Resumen 1

********** enunciado **********

Resumen de la sesión del miércoles 29 de Enero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "1 Introducción a la Inteligencia Artificial"

********** mi resumen **********

Resumen día 29 de Enero:

Vimos una introducción sobre la historia de la inteligencia artificial. Se basa en agentes racionales. Un agente racional es una entidad que percibe y actúa.

********** Nota y aclaraciones del profesor **********

Calificación 8,00 / 10,00

Comentarios de retroalimentación:

Si bien es cierto que abstraemos a un agente como una función que va de secuencias de percepciones a la acción siguiente a realizar, en esta presentación también discutimos diferentes caracterizaciones de la inteligencia, tratando de decidir cuál nos interesa más.
Consideramos caracterizaciones de la inteligencia desde 2 ejes, actuar vs pensar, y hacerlo como humanos vs hacerlo racionalmente. Eso nos da 4 posibles combinaciones que consideramos, y la conclusión es que actuar racionalmente, es decir, actuar para conseguir un objetivo medible, es la caracterización de la inteligencia que nos interesa en este curso. Nos interesa porque es la más sencilla de medir y la que tiene más aplicación para construir sistemas útiles en la práctica. Pensar racionalmente no es siempre necesario para construir un agente útil, por ejemplo un termostato se puede considerar un agente que actúa para conseguir el objetivo de mantener una habitación a temperatura constante, y por tanto actúa racionalmente, pero no realiza ningún proceso de pensamiento racional o razonamiento.

****

Resumen 2

********** enunciado **********

Resumen de la primera sesión del jueves 30 de Enero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "2 Agentes inteligentes"

********** mi resumen **********

Resumen 30 de Enero:

Vimos una visión un poco más detallada sobre los agentes racionales. Éstos están en un entorno percibiendo todo el rato y actúa con actuadores, sensores... Podemos definir el agente mediante una función.

El entorno cambia por el agente o por él mismo. Hay varios tipos de entornos: observables, episódicos, estáticos..

Ejemplo: una aspiradora.

********** Nota y aclaraciones del profesor **********

Calificación 10,00 / 10,00

Comentarios de retroalimentación:
Hubiera estado bien que también comentaras que vimos varias arquitecturas de agentes de alto nivel.

****

Resumen 3

********** enunciado **********

Resumen de la segunda sesión del jueves 30 de Enero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "3 Resolución de problemas y búsqueda no informada", páginas 1 a 6

********** mi resumen **********

Resumen dia 30 de Enero:

Vimos la idea de tener un entorno observable, discreto y determinista en el que el nuevo estado del entorno resultante de una acción se puede determinar antes de realizar la acción.

ejemplo de clase: como llegar a Rumanía por el camino más corto.

********** Nota y aclaraciones del profesor **********

Calificación 8,00 / 10,00

Comentarios de retroalimentación:
Los agentes de resolución de problemas modelan el entorno como un espacio de estados, entre los que nos movemos mediante acciones, cuyo resultado define una función de sucesor. Los agentes planifican un camino desde el estado actual hasta un estado que cumple el test de estado objetivo, buscando en ese espacio de estado, y ejecutan ese plan ignorando las percepciones del entorno

****

Resumen 4

********** enunciado **********

Resumen de la sesión del miércoles 5 de Febrero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "3 Resolución de problemas y búsqueda no informada", páginas 7 a 35

********** mi resumen **********

Resumen dia 5 de Febrero:

Vimos la idea básica de un árbol de búsqueda que consiste en la exploración simulada del espacio de estados mediante la generación de sucesores de estados ya explorados.

Algunas estrategias uniformes de búsqueda que se distinguen por el orden en la expansión de sus nodos.

búsqueda en anchura
búsqueda en profundidad
búsqueda en profundidad con límite
búsqueda uniforme..

********** Nota y aclaraciones del profesor **********

Calificación 9,00 / 10,00

Comentarios de retroalimentación
Buen resumen, pero las estrategias no son uniformes, lo único que es uniforme es la estrategia de búsqueda por coste uniforme.

