uc3m | Universidad Carlos III de Madrid

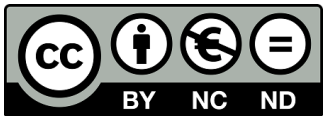
Grado en Ingeniería Informática

2019-2020

Apuntes

Inteligencia Artificial

Jorge Rodríguez Fraile[1]



[1]Universidad: 100405951@alumnos.uc3m.es | Personal: jrf1616@gmail.com

ÍNDICE GENERAL

1. TEMA 2: SISTEMAS DE PRODUCCIÓN.

La aplicación de reglas genera conocimiento nuevo, que lleva a otras reglas.

Componentes:

- **Base de Hechos:** Hechos que representan el problema.
  - Son afirmaciones atómicas, referidas a términos constantes.
  - Si algo no está definido en la BH es falso, Mundo cerrado.
  - Representan hechos simples, objetos, lógica, etc.
  - Se pueden añadir, eliminar y modificar hechos.
- **Base de Reglas:** Reglas que representan el conocimiento para resolver el problema. SI condiciones ENTONCES acciones
  - **Condiciones**:
    - Solo existe la conjunción(y), si se quiere hacer un 'o' se hace una regla para cada uno. Si se quiere hacer un if, se hacen 2 reglas, una de if y otra de else.
    - Las condiciones son sobre hechos de la BH.
    - Contienen variables y constantes definidas en los hechos.
  - **Acciones**:
    - Modifican la BH, añadiendo, modificando o borrando.
    - Operaciones de E/S.
- **Motor de Inferencia:** Responsable de la ejecución de reglas.
  - **Tipos de inferencia**:
    - **Encadenamiento hacia delante(las que podemos completar ya):** Se van mirando las reglas y se comprueba de cuáles de ellas tenemos los antecedentes, y se ejecuta para ver si su resultado nos permite alcanzar a ejecutar una regla nueva. Se utiliza cuando hay pocos datos iniciales y muchas posibles conclusiones. Desventaja: No se centra en las metas. Y hay más comparaciones.

    - **Encadenamiento hacia atrás(queremos completar y buscamos los hechos):** Se fija una submeta, que es obtener un determinado resultado de una regla. Observando sus antecedentes vemos que hechos tenemos, los que nos faltan se convierten en otra submeta, seguiremos este proceso hasta que encontremos una regla de la que tenemos todo y podamos alcanzar a obtener el resultado de la primera submeta, volviendo hacia atrás. Se utiliza cuando hay muchos datos, pero pocos relevantes. Desventaja: Hay que gestionar todas la metas y submetas, y no conocemos el proceso correcto hasta el final.

- **Fases**:
    - **Reducción (opcional)**
    - **Equiparación:** Observar las reglas de las que se cumple el antecedente, con la BH actual, y estas reglas se añaden al Conjunto Conflicto. Puede una misma regla estar varias veces en el CC, si contiene distintos valores. CC={R1, R2(Mario), R2(José)}
    - **Resolución del conjunto conflicto:** Consiste en elegir la estrategia de ejecución de las reglas del Conjunto Conflicto. Para evitar bucles, una instancia se ejecuta una sola vez, se aplica el Principio de refracción, excepto que haya salido y vuelto a entrar.
        - Estrategia por **profundidad**(la más nueva, LIFO): Las últimas en entrar al conjunto son las primeras en ejecutarse.
        - Estrategia por amplitud(la más antigua, FIFO): Las primeras en entrar son las primeras en ser ejecutadas.
    - **Ejecución:** Se ejecutan las reglas del Conjunto Conflicto según la estrategia de resolución, y los resultados se van añadiendo a la Base de Hechos.
        - Añadir: Assert o + Eliminar**:** Retract o - Modificar: Modify
- El ciclo se repite tantas veces como para vaciar el Conjunto Conflicto o hasta una acción de parar.

Herramientas:

Prolog (resultado:-condiciones) y CLIPS (Mirar estructura en diapositivas).

