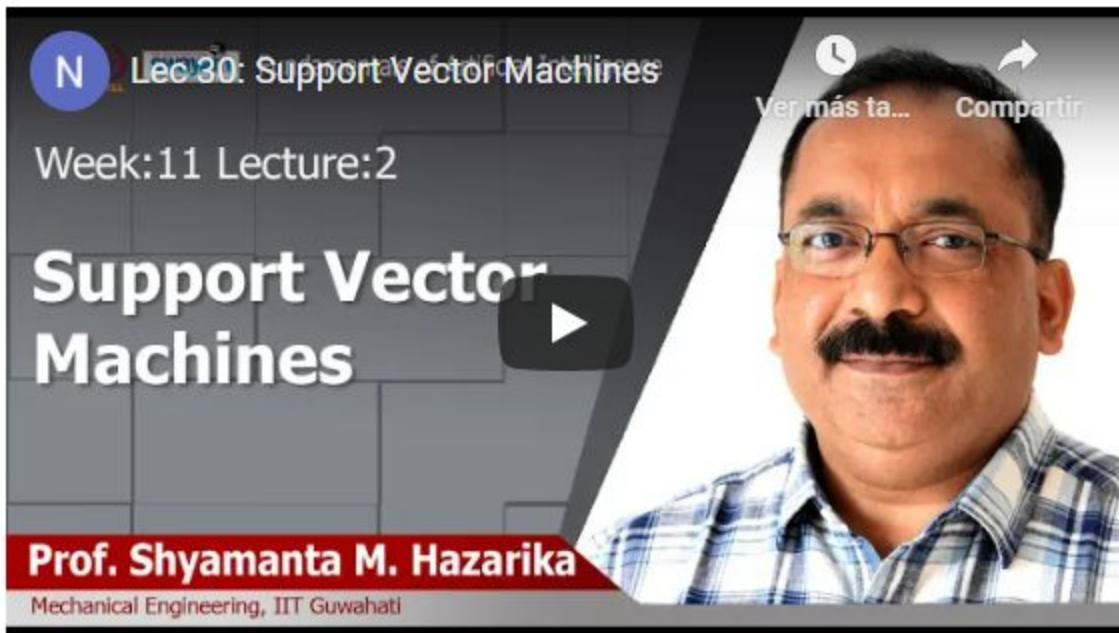


Primero se determinó las variables, siendo los datos que queremos usar para realizar la predicción los años de experiencia, así que estos se convertirán en nuestra variable **independiente** y la usaremos en el eje X. Lo que queremos predecir es el salario que podríamos obtener y este cambia en función de los años de experiencia así que el salario será nuestra variable **dependiente** situada en el eje Y.

Después debemos vaciar nuestra dataset dentro del diagrama, los cuales son representados por los puntos blancos con azul, finalmente en base a un algoritmo que relacione nuestras variables y cree el modelo más preciso posible, este modelo se ve representado por la **línea recta** en la gráfica.

Ahora solo queda usar el modelo generado para nuestras futuras predicciones, el ejemplo nos pide predecir el salario que se obtiene con 4 años de experiencia, para saber esto simplemente ubicamos en el eje Y la posición que represente los 4 años, simbolizado con un punto amarillo en el ejemplo, y trazamos una línea recta hasta la diagonal generada y la posición que tenga con respecto al eje Y será el **salario predicho**.