****

Resumen 5

********** enunciado **********

Resumen de la primera sesión del miércoles 12 de Febrero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación
"3 Resolución de problemas y búsqueda no informada", páginas 36 a 71

********** mi resumen **********

Resumen dia 12 de Febrero:

(la búsqueda en anchura)

Breadth-first search: que es una cola FIFO, consiste en recorrer comenzando desde la raíz e ir explorando todos los vecinos de este nodo. A continuación para cada uno de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el árbol.

(la búsqueda en profundidad)

Depth-first search: que es una cola LIFO, que consiste en ir expandiendo todos y cada uno de los nodos que va localizando, de forma recurrente, en un camino concreto.

iterative deeping search: combina los beneficios de las anteriores.

Los grafos de búsqueda son más eficientes que los árboles de búsqueda.

********** Nota y aclaraciones del profesor **********

Calificación 10,00 / 10,00

Comentarios de retroalimentación:
Buen resumen....

****

Resumen 6

********** enunciado **********

Resumen de la segunda sesión del miércoles 12 de Febrero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "4 Búsqueda informada"

********** mi resumen **********

Vimos diferentes algoritmos de búsqueda.

Best-first search: es un algoritmo de búsqueda que se expande hacia el nodo más prometedor elegido. Está implementada con una cola ordenada en orden decreciente de conveniencia.

Casos especiales:

Greedy search. (se preocupa por el futuro)
Sigue una heurística consistente en elegir la opción óptima en cada paso para llegar a la solución más óptima.
Estima el coste desde n hasta la meta más cercana. Vimos un ejemplo de como llegar hasta Bucarest. Expande el nodo que parece estar cercano al objetivo.

A* search (le da el mismo peso al pasado que al futuro)
Sigue la función heurística que evita expandir caminos que ya son costosos.
Es eficiente para cualquier heurística consistente, pero no se garantiza que otro algoritmo pueda expandirse con menos nodos. Puede ser poco práctica para problemas a gran escala pero podemos usar variantes de A* o heurísticas que no son admisibles pero aceleran la búsqueda. Pero el gran problema es la complejidad del espacio, puede haber muchos estados.

********** Nota y aclaraciones del profesor **********

Calificación 10,00 / 10,00

Comentarios de retroalimentación: Buen resumen....

****

Resumen 7

********** enunciado **********

Resumen de la primera sesión del miércoles 19 de Febrero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "5 Búsqueda local, páginas 1 a 12 "

********** mi resumen **********

En muchos problemas de optimización la ruta es irrelevante. En estos casos podemos usar algoritmos de mejora iterativos, que mantienen un estado actual y tratan de mejorarlo.

Varios ejemplos como el de las n-reinas, el problema del vendedor ambulante..

Hill-climbing: el algoritmo de escalada simple es un algoritmo iterativo que comienza con una solución arbitraria, luego intenta encontrar una mejor solución haciendo un cambio en la solución. Si el cambio produce una mejor solución, se realiza otro cambio en la solución, y así sucesivamente hasta no encontrar más mejoras. (como hemos visto aplicado al problema de las n-reinas)

Variantes del hill climbing:

sideways moves: permitir movimientos cuando no hay movimientos hacia arriba.
Stochastic hill climbing: la escalada estocástica elige al azar los movimientos cuesta arriba.
first-choice hill climbing: la escalada de primera elección implementa la escalada estocástica de colinas generando sucesores aleatoriamente hasta que se genere uno que sea mejor que el estado actual.
Estos algoritmos son incompletos, pueden quedarse atrapados y no poder moverse. Intenta solucionarse con:

Random-restart hill climbing: la idea es recorrer desde un estado aleatorio, si encontramos una solución antes de un tiempo de espera la devolvemos, si no, detenemos la búsqueda y comenzamos de nuevo desde un estado nuevo aleatorio y se va iterando hasta encontrar una solución.
El coste es mayor porque tenemos que ejecutar la búsqueda muchas veces, pero para algunos problemas es eficaz