Resumen representación en el espacio de estados:

- Identificar el estado inicial y los posibles estados.
- Identificar los operadores para pasar de un estado a otro.
- Ver conexiones entre los estados.
- Identificar los estados finales, para saber cuando parar.
- Identificar el número de acciones ejecutadas, el coste.
- Identificar las posibles soluciones desde un estado inicial a un estado final, que resuelva el problema.
- Puede haber prioridad entre las reglas.
- Hay que tratar de hacer los operadores lo más genéricos posible para que sean flexibles.
- Describir solo las que sean necesarias, pero no comprimidas, pueden estar desglosadas.

2. TEMA 3: BÚSQUEDA

Son aquello problemas en los que partido de un estado inicial, por el que comenzaremos, y debemos llegar mediante una seria de operaciones a un determinado estado, el estado final. Es importante antes de comenzar un problema de búsqueda conocer cuáles son los estados posibles, y entre ellos debe estar el estado inicial y el final. La transición entre estos estados se realiza mediante una serie de operaciones. Se pueden representar como un grafo, en el que la raíz es el estado inicial y de el cuelgan todos los estados resultado de aplicar las operaciones. El número de estados posibles aumenta exponencialmente cuando varía el número de datos.

Lista abierta: Lista de todos los nodos que aún no hemos explorado.

Parámetros importantes:

- **Factor de ramificación, b:** Numero de posibles nodos que pueden derivar de otro, posibles sucesores de un nodo. El factor de ramificación medio es la suma de todos los factores de todos lo estados divididos del número de estados. Ojo tener en cuenta retroceso en el árbol.
- **Profundidad del árbol de búsqueda, d**: Cuantos operadores tenemos que aplicar para llegar a la solución, solo se puede calcular tras encontrarla, pero no antes.

Búsqueda no informada: No tenemos información de donde esta la solución, solo sabemos cuál es, por lo que tenemos que seguir un proceso mecánico lógico que recorra todos los estados. No tienen pesos las ramas, por lo que elegir una u otra es igual si llega a la solución.

- **Búsqueda en amplitud**: Sigue el orden de un FIFO, el primero que entra es el primero en salir se ejecutan en orden de entrada, una cola. Si un estado vuelve a aparecer no lo ponemos, porque ya esta contemplado. Este método llega al estado final por medio de menos operadores y eso es mejor, es la solución óptima. Como no hay pesos en las ramas cuando aparece la solución en una rama hemos acabado.
    - Completitud(hay solución y el factor de ramificación es finito en cada nodo), admisibilidad (si todos los nodos tienen el mismo coste, encuentra la solución óptima), eficiencia y consumo de memoria exponencial.

- **Búsqueda en profundidad:** Sigue el orden de un LIFO. El último en entrar será el primero en ejecutar, una pila. Puede alcanzar la solución en menor tiempo, pero posiblemente lo hará mediante más operaciones. Se puede poner un límite de profundidad de las ramas, para que no se llegue a la solución con tantas operaciones. No se ponen estados que ya hayan aparecido. Recorre una rama hasta que no puede avanzar más, entonces retrocede hasta que pueda cansar por una rama.

    - Requiere backtracking(cuando retrocede al llegar al límite de profundidad, no encuentra soluciones en esa rama o solo aparecen duplicados), completitud (no asegura encontrar la solución, aunque la haya), admisibilidad (no asegura que sea la solución óptima) y eficiencia(cuando la meta está lejos del estado inicial o cuando hay problemas de memoria).

- **Complejidad de amplitud y profundidad:**

    - Amplitud: Temporal y espacial exponencial $O(b^d)$
    - Profundidad: Temporal exponencial $O(b^d)$ y espacial lineal O(b).

- **Búsqueda de coste no uniforme**: Cuando aparecen costes en las transiciones. Se buscará el camino a la solución más barato en costes.

