Apuntes de inteligencia artifici...

cmhernandezdel

Inteligencia Artificial I

3º Grado en Ingeniería Informática

Facultad de Informática
Universidad Complutense de Madrid

Apuntes de inteligencia artificial

2º curso. Grado en Ingeniería Informática.

Autor: Carlos M. Hernández (cmhernandezdel)

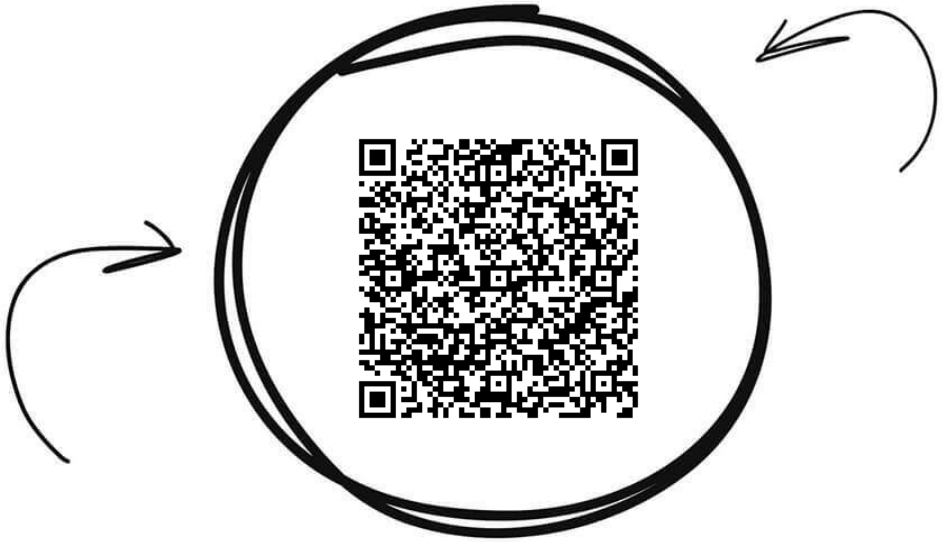
Índice

Tema 1: ..... pág. 3
Tema 2: ..... pág. 4
Tema 3: ..... pág. 6
Tema 4: ..... pág. 8
Tema 5: ..... pág. 10
Tema 6: ..... pág. 11
Tema 7: ..... pág. 13
Tema 8: ..... pág. 14
Tema 9: ..... pág. 17
Tema 10: ..... pág. 19
Tema 11: ..... pág. 21
Tema 12: ..... pág. 22

LA AUTOESCUELA EN TU UNIVERSIDAD

AUTOESCUELA EUROPEA

Inteligencia Artificial I



Banco de apuntes de la

WUOLAH

Comparte estos flyers en tu clase y consigue más dinero y recompensas

1. Imprime esta hoja
2. Recorta por la mitad
3. Coloca en un lugar visible para que tus compis puedan escanar y acceder a apuntes
4. Llévate dinero por cada descarga de los documentos descargados a través de tu QR

Tema 1: Introducción a la IA

Introducción

Alan Turing planteó la pregunta: "¿Pueden pensar las máquinas?". A partir de ahí desarrolló lo que conocemos como el test de Turing, que consistía en meter a una persona en una sala, y si dicha persona no pudiera reconocer si lo que hay en la otra sala es una máquina o una persona, entonces dicha máquina pasaría el test.

Distinguimos entre dos tipos de IA: la IA débil (conseguir que ordenadores hagan tareas para las que se requiere cierta inteligencia) y la IA fuerte (construcción de máquinas inteligentes). Mientras que en programación normal el programador le da unas instrucciones a la máquina, en IA, un experto le da la representación del conocimiento, para que esta IA aprenda por inferencia y decida por búsqueda, teniendo en cuenta siempre el nivel de incertidumbre.

Tema 2: Representación mediante sistemas de producción

Introducción

La idea es que para programar se utilizan unas reglas. El programador le da las reglas y la aplicación de una regla a la situación actual (hechos) genera nuevos hechos.