Máquinas de Vector Soporte (Support Vector Machines, SVMs)



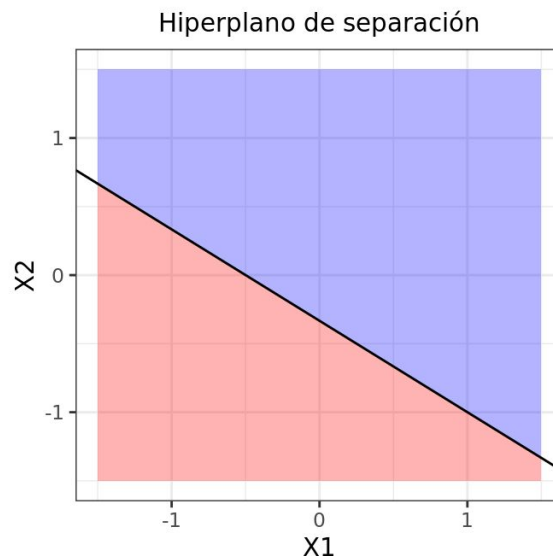
El método de clasificación-regresión Máquinas de Vector Soporte (Vector Support Machines, SVMs) fue desarrollado en la década de los 90, dentro del campo de la ciencia computacional. Si bien originariamente se desarrolló como un método de clasificación binaria, su aplicación se ha extendido a problemas de clasificación múltiple y regresión. SVMs ha resultado ser uno de los mejores clasificadores para un amplio abanico de situaciones, por lo que se considera uno de los referentes dentro del ámbito de aprendizaje estadístico y machine learning.

Las Máquinas de Vector Soporte se fundamentan en el Maximal Margin Classifier, que a su vez, se basa en el concepto de hiperplano. A lo largo de este ensayo se introducen por orden cada uno de estos conceptos. Comprender los fundamentos de las SVMs requiere de conocimientos sólidos en álgebra lineal. En este ensayo no se profundiza en el aspecto matemático, pero puede encontrarse una descripción detallada en el libro Support Vector Machines Succinctly by Alexandre Kowalczyk

En R, las librerías e1071 y LiblineaR contienen los algoritmos necesarios para obtener modelos de clasificación simple, múltiple y regresión, basados en Support Vector Machines.

Hiperplano y Maximal Margin Classifier

En un espacio p -dimensional, un hiperplano se define como un subespacio plano y afín de dimensiones $p-1$. El término afín significa que el subespacio no tiene por qué pasar por el origen. En un espacio de dos dimensiones, el hiperplano es un subespacio de 1 dimensión, es decir, una recta. En un espacio tridimensional, un hiperplano es un subespacio de dos dimensiones, un plano convencional. Para dimensiones $p > 3$ no es intuitivo visualizar un hiperplano, pero el concepto de subespacio con $p-1$ dimensiones se mantiene.



Clasificación binaria empleando un hiperplano

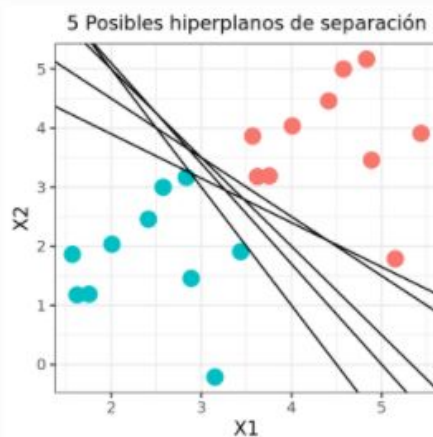
Cuando se dispone de n observaciones, cada una con p predictores y cuya variable respuesta tiene dos niveles (de aquí en adelante identificados como +1 y -1), se pueden emplear hiperplanos para construir un clasificador que permita predecir a qué grupo pertenece una observación en función de sus predictores. Este mismo problema puede abordarse también con otros métodos (regresión logística, LDA, árboles de clasificación...) cada uno con ventajas y desventajas.