    - **Dijkstra**: Amplitud pero ordenado ascendente. Consiste en ir escogiendo aquellas ramas que menor coste tengas, se ordena la lista abierta de menor a mayor y FIFO, pero si hay un estado repetido hay que conspirar su coste para ver si es más barato. Cuando vamos avanzamos hay que ir arrastrando el coste acumulado de las ramas ya recorridas. Es como recorrer en profundidad pero escogiendo el más barato. Termina cuando hemos comprobado que la solución encontrada es la más óptima.
    - **Ramificación y acotación:** En profundidad, seguimos una rama cogiendo el que menos coste hasta una meta, y termina cuando no encontramos otra rama con menor coste que la escogida, ponemos el coste de esa como límite de coste de la rama, ya que si lo supera será mejor la anterior, pero si la encuentra esa es la solución.

- **Pasos a seguir**:
  - Formalizar el problema:
    - Definir los posibles estados, además cuál será el inicial y el final o meta.
    - Operaciones que se pueden realizar en los estados.
  - Estimar la complejidad, el número de estados diferentes del problema.
  - Algoritmo de búsqueda:
    - Orden al generar nodos.
    - Lista abierta, nodos generados y nodos expandidos.
    - Conocer las propiedades del algoritmo: Completitud, admisibilidad, complejidad temporal y espacial.

Búsqueda heurística: Se tiene conocimiento parcial sobre un problema que nos permite resolverlo de una manera más o menos eficiente. Si se tiene el conocimiento perfecto, se puede desarrollar un algoritmo exacto, pero si no se tiene conocimiento se hará una búsqueda no informada.

- Hay que hallar una función heurística h(n), que permita hallar el coste estimado desde el nodo n hasta un nodo objetivo, y esta devuelve un valor real positivo. Se descubre resolviendo modelos simplificados del problema real, simplificando mecánicas como restricciones, pero no el propio problema. Es el coste óptimo del problema relajado. Ejem: Distancias de Manhattan, para problemas con un tablero con coordenadas, $h(n) = |x - x_i| + |y - y_i|$, la suma de distancias.
- **Búsqueda Escalada (Hill Climbing)**: Básicamente buscando el camino que genera menores h, solo fijándonos en la h y sin acumular, hasta que no se pueda encontrar uno más bajo que el anterior, o encuentre el final. Consiste en coger un nodo de la lista abierta y obtener sus sucesores, calculamos la función heurística de cada uno, y nos quedamos en la lista abierta solo con aquel que tiene menor coste(no se acumulan valore, es el puro coste heurístico) y ese coste además debe ser menor que el del padre. No hay backtracking, por lo que si ese único nodo que usamos no puede avanzar, hemos terminado. No se analizan los nodos repetidos, ya fueron descartados. En cuanto un nodo sucesor sea final, el proceso termina.
  - No completo (NO, ya que no siempre encuentra la solución), no admisible(NO, si no encuentra siempre la solución no será admisible) y eficiencia (es rápido y útil)
  - Posibles soluciones: Poder hacer backtracking, avanzar por un par de ramas a la vez, reinicio aleatorio o que siga avanzando aunque el sucesor tenga un mayor coste.

- **Búsqueda Mejor Primero**: Básicamente ir calculando para cada sucesor la suma del h propio y el coste en llegar (OJO, pero sin tener en cuenta los h anteriores, es la suma de transiciones pura), e ir avanzando por el que menor suma tiene. Consiste en ir cogiendo de la lista abierta el nodo de menor coste y obtener sus sucesores, de los que calculamos su coste que es la suma de la función heurística de ese nodo y el coste desde el inicio hasta llegar a ese nodo, y ordenamos la lista abierta en orden ascendente en coste, pero no desechamos ningún nodo. Este algoritmo siempre encuentra la solución y es la más óptima. Los nodos recorridos pasan a la lista cerrada, que servirá para ver si un nodo repetido lo hemos encontrado con menor coste, por lo que se convertiría en mejor camino. Si el nodo final es escogido termina, pero no solo si aparece como un sucesor. $f(n) = g(n) + h(n)$ El coste total es la suma del coste desde el inicio al nodo n y el coste desde n hasta el nodo final.

    - Al final se pueden calcular los valores reales, que se indican con un *.
    - Completitud (si existe solución la encuentra), admisibilidad (encuentra la solución óptima, si: sucesores finitos, los costes son mayores que 1 y $h(n)$ es admisible, menor que el $h^*(n)$) y complejidad exponencial.
    - Cuando mayor $h(n)$ mejor informada esta la función heurística y menor numero de nodos sucesores aparecen.