********** Nota y aclaraciones del profesor **********

Calificación 8,00 / 10,00

Comentarios de retroalimentación:
No todos los estados son solución, sólo son solución aquellos estados que pasan el goal test de la formulación del problema: eso se dice en la página 6 de la presentación. Recordemos que estamos resolviendo problemas definidos tal y como describimos en la presentación sobre búsqueda no informada, en la página 6. Por ejemplo, un estado de las n-reinas en las que alguna reinas se atacan no es una solución, sólo son solución los estados en los que las reinas no se atacan, que son los que pasan el goal test. De hecho que un estado sea solución, es decir, que pase el goal test, es la condición de parada de random-restart hill climbing.
Las variantes de hill climbing distintas de random-restart son incompletas porque se pueden quedar paradas en un estado que no es solución. Sí hill climbing se queda atrapado en un estado que es solución eso no es problema sino el caso mejor, porque en ese caso hill-climbing devolverá una solución como resultado

****

Resumen 8

********** enunciado **********

Resumen de la segunda sesión del miércoles 19 de Febrero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "5 Búsqueda local, páginas 13 a 20 "

********** mi resumen **********

Haciendo un resumen vemos que el algoritmo Hill climbing es eficiente pero no completo, ya que puede atascarse, y el Random walk es completo pero ineficiente.

El algoritmo Simulated annealing combina ambos algoritmos (Hill climbing y Random walk) para lograr eficiencia y completitud.

La idea es escapar de los máximos locales al permitir algunos movimientos malos pero gradualmente disminuir su tamaño y frecuencia.

El algoritmo Local Beam search realiza un seguimiento de ‘k’ estados, similar al de Hill climbing. Comienza con ‘k’ estados generados aleatoriamente, para cada caso genera sucesores de todos los ‘k’ estados, si obtenemos un mejor estado en los sucesores devuelve el mejor y si no, ‘k’ itera.

El problema es que los estados ‘k’ no son diversos, se concentran en una pequeña región del espacio de estados, por lo que busca una versión más costosa de la escalada. La solución es la Stochastic Local Beam search, análoga a la Stochastic Hill Climbing.

********** Nota y aclaraciones del profesor **********

Calificación 10,00 / 10,00

Comentarios de retroalimentación:
Buen resumen.
Entiendo que con "disminuir su tamaño" te refieres a la diferencia entre el valor del estado actual y el valor del sucesor, que influye en la probabilidad de moverse a un sucesor peor.

****

Resumen 9

********** enunciado **********

Resumen de la primera sesión del miércoles 26 de Febrero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "5 Búsqueda local", páginas 21 a 23

********** mi resumen **********

Vemos un repaso de lo visto anteriormente y la última parte del tema "5 Búsqueda local".

Los algoritmos genéticos (Genetic algorithms) juntan la búsqueda local estocástica + genera sucesores a partir de pares de estados. Ahora un sucesor proviene de combinar dos estados principales.

Comienza con ‘k’ estados generados aleatoriamente, que llamamos población; cada uno de esos estados se llama individual
Cada individuo se representa como una cadena sobre un alfabeto finito.
Define una función de aptitud que debería devolver valores más altos para mejores estados.
La probabilidad de ser elegido para la reproducción es directamente proporcional a la aptitud.
Por último vemos como algunos de los algoritmos de búsqueda locales vistos no funcionan para resolver el problema propuesto de querer ubicar tres aeropuertos en Rumanía, pero hay otros como: first-choice, hill-climbing y simulated annealing, si podrían resolver el problema pero estos algoritmos escogen un sucesor aleatorio. Pero la mejor forma de resolverlo es con los espacios de estado continuo.
Los métodos de discretización convierten el espacio continuo en espacio discreto. Usan métodos de gradiente. Los gradientes nos dan un criterio para elegir el próximo vecino sin considerar un conjunto infinito de opciones.

********** Nota y aclaraciones del profesor **********

Calificación 10,00 / 10,00

Comentarios de retroalimentación:
Muy buen resumen

****

Resumen 10

********** enunciado **********

Resumen de la segunda sesión del miércoles 26 de Febrero.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "6 Búsqueda con adversario y juegos", páginas 1 a 34

********** mi resumen **********

Tipos de juegos: determinista, casualidad, información perfecta, información imperfecta.