Para facilitar la comprensión, las siguientes explicaciones se basan en un espacio de dos dimensiones, donde un hiperplano es una recta. Sin embargo, los mismos conceptos son aplicables a dimensiones superiores.

```
set.seed(66)
X1 <- rnorm(n = 10, mean = 2, sd = 1)
X2 <- rnorm(n = 10, mean = 2, sd = 1)

observaciones <- data.frame(X1 = c(X1, X1 + 2), X2 = c(X2, X2 + 2) ,
                           clase = rep(c(1, -1), each = 10))
observaciones$clase <- as.factor(observaciones$clase)

ggplot() +
  geom_point(data = observaciones, aes(x = X1, y = X2, color = clase), size = 4) +
  geom_abline(intercept = 9, slope = -2) +
  geom_abline(intercept = 8.5, slope = -1.7) +
  geom_abline(intercept = 8, slope = -1.5) +
  geom_abline(intercept = 6.5, slope = -1) +
  geom_abline(intercept = 5.4, slope = -0.75) +
  theme_bw() +
  labs(title = "5 Posibles hiperplanos de separación") +
  theme( legend.position = "none",
        plot.title = element_text(hjust = 0.5, size = 11))
```



Support Vector Classifier o Soft Margin SVM

El Maximal Margin Classifier descrito en la sección anterior tiene poca aplicación práctica, ya que rara vez se encuentran casos en los que las clases sean perfecta y linealmente separables. De hecho, incluso cumpliéndose estas condiciones ideales, en las que exista un hiperplano capaz de separar perfectamente las observaciones en dos clases, esta aproximación sigue presentando dos inconvenientes:

Dado que el hiperplano tiene que separar perfectamente las observaciones, es muy sensible a variaciones en los datos. Incluir una nueva observación puede suponer cambios muy grandes en el hiperplano de separación (poca robustez).

Que el maximal margin hyperplane se ajuste perfectamente a las observaciones de entrenamiento para separarlas todas correctamente suele conllevar problemas de overfitting.

Por estas razones, es preferible crear un clasificador basado en un hiperplano que, aunque no separe perfectamente las dos clases, sea más robusto y tenga mayor capacidad predictiva al aplicarlo a nuevas observaciones (menos problemas de overfitting). Esto es exactamente lo que consiguen los clasificadores de vector soporte, también conocidos como soft margin classifiers o Support Vector Classifiers. Para lograrlo, en lugar de buscar el margen de clasificación más ancho posible que consigue que las observaciones estén en el lado correcto del margen; se permite que ciertas observaciones estén en el lado incorrecto del margen o incluso del hiperplano.

La siguiente imagen muestra un clasificador de vector soporte ajustado a un pequeño set de observaciones. La línea continua representa el hiperplano y las líneas discontinuas el margen a cada lado. Las observaciones 2, 3, 4, 5, 6, 7 y 10 se encuentran en el lado correcto del margen (también del hiperplano) por lo que están bien clasificadas. Las observaciones 1 y 8, a pesar de que se encuentran dentro del margen, están en el lado correcto del hiperplano, por lo que también están bien clasificadas. Las observaciones 11 y 12, se encuentran en el lado erróneo del hiperplano, su clasificación es incorrecta. Todas aquellas observaciones que, estando dentro o fuera del margen, se encuentren en el lado incorrecto del hiperplano, se corresponden con observaciones de entrenamiento mal clasificadas.

Imagen clasificador vector soporte obtenida del libro ISLR

La identificación del hiperplano de un clasificador de vector soporte, que clasifique correctamente la mayoría de las observaciones a excepción de unas pocas, es un problema de optimización convexa. Si bien la demostración matemática queda fuera del objetivo de esta introducción, es importante mencionar que el proceso incluye un hiper parámetro de tuning C . C controla el número y severidad de las violaciones del margen (y del hiperplano) que se toleran en el proceso de ajuste. Si $C=\infty$, no se permite ninguna violación del margen y por lo tanto, el resultado es equivalente al Maximal Margin Classifier (teniendo en cuenta que esta solución solo es posible si

las clases son perfectamente separables). Cuando más se aproxima C a cero, menos se penalizan los errores y más observaciones pueden estar en el lado incorrecto del margen o incluso del hiperplano. C es a fin de cuentas el hiper parámetro encargado de controlar el balance entre bias y varianza del modelo. En la práctica, su valor óptimo se identifica mediante cross-validation.