3. TEMA 4: REDES BAYESIANAS

Razonamiento Probabilístico:

Incertidumbre: Confianza que tenemos de un suceso.

- Se expresa como una probabilidad, que es una medida de:
  - Proporción de veces en que algo es cierto.
  - Grado de creencia en que algo es cierto.
- **Variable aleatoria**: Que pueden tomar valores en un dominio, el valor asociado es desconocido, pero podemos saber la probabilidad de cada valor posible.
- **Espacio muestral Ω:** Todos los posibles resultados de un experimento aleatorio.
- **Evento**: Subconjunto del espacio muestral, hay varios.
- **Evento atómico:** Evento de un único elemento y define un único estado.
- **Distribución de probabilidad**: Asigna a cada evento un valor que representa la probabilidad de que ocurra ese evento. Para hallar la probabilidad debemos tener datos de e. Se puede expresar como una vector formado por todas las probabilidades. $P(e)$
  - Toma valores entre 0 y 1, incluidos.
  - La suma de todas las probabilidades del evento es 1.
- **Probabilidad de un evento no atómico:** Suma de las probabilidades de todos los eventos atómicos que lo componen, se suman los posibles.
  - $P(A) = \sum P(e)$
- **Teorema de la probabilidad total**: La suma de las probabilidades que tiene de que ocurra A condicionando con otro evento, por la probabilidad de ese evento. $P(A) = \sum P(A, B1, B2, ...)$
- **Regla del producto:** $P(A/B) = P(A, B)/P(B) = P(A \cap B)/P(B)$

- **Probabilidad a posteriori(condicional)**: Sobre una observación, tenemos evidencias. Esa evidencia modifica el conocimiento sobre el dominio.
  - $P(A|B) = P(A \cap B)/P(B) = \alpha P(B \cap A)$
  - **Regla de la cadena:** $P(A \cap B) = P(A|B)P(B)$
  - **Comparar probabilidades condicionadas:** Cuando la B es la misma, pero la A cambia, la suma de ambas probabilidades debe dar 1, por lo que dejamos como incógnita $\boldsymbol{\alpha = 1/P(B)}$ que es la **constante de normalización**. El que mayor x tenga es más probable siendo x el término que acompaña a α.
  - α permite también hallar $P(B)$ cuando no se nos da y conocemos las condicionadas.
- **Probabilidad a priori:** Probabilidad que algo ocurra cuando no tenemos información.
- **Probabilidad conjunta:** Probabilidad de un conjunto de variables.
  - Se puede representar de manera tabular, en una tabla, pero en problemas reales no es viable, ya que hay ciento o miles de variables. Hay que tener en cuenta el coste y tiempo de respuesta.
- **Teorema de Bayes:** Se pueden calcular unas condicionales a través de otras.
  - $P(A/B) = (P(B/A)P(A))/P(B) = \alpha P(B/A)P(A)$
- **Independencia**: A y B lo son si y solo si la ocurrencia de uno de ellos no afecta a la ocurrencia del otro.
  - $\boldsymbol{P(A|B) = P(A)}$ La probabilidad de A no se ve afectada por B.
  - $\boldsymbol{P(A, B) = P(A)P(B)}$
  - **Reducción de tamaño de distribución:** Eso implica algoritmo más eficiente y menos datos a especificar.
- **Inferencia**: Calcular la probabilidad de eventos (probabilidad a posteriori) dada cierta evidencia. Se usa para:
  - **Predicción.**
  - **Diagnosis.**
  - **Clasificación.**
  - **Toma de decisiones:** Elegir acciones más útiles.

Redes Bayesianas:

- **Independencia condicional:** Cuando hay relaciones entre varias variables y están relacionadas, pero por medio de otra, que no es causa directa. Hay un intermediario, por lo que conociéndolo depende del intermediario, no del que está alejado un nodo.