Una instancia es una especificación de una regla que se cumple. Las instancias se almacenan en la agenda, y con ellas se crean nuevos hechos. Cuando se vacía la agenda, se termina.

Un sistema de producción es un sistema que aplica razonamiento basado en reglas, llamadas reglas de producción. El conocimiento del problema (hechos) está separado del conocimiento de la resolución (reglas). Los hechos pueden cambiar (sistema no monótono). El motor de inferencia es lo que va haciendo que cambie el estado del sistema.

Ejemplo suponiendo que la resolución de conflictos es aleatoria:

	Agenda	¿Ejecuta?	Resultado
C1	A, B -> D, ¬E	Sí	+D, -E (A, B, C, D)
C2	A, D -> F B, D -> E, ¬F	No Sí	+E, -F (A, B, C, D, E)
C3	A, D -> F	Sí	+F (A, B, C, D, E, F)
C4	F, E -> T	Sí	+T (A,B,C,D,E,F,T)

Representación del conocimiento

La base de hechos contiene afirmaciones atómicas, referidas a términos constantes. Los hechos pueden variar, como ya habíamos dicho. Tres operaciones: assert, retract y modify.

Las reglas tienen en su parte izquierda las condiciones y en su parte derecha las acciones.

Ejemplo: Resolver el siguiente sistema de producción con resolución aleatoria de conflictos:

R1: if A(x) and B(y) then C(y)

R2: if A(x) and C(x) then D(x)

Hechos: A(m), B(m), B(j), C(a)

	Hechos	Agenda	¿Ejecuta?	Resultado
C1	A(m) B(m) B(j) C(a)	A(m), B(j) -> C(j) A(m), B(m) -> C(m)	No Sí	+ C(m)
C2	A(m) B(m) B(j) C(a) C(m)	A(m), B(j) -> C(j) A(m), C(m) -> D(m)	Sí No	+ C(j)
C3	A(m) B(m) B(j) C(a) C(m) C(j)	A(m), C(m) -> D(m)	Sí	+ D(m)
C4	A(m) B(m) B(j) C(a) C(m) C(j) D(m)	Vacía	FIN	

Motor de inferencia

El motor de inferencia realiza tres funciones: equiparación (determinar qué instancias pueden ejecutarse, el llamado conjunto conflicto); resolución de conflictos (escoger qué instancia se ejecuta) y ejecución (modifica hechos, reglas o realiza alguna acción).

Hay diferentes estrategias para la resolución de conflictos: más nueva, más antigua, más específica, más general, aleatorio…

Es importante destacar que no se activan instancias que se han activado ya (refracción).

Utilizaremos siempre encadenamiento hacia delante, cuyas ventajas son que simula el pensamiento humano y que es declarativo (da igual el orden de resolución), pero son ineficientes y es difícil representar el control de flujo. Se usa en planificación, sistemas expertos…

Tema 3: Búsqueda no informada

Introducción

Vamos a representar los problemas con grafos, donde los nodos son el estado completo, y los arcos las transiciones entre estados. Lo que vamos a buscar es un camino que lleve del estado inicial a un estado final.

Vamos a tomar sistemas con las siguientes restricciones: entorno estático, entorno discreto (se escoge entre un número de opciones fijas) y determinismo (el resultado de cada operación está bien definido).

La búsqueda consiste en recorrer el grafo en un orden determinado. Para evitar ciclos, se puede controlar que no se repitan nodos.

Necesitaremos conocer tres propiedades: b (factor de ramificación), d (profundidad hasta la solución) y m (máxima longitud de un camino).

Para comparar los algoritmos podemos utilizar diferentes métricas, como la completitud (encuentra una solución siempre, si existe), optimalidad (encuentra la mejor solución), complejidad temporal, complejidad espacial, eficiencia…

Búsqueda en amplitud

Este algoritmo es completo (encuentra siempre solución) y además, óptimo (encuentra siempre la solución menos profunda). Su complejidad espacial y temporal es $O(b^d)$.