El proceso de optimización tiene la peculiaridad de que solo las observaciones que se encuentran justo en el margen o que lo violan influyen sobre el hiperplano. A estas observaciones se les conoce como vectores soporte y son las que definen el clasificador obtenido. Esta es la razón por la que el parámetro C controla el balance entre bias y varianza. Cuando el valor de C es pequeño, el margen es más ancho, y más observaciones violan el margen, convirtiéndose en vectores soporte. El hiperplano está, por lo tanto, sustentado por más observaciones, lo que aumenta el bias pero reduce la varianza. Cuanto mayor es el valor de C , menor el margen, menos observaciones serán vectores soporte y el clasificador resultante tendrá menor bias pero mayor varianza.

Otra propiedad importante que deriva de que el hiperplano dependa únicamente de una pequeña proporción de observaciones (vectores soporte), es su robustez frente a observaciones muy alejadas del hiperplano. Esto hace al método de clasificación vector soporte distinto a otros métodos tales como Linear Discriminant Analysis (LDA), donde la regla de clasificación depende de la media de todas las observaciones.

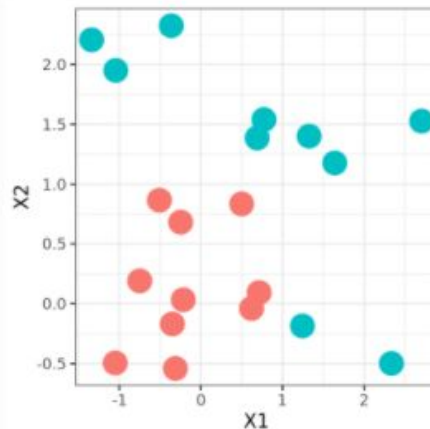
Nota: en el libro Introduction to Statistical Learning se emplea un término C que equivale a la inversa del descrito en este documento.

Ejemplo

Para mostrar el uso de un Support Vector Classifier como clasificador binario, se simulan observaciones en un espacio bidimensional que pertenecen a dos clases. Este ejemplo se ha obtenido de los videos asociados al libro Introduction to Statistical Learning, que no es igual al presentado en el libro.

En los siguientes ejemplos, se emplea la función `svm()` contenida en el paquete `e1071`. Esta función ajusta Support Vector Classifier si se le especifica el argumento `kernel="lineal"` (como se describe más adelante, el método de Support Vector Machines es equivalente al Support Vector Classifier cuando el kernel utilizado es lineal). El argumento `cost` determina la penalización aplicada por violar el margen, es el nombre que emplea esta función para el hiperparámetro C .

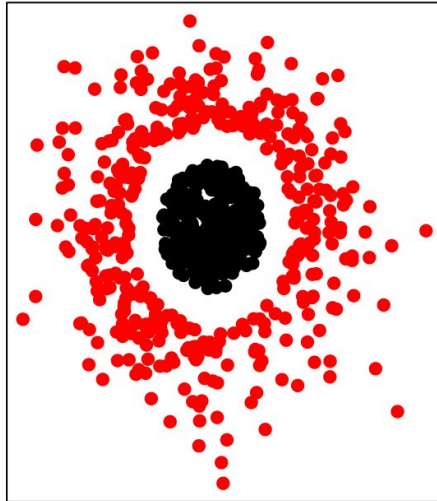
```
set.seed(10111)
coordenadas <- matrix(rnorm(40), 20, 2)
colnames(coordenadas) <- c("X1", "X2")
y <- c(rep(-1,10), rep(1,10))
coordenadas[y == 1, ] <- coordenadas[y == 1, ] + 1
datos <- data.frame(coordenadas, y)
ggplot(data = datos, aes(x = X1, y = X2, color = as.factor(y))) +
  geom_point(size = 6) +
  theme_bw() +
  theme(legend.position = "none")
```