$P(X, Y|Z) = P(Y|Z)P(X|Z)$ y también $P(X|Y, Z) = P(X|Z)$

Nos permite reducir el número de parámetros, normalmente de exp. a lineal.

En las redes bayesianas se asume que hay variables ind. condicionalmente.

Red Bayesianas: Es un Grafo Acíclico Dirigido (DAG), en la que los nodos son las variables, los arcos indican causa directa. Cada nodo tiene una tabla de probabilidad condicional (CPT), indica la influencia de los nodos padre sobre el. Cuando no tiene es la probabilidad a priori.

- Se puede expresar de forma compacta como la probabilidad de todos separados por comas, distribución de probabilidad conjunta, gracias a que se asume independencia condicional y después podemos factorizarla.
- Es ir sacando términos, así:
  - $P(A, B, C) = P(A|B, C)P(B, C) = ...$

$$P(X_1, ..., X_n) = \prod_{i=1}^{n} P(X_i/Padres(X_i))$$

- La CPT tendrá 2^a filas en un nodo con a padres. Se pone la probabilidad de que sea true, el propio nodo, ya que de que sea false es la inversa 1-P()
- Complejidad $O(n \cdot 2^b)$ n variables y b padres.

Causalidad vs. Correlación: Que numéricamente estén relacionados, no quiere decir que existan una relación directa, pueden existir variables intermedias. Fijarse si esto ocurre.

- Correlación no implica causalidad, pero Causalidad implica correlación.

Inferencia: Calcular la probabilidad a posteriori de una variable dada cierta evidencia.

- Hay que **tener en cuenta**: La variable pregunta, las variables evidencia y las ocultas.

- **Inferencia exacta por enumeración**:
  - 1. Se aplica la regla del producto. Donde $\alpha = \frac{1}{lo\ de\ la\ derecha}$ o $\frac{1}{\sum_s \alpha P_s}$
    - $P(A|B) = \alpha P(A, B)$**.**
  - 2. Después aplicamos la regla de la probabilidad total, OJO con los sumatorios y poder sacar variables para simplificar o que sume 1. En general las variables que no son ancestros de variables pregunta o variables evidencia son irrelevantes.
    - $\alpha P(A, B) = \alpha \sum P(A, B, C)$**.**
  - 3. Lo de dentro del sumatorio se factoriza y operamos.
    - Las evidencias tendrá el valor fijo, pero las ocultas se contemplan con el sumatorio. Añade en forma de sumatorio las variables ocultas. Cuando no se indica si la variable pregunta es true o false, se hallan ambas posibilidades por separado, esto a su vez nos permite hallar α, suman 1 ambas prob.
  - El problema de este método son la variable ocultas que tienen complejidad exponencial.
  - Es eficiente para poliarboles, que como mucho haya un arco entre cada par de nodos. Esta estructura da lugar a una complejidad lineal.
  - No es eficiente cuando la estructura de la red no es un poliarbol, pasa de ser complejidad lineal a ser exponencial. En estos casos se usa la inferencia aproximada.

- **Inferencia aproximada**: Se hace mediante estimaciones, para ello se hace un muestreo de la red Bayesiana, comenzando desde padres hasta los hijos. Muestrear, quiere decir que siguiendo la distribución de probabilidad sacamos un número aleatorio. Por ejemplo $P(Nublado) = (0{,}5, 0{,}5)$ Sacamos un numero entre 0 y 1, si cae en la parte de la izquierda es true, si no false. $0 - 0{,}5$ true y $0{,}5 - 1$ false.
  - De esta manera sacamos que valor tiene cada variable y podemos centrarnos en ese caso concreto, y hallar su probabilidad como haríamos en el caso anterior.
  - **Muestreo directo:** La probabilidad de un evento se estima como el numero de casos del evento generados por muestreo dividido entre el numero total de casos muéstrales.

$$P(x_1, x_2, ..., x_n) \simeq \frac{\#casos(x_1, x_2, ..., x_n))}{\#totalCasos}$$

$$P(X/e) = \frac{\#casos(X, e))}{\#casos(e)}$$

Clasificador *Naïve Bayes*: Caso especial de red bayesiana con una estructura en la que de un nodo cuelgan el resto. El nodo raíz es del que deseo conocer la probabilidad, y los hijos son los atributos y son independientes entre ellos.

$$p(c_i/a_1...a_n) = \frac{P(a_1...a_n/c_i)p(c_i)}{p(a_1...a_n)} \rightarrow p(c_i/a_1...a_n) = \alpha p(C = c_i) \prod_k P(A_k = a_k/C = c_i)$$

- **Se puede resolver con** Bayes normal o mediante los métodos de inferencia.