En un problema de búsqueda normal, la solución óptima sería una secuencia de acciones que conduzcan a un estado objetivo. En una búsqueda contradictoria, MIN tiene algo que decir al respecto. MAX no tiene forma de saber con certeza qué efectuará un movimiento en la jugada, porque MIN puede reaccionar de cualquier manera (análoga al problema de la búsqueda con acciones no deterministas).

En lugar de sucesores consideramos:

Acciones: conjunto de posibles acciones del estado ‘s’.
Resultados: conjunto de posibles estados resultantes de realizar una acción en un estado s.
Solución un subárbol que:

- tiene un nodo objetivo en cada hoja
- especifica una acción en cada uno de sus nodos OR
- incluye cada rama de resultado en cada uno de sus nodos AND.

Minimax elige moverse a la posición con el valor minimax más alto.
La estrategia óptima se puede determinar a partir del valor mínimo de cada nodo.
La decisión minimax en la raíz es la acción A1, que es la opción óptima para MAX porque conduce al estado con el valor minimax más alto.

La poda α-β es una técnica para calcular el valor correcto de minimax sin mirar cada nodo en el árbol del juego.
α: estimación más alta del valor minimax que hemos visto hasta ahora desde un nodo MAX.
β: estimación más baja del valor minimax que hemos visto hasta ahora desde un nodo MIN.

El algoritmo α-β es muy similar al algoritmo minimax, excepto por el mantenimiento de α y β, y la prueba de poda. La poda no afecta el resultado final: calcula correctamente la decisión minimax.

La búsqueda α-β todavía tiene que ir hasta los estados terminales de una parte del espacio del juego: esos son los casos base de la recursividad. La profundidad no es práctica para decidir movimientos en pocos minutos.

********** Nota y aclaraciones del profesor **********

Calificación 10,00 / 10,00

Comentarios de retroalimentación:
Buen resumen.
En el algoritmo minimax la estrategia óptima no se determina a partir del valor mínimo de cada nodo: lo que se hace es suponer que el jugador MAX tomará la decisión que maximiza la utility, por tanto cogiendo la rama que maximice el valor minimax sobre el subárbol correspondiente, mientras que el jugador MIN tomará la decisión que minimiza la utility, por tanto cogiendo la rama que minimice el valor minimax sobre el subárbol correspondiente

****

Resumen 11

********** enunciado **********

Resumen de la primera sesión del miércoles 4 de Marzo.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "6 Búsqueda con adversario y juegos", páginas 35 a 44

********** mi resumen **********

El algoritmo para el juego no determinista 'Expect Minimax' ofrece un juego perfecto, al igual que 'Minimax', excepto que también debemos manejar los nos de cambio usando el valor esperado para los nodos casuales.

En los juegos no deterministas también usamos Eval(evaluación) y CutOff. Pero el uso de una función que define el mismo orden sobre los estados que la utilidad no garantiza que se tome la misma decisión de 'minimax'.

Esto se debe a que 'Expected Minimax' combina valores no solo con las operaciones min y max, que solo se preocupan por el orden, sino que también se combinan usando un promedio ponderado, que se preocupa por la magnitud relativa de los resultados.

También hay una variante de alfa-beta con nodos casuales, pero es mucho menos efectiva. A medida que aumenta la profundidad, la probabilidad de llegar a un nodo dado se reduce, el valor de búsqueda anticipada disminuye.

Una alternativa a alfa-beta con posibilidad de optimizar la búsqueda en H-Expect Minimax es la simulación de Monte Carlo. La simulación monte Carlo parte de un estado dado, juega miles de juegos usando tiradas de dados al azar para los nodos y búsquedas alfa-beta u otro algoritmo.

Los juegos de información imperfecta calculan el valor mínimo de cada acción en cada oferta, luego elige la acción con el mayor valor esperado sobre todas las ofertas.

Esto reduce el problema parcialmente observable (debido a la falta de información generada aleatoriamente) al problema determinista previo completamente observable.

También podemos usar H-Minimax en lugar de Minimax para ajustar el equilibrio de precisión / rendimiento.

La incertidumbre limita la asignación de valores a los estados.

Las decisiones óptimas dependen del estado de la información, no del estado real.