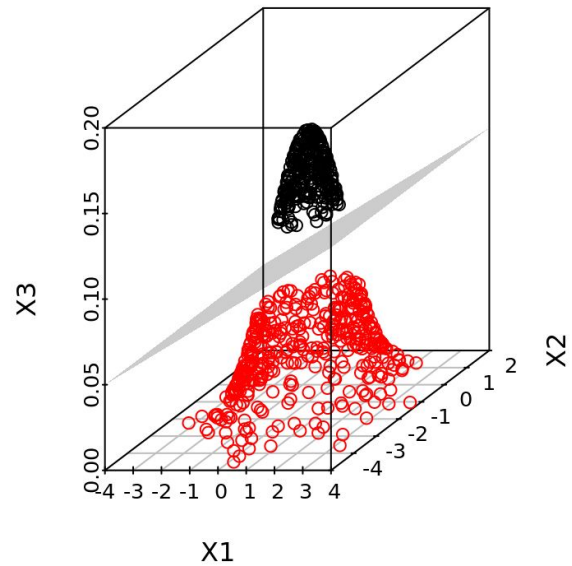
Máquinas de Vector Soporte

El Support Vector Classifier descrito en los apartados anteriores consigue buenos resultados cuando el límite de separación entre clases es aproximadamente lineal. Si no lo es, su capacidad decae drásticamente. Una estrategia para enfrentarse a escenarios en los que la separación de los grupos es de tipo no lineal consiste en expandir las dimensiones del espacio original.

El hecho de que los grupos no sean linealmente separables en el espacio original no significa que no lo sean en un espacio de mayores dimensiones. Las imágenes siguientes muestran cómo dos grupos, cuya separación en dos dimensiones no es lineal, sí lo es al añadir una tercera dimensión.



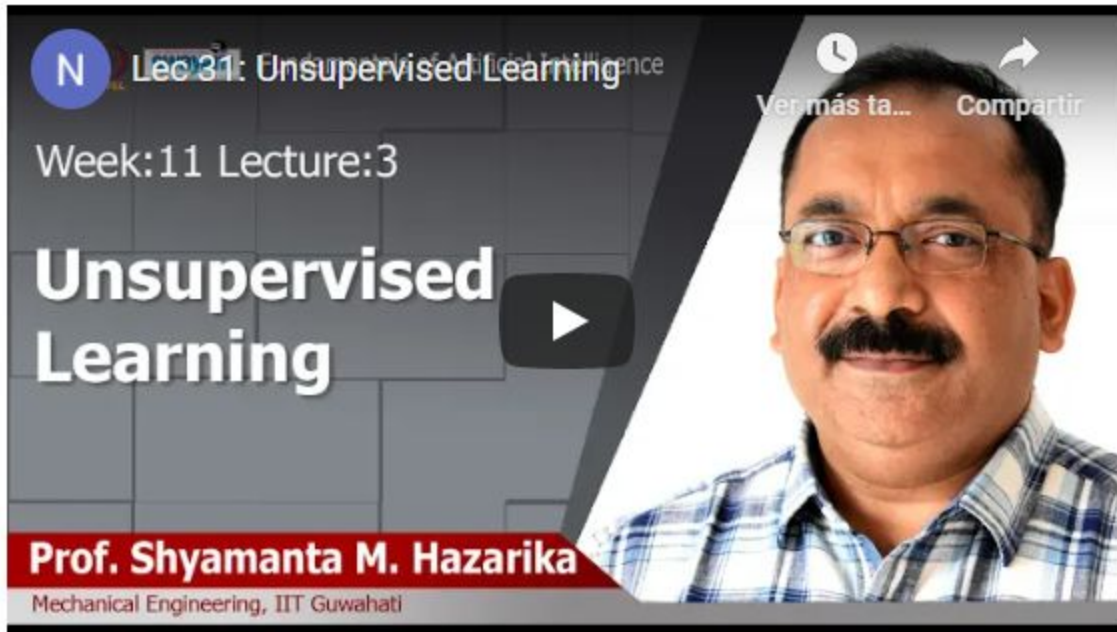
X1



REFERENCIAS

https://www.cienciadedatos.net/documentos/34_maquinas_de_vector_soporte_support_vector_machines