Razonamiento Probabilístico en el Tiempo:

- Hasta el momento hemos contemplado solo mundos estaticos, pero ahora consideramos el paso del tiempo. Cuando el mundo es no determinista, que hay varias posibles decisiones en cada paso, se emplea el procesos de decisión de Markov.
- **Modelos de Markov:** Se representa con un grafo la probabilidad de las transiciones, estando en un tiempo t y yendo a t+1, el momento siguiente. El estado actual determina la distribución de probabilidad del estado siguiente. El momento inicial será E_0.
  - **Hipótesis de Markov:** Se asume que el estado actual solo depende del anterior, es condicionalmente independiente de los que no son el inmediato anterior.

    $$P(E_{t+1} = s1/E_t = s2)$$

  - Se puede representar como una red bayesiana, en la que los estado que no dependen de otros tienen su probabilidad a priori y las que dependen de otro su tabla de probabilidad condicionada.
  - **Calcular la probabilidad:** Se hace como inferencia exacta. Lo primero es hacer la regla del producto, después hacer la probabilidad total con sumandos y finalmente factorizar. Pero esta opción es muy pesada ya que hay que considerar todos los posibles caminos anteriores.

    $$P(E_t = s_j) = \sum_{\textit{Todos los caminos que acaban en el estado} P(E_0,E_1,...,E_t=s_j)}$$

  - **Programación dinámica:** Es una técnica/truco, que consiste en hacer una definición recursiva en función del instante anterior. Se comienza desde el final, y se van llamando unas a otras hasta el inicial. Algoritmo de Simulación hacia delante, que tiene complejidad lineal para un tiempo t.

    $$\forall j, P(E_{t+1} = s_j) = \sum_{i=1}^{N} P(E_{t+1} = s_j/E_t = s_i)P(E_t = s_i)$$

- **Distribuciones estacionarias:** Normalmente solo podemos predecir a corto plazo, a medida que nos alejamos de los datos iniciales menos sabemos, llegando en el infinito a ser equiprobable y no nos permite decantarnos por ninguna.
    - La incertidumbre se acumula, hasta que no sabremos cual es el estado.
    - Para la mayoría de las cadenas la distribución al final es independiente de la inicial, la distribución al final es igual si empezamos con $P(X_0 = lluvia) = 1$

- **Modelos de Markov Ocultos (HMMs):** Hay variables observables, evidencias, que determinan el estado. Tiene forma de red bayesiana en el que las observaciones cuelgan del estado y solo dependen del estado actual. Cada estado depende del inmediato anterior.

    - **Se asume:**
        - Es un proceso estacionario.
        - Se cumple la hipotesis de Markov.
        - Las observaciones en t solo dependen del estado en t. P(O/E)
    - **Se define por:**
        - Conjunto de estados y de observaciones.
        - La probabilidad a priori, aquel que no depende del anterior. $P(E_0)$
        - El modelo de transiciones. $P(E_1/E_0)$
        - El modelo de observaciones. $P(O_1/E_1)$
    - La probabilidad de las transiciones se representan en la tabla de probabilidad condicionada.
    - **Inferencia en HMMs:**
        - Se puede resolver por definición recursiva.
        - La forma mas eficiente es por programación dinámica, pero no la veremos.
        - Nosotros las resolveremos mediante inferencia exacta en redes Bayesianas, a pesar de que es menos eficiente.
        - **Se factoriza como:**
          $P(E_0, ..., E_t, O_0, ..., O_t) = P(O_0/E_0)P(E_0) \prod_{t=1}^{T} P(E_t/E_{t-1})P(O_t/E_t)$