El algoritmo es el siguiente:

1. Crear LISTA con el nodo inicial
2. Set ÉXITO = false
3. Hasta que LISTA esté vacía o ÉXITO==true
    a. Generar los sucesores del nodo actual (sin repeticiones)
    b. Si algún sucesor es el nodo meta, set ÉXITO = true
    c. Si no, añadir sucesores al final de LISTA
4. Si ÉXITO==true
    a. Solución: camino desde la raíz al nodo meta.
5. Si no,
    a. Solución: fracaso

Su principal variante es el algoritmo Dijkstra, que consiste en una especie de algoritmo en amplitud pero teniendo en cuenta los costes para decidir a cuál vamos. En cada iteración se coge el de menor coste.

Búsqueda en profundidad

Se produce retroceso (backtracking) cuando se ha llegado al final de los sucesores de un nodo, al límite de profundidad, se repite el estado actual con uno ya estudiado o se sabe que el estado no conduce a la solución (poda).

El algoritmo es el siguiente:

1. Crear LISTA con el nodo inicial
2. Set ÉXITO = false y M = profundidad máx.
3. Hasta que LISTA esté vacía o ÉXITO==true
    a. Generar los sucesores del nodo actual (sin repeticiones)
    b. Si algún sucesor es el nodo meta, set ÉXITO = true
    c. Si no, añadir sucesores al principio de LISTA
4. Si ÉXITO==true
    a. Solución: camino desde la raíz al nodo meta.
5. Si no,
    a. Solución: fracaso

No es completo, aunque con detección de ciclos y backtracking sí. Tampoco es óptimo, pero es muy eficiente si la meta está alejada o hay problemas de memoria. Su complejidad temporal es la misma que el de amplitud, pero su complejidad espacial es O(bd).

Este algoritmo tiene algunas variantes que son profundización iterativa (vamos aumentando el límite de profundidad cada vez, haciéndolo completo) y profundidad bidireccional (buscamos desde arriba y desde abajo a la vez, reduciendo la complejidad temporal).

Tema 4: Búsqueda heurística

Introducción

Mantiene las características de la búsqueda no informada, pero a la hora de decidir, tenemos información que nos permite mejorar el algoritmo. Lo importante a la hora de implementar la búsqueda heurística es que podamos ordenar los nodos.

Heurísticas

Llamamos heurística al conocimiento parcial sobre un problema que permite resolver problemas más eficientemente. El estado puede darnos información parcial de cómo resolver el problema.
Por ejemplo, en un plano, una heurística en un plano de una ciudad podría ser la distancia en línea recta entre dos puntos. Nos da una información relativamente acertada de lo lejos que están, pero no el camino real (por ejemplo, no cuenta calles prohibidas, edificios…).

Las funciones heurísticas se descubren resolviendo modelos relajados (con menos restricciones) que el modelo original.

Decimos que una heurística es admisible si para cada nodo, la función heurística nunca sobreestima la distancia del camino real por el mejor camino posible. Es decir, el valor de la función heurística en cada nodo nunca debe sobrepasar el valor real del coste de ir desde dicho nodo hasta la meta por el mejor camino posible.

Búsqueda avara (greedy)

La idea es elegir en cada momento el mejor nodo entre los sucesores directos del nodo actual. Es análoga a la búsqueda en profundidad, pero con heurística.
Como solo utiliza información local, no gasta mucha memoria. Una variante de este algoritmo es la búsqueda local en escalada (hill climbing), en la que directamente solo generamos los sucesores cuyo valor de la función heurística sea mejor que el del padre.

La búsqueda avara por lo general no hace backtracking ni controla la repetición, y la de escalada tampoco. Por tanto, no es completa ni óptima, pero es rápida y eficiente si la función es monótona.

Algoritmos de primero el mejor

La idea es elegir en cada momento el nodo más prometedor entre una lista de pretendientes. Son análogos a la búsqueda en amplitud, pero con heurística.
Al guardar información de todos los nodos posibles, requiere memoria exponencial. Además usa detección de ciclos. Consisten básicamente en ir generando todos los nodos posibles y meterlos en la lista ABIERTA, y escoger de ABIERTA el mejor, para luego reordenar ABIERTA.