********** Nota y aclaraciones del profesor **********

Calificación 10,00 / 10,00
Comentarios de retroalimentación:
Buen resumen

****

Resumen 12

********** enunciado **********

Resumen de la segunda sesión del miércoles 4 de Marzo.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "7 Problemas de satisfacción de restricciones", páginas 1 a 31.

********** mi resumen **********

Problemas de satisfacción de restricciones.

La prueba de objetivos es un conjunto de restricciones que especifican combinaciones permitidas de valores para subconjuntos de variables.

Es un simple ejemplo de lenguaje de representación formal. Proporciona algoritmos útiles de propósito general con más potencia que el espacio estándar.

- La asignación de un valor a una variable a menudo hace que el conjunto de valores admisibles para otras variables sea más pequeño: propagación de restricciones.
- Una vez que una asignación de valores a algunas variables no es una solución, podemos dejar de asignar valores inmediatamente a otros valores, ya que eso no conducirá a una solución.
- Podemos centrarnos en las variables involucradas en más restricciones, especialmente aquellas difíciles de satisfacer, para podar el espacio de búsqueda más rápido.

Vemos un ejemplo: mapa-coloreado. Cuya solución al problema son tareas que satisfacen todas las limitaciones.

Problemas de satisfacción de restricciones binarias, cada restricción se relaciona como máximo con dos variables.
Los algoritmos CSP de uso general utilizan la estructura gráfica para acelerar la búsqueda.

Variedades de restricciones:

- Las restricciones unarias implican una sola variable.
- Las restricciones binarias implican pares de variables.
- Las restricciones de orden superior implican 3 o más variables.

Preferencias (restricciones suaves) a menudo representables por un costo para cada asignación variable. Problemas de optimización restringidos.

Restricción disyuntiva: alternativa entre 2 restricciones, solo una de ellas necesita ser satisfecha para que toda la restricción sea satisfecha.
Búsqueda de retroceso

Las asignaciones variables son conmutativas. solo necesita considerar asignaciones a una sola variable en cada nodo.

Cada nivel del árbol corresponde a una variable particular.

Búsqueda en profundidad de CSP con asignaciones de una sola variable se denomina búsqueda de retroceso.

Necesitamos llegar a profundidad 'n' de todos modos para encontrar una asignación para todas las variables. Es el algoritmo desinformado básico para los CSP.

El algoritmo alterna entre:

- Buscar: elegir nuevas asignaciones de variables.
- Propagación de restricciones: tipo específico de inferencia que elimina los dominios de las variables.
- Valores mínimos restantes. Elige la variable con la menor cantidad de valores legales.

Grado heurístico. Elige la variable con más restricciones en las variables restantes.

Valor de menor restricción. Dada una variable, elegir el valor menos restrictivo, el que descarta la menor cantidad de valores en las variables restantes.

Comprobación de avance. Realiza un seguimiento de los valores legales restantes para las variables no asignadas. La búsqueda finaliza cuando alguna variable no tiene valores legales.

Consistencia del arco. La forma más simple de propagación hace que cada arco sea consistente.

********** Nota y aclaraciones del profesor **********

Calificación 9,00 / 10,00

Comentarios de retroalimentación:
Este resumen es excesivamente largo, por favor intenta hacer resúmenes más concisos en el futuro

****

Resumen 13

********** enunciado **********

Resumen de la clase de repaso del Tema 8.
Escribe un resumen corto de en torno a 140 caracteres de la lección vista en clase, correspondiente a la presentación "clase de repaso del Tema 8".

********** mi resumen **********

Learning agent.

Contamos con una representación factorizada (representación plana, atributo-valor). Cada ejemplo de entrada es un vector (vectores de atributos).
El conjunto de datos es del tipo : entrada-salida.
Necesitamos mapear las entradas para generar la mejor salida posible.

Tipos de aprendizaje:

Aprendizaje no supervisado:
Tenemos muchos datos pero no tenemos información asociada a esos datos.
El algoritmo agrupa los datos según lo que se parecen entre sí.