Unsupervised Learning



Aprendizaje sin supervisión

En algunos problemas de reconocimiento de patrones, los datos de entrenamiento consisten en un conjunto de vectores de entrada x sin ningún valor objetivo correspondiente. El objetivo en estos problemas de aprendizaje sin supervisión puede ser descubrir grupos de ejemplos similares dentro de los datos, donde se denomina agrupamiento, o determinar cómo se distribuyen los datos en el espacio, conocido como estimación de densidad. Para decirlo en términos más simples, para un espacio de n muestras x_1 a x_n , no se proporcionan etiquetas de clase verdaderas para cada muestra, por lo que se conoce como aprendizaje sin maestro.

Problemas con el aprendizaje no supervisado:

El aprendizaje no supervisado es más difícil en comparación con las tareas de aprendizaje supervisado.

¿Cómo sabemos si los resultados son significativos dado que no hay etiquetas de respuesta disponibles?

Deje que el experto mire los resultados (evaluación externa)

Definir una función objetiva sobre agrupación (evaluación interna)

¿Por qué se necesita el aprendizaje no supervisado a pesar de estos problemas?

Anotar grandes conjuntos de datos es muy costoso y, por lo tanto, solo podemos etiquetar algunos ejemplos manualmente. Ejemplo: reconocimiento de voz

Puede haber casos en los que no sepamos en cuántas / en qué clases se dividen los datos.

Ejemplo: minería de datos

Es posible que deseemos utilizar la agrupación en clústeres para obtener una idea de la estructura de los datos antes de diseñar un clasificador.

El aprendizaje no supervisado se puede clasificar en dos categorías:

Aprendizaje paramétrico no supervisado

En este caso, asumimos una distribución paramétrica de datos. Supone que los datos de muestra provienen de una población que sigue una distribución de probabilidad basada en un conjunto fijo de parámetros. Teóricamente, en una familia normal de distribuciones, todos los miembros tienen la misma forma y están parametrizados por media y desviación estándar. Eso significa que si conoce la media y la desviación estándar, y que la distribución es normal, conoce la probabilidad de cualquier observación futura. El aprendizaje paramétrico no supervisado implica la construcción de modelos de mezcla gaussianos y el uso del algoritmo de maximización de expectativas para predecir la clase de la muestra en cuestión. Este caso es mucho más difícil que el aprendizaje supervisado estándar porque no hay etiquetas de respuesta disponibles y, por lo tanto, no hay una medida correcta de precisión disponible para verificar el resultado.

Aprendizaje

no supervisado no paramétrico En la versión no paramétrica del aprendizaje no supervisado, los datos se agrupan en grupos, donde cada grupo (con suerte) dice algo sobre las categorías y clases presentes en los datos. Este método se usa comúnmente para modelar y analizar datos con tamaños de muestra pequeños. A diferencia de los modelos paramétricos, los modelos no paramétricos no requieren que el modelador haga suposiciones sobre la distribución de la población y, por lo tanto, a veces se denominan métodos sin distribución.

¿Qué es la agrupación en clústeres?

El agrupamiento puede considerarse el problema de aprendizaje no supervisado más importante ; por tanto, como cualquier otro problema de este tipo, se trata de encontrar una estructura en una colección de datos sin etiquetar. Una definición vaga de agrupamiento podría ser “el proceso de organizar objetos en grupos cuyos miembros son similares de alguna manera”. Un clúster , por tanto, es una colección de objetos que son “similares” entre ellos y son “diferente” a los objetos que pertenecen a otros grupos.