Hay un algoritmo en especial, el A*, que utiliza una función para elegir, f(n), que consiste en la suma de la heurística, h(n), y el coste acumulado hasta ese nodo, g(n). Sólo se llega a la meta cuando se añade el nodo final a CERRADA (es decir, cuando se visita, en contraposición a los algoritmos anteriores en los que parábamos cuando se generaba).
Es completo y óptimo si los costes son positivos y la heurística es admisible.

Tema 5: Razonamiento bayesiano

Incertidumbre

En la práctica, no podemos utilizar lógica clásica para razonar porque existe la incertidumbre. Esta incertidumbre puede ser de representación (puede que no exista conocimiento completo del problema, vaguedad en la representación…) o limitaciones prácticas (ignorancia práctica, coste computacional…).

Probabilidad

La probabilidad puede interpretarse o bien como la proporción de veces que algo es cierto, o bien como el grado de creencia en que algo es cierto. Representaremos las variables aleatorias como una letra mayúscula: X.

Probabilidad condicionada

Representamos por P(X|Y) la probabilidad de que ocurra el evento X dado que ocurre el evento Y.

P(X|Y) = P(X^Y) / P(Y), dado que P(Y) != 0.

Se puede representar con una matriz, donde las columnas suman 1. Las filas son las X para distintos valores de Y, mientras que las columnas son las Y para distintos valores de X.

Regla del producto: P(A^B) = P(A|B) x P(B)

Regla de la cadena: P(X1, X2… Xn) = P(Xn) x P(Xn|X1, X2… X(n-1))

Teorema de Bayes: P(A|B) = P(B|A) x P(A) / P(B)

Razonamiento bayesiano

Utilizaremos el razonamiento bayesiano para realizar tres clases de tareas:

- Predicción (ej: saber si una zona se inundará según la meteorología).
- Diagnóstico (ej: determinar si alguien está enfermo según unos síntomas).
- Clasificación (ej: determinar a qué clase pertenece una observación).

Tema 6: Redes bayesianas

Conceptos básicos

Decimos que dos variables son independientes si no están relacionadas. Es decir, A y B son independientes si P(A|B) = P(A) o viceversa.
Decimos que X e Y son condicionalmente independientes respecto de Z si P(X|Y, Z) = P(X|Z).

La independencia condicional nos va a servir para reducir el número de variables que se relacionan entre sí. Así se facilita el cálculo y necesitamos realizar menos cálculos.

Representación con redes bayesianas

Una red bayesiana es un grafo dirigido, sin ciclos, donde hay un nodo por variable. Un arco X -> Y significa que X tiene una influencia directa sobre Y, es decir, que P(Y) = P(Y|X).
Una tabla de probabilidad condicional con k padres tiene 2^k filas.

Ejemplo:

A	P(j|A)
a	0.9
¬a	0.05

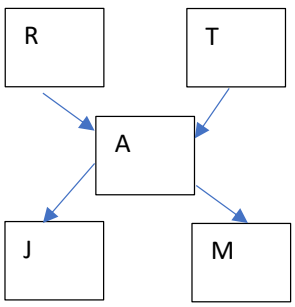
Nótese que no tienen por qué sumar 1 las columnas, sino que serían las filas (donde aparecería P(¬j|A) donde sí que se tendría en cuenta que sumen 1.

La regla genérica es que P(X1, X2… Xn) = prod. (P(Xi) | Padres(Xi))

Inferencia con redes bayesianas

La tarea principal es calcular probabilidades dada cierta evidencia. Habrá que hacer consultas, y habrá algunas variables que estén ocultas.

Ejemplo: Dada la siguiente red bayesiana, calcular la probabilidad de que haya robo dado que llaman Juan y María.