- Tareas típicas de Inferencia:
  - **Problema de Evaluación**: Calcular la probabilidad de una secuencia de observaciones.
  - **Problema de Decodificacion**: Dada una secuencia de observaciones, determinar cual es la secuencia de estados correspondiente que explica mejor esas observaciones.
  - **Problema de Filtrado:** Distribución de probabilidad del estado actual dada cierta evidencia histórica, desde t hacia atrás.
  - **Problema de Predicción:** Probabilidad de estados futuros dada evidencia.
- **Procesos de Decisiones de Markov (MDPs)**:
  - Se toma una decision en cada instante. El caso general es que sean acciones no deterministas, hay varias posibilidades en cada caso y cada una de ellas tiene un probabilidad y debemos decidir cuál tomamos.
  - Las **probabilidades de transicion** dependen de las acciones.

$$P(S_{t+1} = s'/S_t = s, A_t = a) = P(s'/s, a)$$

  - Hay un **refuerzo o un coste**, indica como de bueno o malo es esa transición, por cada par de estado-accion:

$$R(S_t = s, A_t = a) = R(s, a)$$

  - Se define como una tupla **<S, A, P, R>:**
    - **S**, estados.
    - **A**, acciones.
    - **P**, probabilidades P(s'/ s, a). Cada estado tiene una probabilidad para cada accion con sus estados contiguos.
    - **R**, refuerzos o costes R(s, a).
  - El objectivo es determinar que accion ejecutar en cada estado para maximizar el refuerzo o minimizar el coste.
  - Al no ser determinista hay **dos opciones:**
    - **Politica** (Acciones estocásticas): Se determinada para cada estado un accion determinada.
      - Es un mapeo completo de estados a acciones, pero no es una secuencia de acciones. Aunque haya un fallo en la ejecucion el agente puede seguir.
      - Maximiza el refuerzo esperado o minimiza el coste esperado en lugar de alcanzar un estado meta.

    - ◇ Para cada MDP existe una politica optima.
    - ◇ Determina que hacer independientemente del efecto de cualquier accion en cualquier instante de tiempo.
  - **Replanificar**: Se vuelve a planificar cuando sale del anterior, de esta manera cada estado tiene en cuenta el anterior.
- **Accion estocástica**: Consigue el efecto deseado con probabilidad p.
- Para el caso de **minimizar coste:**
  - **Politica optima π***: mínimo coste esperado.
    - ◇ **Acciones deterministas**: Si la accion a lleva al estado s', el coste es: **C(a) + costeDesde(s')**
    - ◇ **Acciones no deterministas**: Si la accion a tiene efectos probabilisticos, el coste esperado es:
      C(a) + $\sum$sub(s') P(s'| s, a) * costeDesde(s')
- **Funcion de valor:** coste esperado de un estado.
  - **V(s):** coste esperado de alcanzar la meta desde s.
  - **Busqueda**: Coste del camino optimo desde s. Se puede usar como heuristica, y es la heurística perfecta $h^*(s)$
  - **MDP**: Coste esperado de la estrategia optima para alcanzar la meta desde s. Conociendo V(s) podemos calcular una politica optima π^*.
- **Calcular V(s).**
  - **Ecuaciones de Bellman**. Acciones deterministas (Es programación dinámica, recursividad)
    - ◇ Si s es un estado meta, $V(s) = 0$. Es sumidero. Inicialmente todos están a 0.
    - ◇ En el resto de casos, donde s' es el estado resultado de aplicar a en s:

$$V(s) = \min_{a \in A(s)} [c(a) + V(s')]$$

  - **Ecuaciones de Bellman**. Acciones no deterministas.
    - ◇ Dominios estocasticos, apartir de la accion a: $c(a) + \sum_{s' \in S} P_a(s'/s)V(s')$
    - ◇ Entonces, el estado meta $V(s) = 0$ para costes y en el resto de casos:

$$V(s)_{i+1} = R(s) + \max_{a \in A(s)} [\gamma \sum_{s' \in S} P_a(s'/s)V(s')]$$