P(r| j, m) = a * P(r,j,m) = a * [P(r,j,m| t,a) + P(r,j,m|¬t,a) + P(r,j,m| t,¬a) + P(r,j,m |¬t,¬a)

Calculamos P(r,j,m | t,a) como P(r) * P(t) * P(a|r,t) * P(j|a) * P(m|a)
Esto es así porque si recordamos solo se usa la del padre inmediatamente superior.
Siguiendo el ejemplo se haría también P(¬r | j,m). Esto nos dará dos numeritos que podemos normalizar haciendo:

P(r | j,m) = P(r | j,m) / [P(r |j,m) + P(¬r | j,m)] y P(¬r | j,m) = 1 – P(r| j,m)

Una vez hechos todos estos cálculos tendremos las probabilidades correctas.

Hay otras técnicas, como el muestreo directo, y aplicaciones como los modelos ingenuos de Bayes (naïve Bayes Models), que veremos más adelante en temas posteriores.

Tema 7: Modelos de Markov

Introducción a los modelos de Markov

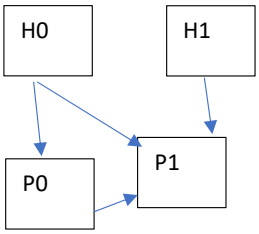
Se trata de hacer razonamiento probabilístico en el tiempo (ej. Llueve hoy, qué probabilidad habrá de que llueva mañana). Se pueden ver en forma de cadena, donde el pasado influye en el futuro, pero el futuro no influye en el pasado.

X0 -> X1 -> X2 y X0 -> X2.

Sin embargo, para simplificar esto (ya que el número de arcos crece exponencialmente con el tiempo), podemos aplicar la suposición de Markov, es decir, suponer que un instante de tiempo solo depende del instante inmediatamente anterior. Es decir, dado el presente, el pasado y el futuro son independientes entre sí (proceso estacionario).

Modelos ocultos de Markov

Quieren decir que hay hechos ocultos, que no podemos ver directamente.



Donde H son los hechos ocultos (hidden) y P los observables (perceptibles).

Aplicaremos las mismas dos simplificaciones de siempre:

- El instante t solo depende de t-1
- Es estacionario (me da igual coger t1-t2, que t2-t3, que tn-tn-1).

Se representan con forma de diagramas de transiciones, que veremos más adelante.

Modelos de Markov

P (lluvia -> sol) = 0.1, P (sol -> lluvia) = 0.1, P (sol -> sol) = 0.9, P (lluvia -> lluvia) = 0.9.

¿Cuál es la probabilidad de que en el día 2 haga sol?

P(sol – sol – sol) = 0.9 * 0.9

P(sol – lluvia – sol) = 0.1*0.1

P(sol día 2) = 0.81 + 0.01 = 0.82

Tareas de inferencia:

1. Filtrado o monitorización: dada una evidencia histórica, probabilidad del estado actual.
2. Predicción: dada una evidencia histórica, probabilidad de estados futuros.
3. Suavizado: Dada evidencia actual, estimar estados pasados.
4. Explicación más probable: Encontrar dada una evidencia los valores que maximicen (por ejemplo, en el problema anterior, sol-sol-sol).

Tema 8: Procesos de decisión de Markov

Definición

La distribución de probabilidad de un estado va a depender de la acción escogida. Se utilizarán los procesos de decisión de Markov para atacar este tipo de problemas.

Se definen con una tupla <S, A, P, R, g> donde S es el conjunto de estados, A el conjunto de acciones, P la tabla de transiciones y probabilidades, R la función de refuerzo y g el factor de descuento. Nosotros vamos a tomar como que R=coste y g=1 siempre, y nuestra intención va a ser minimizar el coste.

Para cada MDP definimos una política p, que es una forma de decidir qué acción realizamos en cualquier estado.

El MDP tiene que tener una serie de propiedades:

- Estados observables
- El resultado de S' solo depende de S (markoviano)
- El proceso es estacionario: para s,a Pa(S'|S) siempre. Es decir, depende solo del estado anterior.
- Para cada estado y acción hay una distribución de probabilidad (determinista).

Si la política es completa siempre sabremos qué hacer, y si la política es la que toma los menores valores es óptima.

Llamamos V(s) al valor que "pagamos" para llegar a la meta desde S. Es el coste esperado de la estrategia óptima para ir desde S hasta la meta. Por tanto la política p es una función que toma un estado S y devuelve una acción A. Resolver un MDP será encontrar una política óptima para el estado inicial.

Resolución de un MDP

Se calcula para cada estado y acción el coste esperado, Q(s,a):

$Q(s,a) = c(a) + \sum_{s' \in s} [Pa(s'|s) * V(s')]$

El valor de s será el mínimo de estas Q, y la política óptima será la a que de ese mínimo.

mín (Q(s,a)).

Utilizaremos el algoritmo por iteración de valores, que consiste en lo siguiente:

1. Inicialmente, V(s) = 0 para todos los estados.
2. Para cada posible estado distinto de la meta (LOOP):
    a. $V(s) = \text{mín}_{a€A(s)} (c(a) + \sum_{s'€s} [Pa(s'|s) * V(s')])$
3. Una vez hechos todos los V(s) volver a hacer LOOP hasta que converjan los valores.
4. Para ver la política p(S) se hace una iteración más desde S (estado inicial) y se halla el mínimo.

Tema 9: Sistemas con lógica borrosa

Introducción

Los problemas reales tienen cierta incertidumbre. Normalmente las cosas no son ciertas o falsas, sino que hay cierto grado de verdad. Hay dos tipos de incertidumbre: la vaguedad y la probabilidad.

La lógica borrosa se utiliza en los controladores (p. ej. Robots) porque no hace falta simular el sistema físico y se construyen con poquitas reglas. También se usa en tomas de decisiones.

Teoría de conjuntos

En lógica clásica se podían utilizar las operaciones de conjuntos. En un conjunto borroso se puede asociar un grado de pertenencia de cada elemento a cada conjunto entre [0,1]. Por ejemplo, todos los animales son rápidos, en cierta medida.

Decimos que los conjuntos borrosos se superponen porque un mismo elemento puede tener distintos grados de pertenencia a varios conjuntos. Para operar con ellos se siguen las siguientes reglas:

- Unión de conjuntos = máximo entre las dos funciones.
- Intersección de conjuntos: mínimo entre las dos funciones.
- Complementario de un conjunto: (1 – la función)

Lógica borrosa

Se usan funciones en lugar de pertenencia o de true/false. La teoría de conjuntos borrosos puede utilizarse para representar expresiones en lenguaje natural (variable lingüística).

Los modificadores lingüísticos (muy, poco…) constituyen nuevos conjuntos borrosos. Las proposiciones borrosas pueden ser atómicas o compuestas.

Inferencia borrosa

Hay que definir los extremos (donde la función toma el valor 1 y donde toma el valor 0). El razonamiento borroso es una generalización de los métodos de inferencia de la lógica binaria, que se implementa como IF <prop borrosa> THEN <prop borrosa>.

Requisitos del mecanismo de inferencia:

- Propagar la incertidumbre a través del modus ponens
- Combinar la incertidumbre con el OR y el AND
- Combinar la incertidumbre de dos reglas que concluyan lo mismo.

Método de Mamdani, método máx-min. Para calcular el consecuente, si el antecedente no es totalmente cierto se queda con una parte (se trunca) el conjunto.

Si el antecedente no fuese borroso, se representaría como un único conjunto borroso con un único valor y se seguiría con el procedimiento habitual.

Para desborrosificar:

- Centro de gravedad
- Media de los máximos
- Media de los centros de los triángulos

La técnica que se ve en este curso es la del centro de gravedad, y se ve en los ejercicios con mayor detalle.

Diseño de sistemas basados en lógica borrosa

Esquema de un sistema tipo Mamdani:

1. Borrosificación (si hay entradas borrosas)
2. Inferencia (con las técnicas ya vistas)
3. Deborrosificación
4. (En sistemas de control, realimentación de la entrada)

Tema 10: Aprendizaje automático

Introducción

Se trata de hacer programas que mejoren a partir de la experiencia. La motivación de esto es introducir conocimiento a través de ejemplos (sobre todo, para problemas mal definidos o sin algoritmos conocidos) y que no hay suficiente cantidad de personas que pueda analizar tal cantidad de datos.