    - ◇ **Politica optima**: Consiste en elegir las accione que han generado el mínimo.

$$\pi^*(s) = \arg\max_a [\gamma \sum_{s' \in S} P_a(s'/s)V^*(s')]$$

- ◇ **Resolviendo las ecuaciones de Bellman:**
  Algoritmo de Iteración de Valor:
  El $V(s)$ de los finales es 0 para costes y el refuerzo si hay refuerzos.
  Inicialmente todos se ponen a 0.
  Se hacen tantas rondas como sean necesarias para que los valores de $V(s)$ se estabilicen entre rondas. Y cada ronda calcula $V(s)$ para cada estado.
  Termina cuando alcanza un punto fijo, cuando los valores no cambian en dos iteraciones sucesivas. Se estabiliza.

Lógica Borrosa:

- Los conceptos no son ciertos o falsos de forma clara, a diferencia de la lógica clásica en la que los valores solo son true o false. Se emplean conceptos y modificadores que son difusos.
- **Representación:**
  - Conjunto borroso.
  - **Función de pertenencia:** Indica en que medida un elemento pertenece al conjunto borroso. Valores en el rango [0, 1], en el que 0 representa absolutamente falso y 1 absolutamente verdadero.
- **Lógica borrosa vs. probabilidad:**
  - **Probabilidad**: Lo hechos ocurren o no. Expresan conocimiento parcial.
  - **Lógica borrosa**: Conceptos vagos, inciertos. Expresa grado de verdad parcial.
- **Sistema de reglas borrosas:**
  - Descripción del problema, las reglas que aparecen.
  - Definir los términos borrosos de las reglas, mediante su función de pertenencia.
    - **Variables**: Expresan cualidades.
    - **Valores**: Las variables toman valores de un dominio discreto. Los límites entre los valores son borrosos.
    - **Tipos**: Sigmoide(la S), Gausiana (n), triangular y trapezoidal. Las dos últimas son las que utilizaremos nosotros.
    - **Modificadores**: Operan sobre la función de pertenencia. ^{2}, ^{3}...

- **Combinar términos:**
  - **Conjunción**: Intersección. El menor de ambas para cada x. MIN
  - **Disyunción**: Unión. El máximo en cada x de las funciones. MAX
  - **Negación**: Los valores inversos de cada x. 1-valor
- **Combinar reglas**, para generar una salida única. (Varias son parcialmente cercas)
  - p->q p es cierto en un grado, entonces q también es cierto en un grado.

- **Sistema de reglas borrosas:**
  - Entrada datos nítida.
  - Borrosificar el dato nítido.
  - Base conocimiento, reglas borrosas.
  - Desborrosificar, pasar a un valor nítido.
  - Salida dato.

- **Inferencia con reglas borrosas:** 4 pasos.
  - **Método de Mandami**: es el método típico de inferencia borrosa.
    - **Borrosificar las entradas:**
      - Determinar en que grado la entrada nítida pertenece a los conjuntos borrosos, para cada valor que puede tomar la variable.
    - **Evaluación de reglas:**
      - En que medida las entradas borrosificadas verifican los antecedentes de la regla, se evalúan todas la reglas. AND es intersección, OR es unión y NOT la inversa. Si es un solo valor se coge directamente el grado de la entrada.
      - Se obtiene la **Similitud**, que es el mayor grado que puede obtener el consecuente.
      - **El consecuente**, resultado de cortar la función de pertenencia del consecuente al nivel que marca la similitud del antecedente.
    - **Agregación de los consecuentes:**
      - Unificación de las salidas de todas las reglas tras evaluarlas en un solo conjunto borroso. Nosotros hacemos el MAX para cada x, unión.
    - **Desborrosificar el resultado:**
      - Convertir el resultado en un valor nítido, el más común es centro de gravedad o centro de del área.

4. TEMA 5: APRENDIZAJE AUTOMÁTICO

Acceso en Drive a las diapositivas

5. TEMA 6: TECNICAS BIOINSPIRADAS

Acceso en Drive a las diapositivas

6. TEMA 7: ROBÓTICA

Acceso en Drive a las diapositivas