Algunos ejemplos de uso del aprendizaje automático (machine learning en inglés) son sanidad (diagnóstico), banca, domótica, robots…

Definiciones

Tipos de tareas:

a. Clasificación: dados unos datos, asignarles una clase.
b. Predicción/regresión: dados unos datos, predecir un valor futuro.
c. Agrupación/clustering: hacer grupos en base a unos atributos.
d. Diagnóstico
e. Caracterización: dados n grupos, ver en qué se parecen entre sí.
f. Optimización

Llamamos atributo a cada característica que define a un elemento. Las instancias son conjuntos de valores de estos atributos (cada ejemplo), y clase, los grupos en los que quiero dividir a mis instancias.

Llamamos generalización de un conjunto de ejemplos de una clase (o hipótesis) a la descripción que representa a las instancias de una clase (incluso a las no observadas). Cada una de esas instancias se llama ejemplo positivo de esa clase.

Técnicas de aprendizaje automático

- Aprendizaje no supervisado: las observaciones no tienen una clase asociada. Se buscan patrones en los datos y se usa para describirlos.
- Aprendizaje supervisado: cada observación incluye el valor de la clase a la que pertenece y se usa en predicción.
- Aprendizaje por refuerzo: el sistema recibe recompensa en función de las decisiones que toma (Q-learning).

Diseño de sistemas de aprendizaje automático

Etapas:

1. Elegir qué datos, atributos… usaremos.
2. Definir el tipo de la función objetivo.
3. Representar la función objetivo.
4. Seleccionar algoritmo de aprendizaje automático y aplicarlo a la función objetivo.
5. Evaluar el sistema.

Problemas:

- Atributos mal elegidos
- Ruido
- Demasiados atributos
- Pocos datos

Tema 11: IA en robótica

Un robot es una máquina programable que opera de forma autónoma e interactúa con el entorno mediante sensores y actuadores. Ej: rovers, UAVs…

Podríamos utilizar lógica clásica o lógica borrosa y aprender por ejemplo con refuerzo. El comportamiento puede ser reactivo (reglas) o deliberativo (MDP). El deliberativo se hace de forma aleatoria hasta que se tiene la tabla de probabilidades y luego ya MDP.

Algunos dominios son multi-robot: surge la necesidad de comunicarse y cooperar. La coordinación puede ser centralizada (Si R1 cerca de R2, mueve R2) o distribuida (si estoy cerca de otro robot, me muevo).

Tema 12: Algoritmos bioinspirados

Redes de neuronas

Aprenden a partir de ejemplos (aprendizaje automático) para problemas de agrupación, clasificación, predicción…

Si tenemos muchas de estas interconectadas, en una red de neuronas, que propagan las señales y tienen reglas de aprendizaje que van modificando los pesos.

El aprendizaje consiste en determinar pesos y umbral a partir de los ejemplos, y puede ser supervisado (usamos la salida que queremos obtener para guiarlos) o no supervisado. Problema de sobreajuste: la red no reacciona ante casos nuevos.

Ejemplos: perceptrón simple, adaline, mapas de Kohonen.

Las variables de E/S tienen que ser numéricas.

Algoritmos evolutivos

Búsqueda basada en evolución simulada:

- Se tienen soluciones potenciales.
- Se conservan las adecuadas y se tiran las que no lo son.
- Se termina cuando se obtiene una solución “suficientemente buena”.

Funcionan por generaciones (ciclos). En cada ciclo se tiene una población formada por individuos. Cada individuo es una solución, y se usa una función de adecuación (fitness) para evaluarlos.

Llamamos genotipo al cromosoma (cadenas de bits) y fenotipo al individuo (el significado de los bits.

Operaciones:

- Selección (ruleta o torneo [el mejor]).
- Reemplazo.
- Cruce (mezcla).
- Mutación (cambiar 1 bit).