Aprendizaje supervisado:
Tenemos un conjunto de datos asociados. Lo que hacemos es aprender en base a esos datos, para hacer predicciones sobre estos datos y tener una función que los asocie.

ej.: tenemos muchos correos que son spam y otros que no, construyo un algoritmo para que aprenda y así poder decidir si el próximo correo es spam, ya tenemos datos clasificados (aprende de ellos) , en cambio el aprendizaje no supervisado solo los agrupan por similitud.

En resumen, partimos de un conjunto de entrenamiento (datos ya clasificados, desde los que partimos para aprender) construyó un algoritmo de entrenamiento para encontrar una hipótesis y para saber lo buena que es, se aplica sobre otro conjunto de datos que no haya visto el algoritmo y vemos si con lo que ha aprendido puede distinguir bien los otros datos y generalizar bien.

Tipos de aprendizaje supervisado, dependiendo del conjunto de valores que devuelve la función tenemos:

Clasificación:
si la salida es finita. ej: correos, clases: spam y no spam.

Regresión:
si la salida es un número real. ej: información de casas y precios, para cada casa tenemos unos atributos (metro cuadrado, habitaciones, zona..) cada casa tiene asociado un precio. El objetivo es hacer una función que dado una casa con sus características nos devuelva un número real, el precio.

Aprendizaje por refuerzo:

- castigos o recompensas.

Es iterativa, menos general, dirigido a cosas concretas y que se tenga que reaccionar en el momento.

Nosotros buscamos hipótesis en un espacio de hipótesis y tiene que ver con los datos y el algoritmo que usemos.

La función de hipótesis que obtengamos tiene que ser consistente con nuestros datos, debe cumplir los datos de entrenamiento o ajustarse a ellos.

Habrá varias hipótesis que se ajustan, pero nos quedamos con la más simple.

Lo que debe hacer la hipótesis que escogemos es generalizar mejor nuestros datos. La que mejor asocie nuestros datos y otros.

Idea de esto:
Tenemos un conjunto de datos -> usamos un algoritmo de aprendizaje -> este algoritmo tiene un espacio de hipotiposis -> nos quedamos con la hipótesis más simple.

Algoritmos de aprendizaje de decisión:

Árboles de decisión:

Sirven para hacer clasificación.
Genera un árbol de decisión (salida).

Es fácil de interpretar, cada nodo es una pregunta asociada a un atributo, las aristas son los valores de los atributos, las hojas son valores finales de clasificación.
Proporcional a la lógica proposicional.

Sirve para bastantes tareas pero no para todas. El espacio de hipótesis son todos los posibles árboles para nuestros datos.

Usamos los atributos más discriminatorios para tener el árbol de decisión más pequeño posible, nos guiamos por la entropía de los atributos, nos quedamos con los que tengan menos entropía.

****

Resumen 14

********** enunciado **********

Resumen de la segunda parte del Tema 8.

********** mi resumen **********

SOBRE-AJUSTE

Cuando un árbol de decisión tiene muchos datos de entrada. Se evita eliminando nodos con los que la ganancia de información es casi nula.

EVALUAR Y ELEGIR UNA HIPÓTESIS

Los datos de test tienen que ser similares a los datos de entrenamiento.
Podemos medir la precisión de una hipótesis según las veces que se equivoque. Se hace con los datos del test.

ESTIMACIÓN / EVALUACIÓN DE CALIDAD

holdout cross-validation: separa aleatoriamente los datos (entrenamiento, tes
k-fold cross-validation: (mejor) Separa los datos en k subconjuntos de igual tamaño. Hace k ejecuciones de aprendizaje.
Unos cuantos trozos para entrenar y otros para test y repetimos en varios experimentos. Luego se hace la media de esos experimentos.

HYPERPARAMETER

Si el algoritmo de aprendizaje tiene hiper parámetros hay que ajustarlos.

conjunto de test
conjunto de entrenamiento
conjunto de validación

SELECCIÓN DE MODELO

Buscar el espacio de hipótesis mejor para no sobre ajustar los datos

Optimizar: dentro del espacio de hipótesis encontrando la mejor.
Medimos cuánto se equivoca un algoritmo en una predicción si se usa una función de coste.
Para minimizar la función de coste:

gradient descent: para encontrar dentro del espacio de peso el valor menor, gracias al valor que marcan las derivadas parciales.
Stochastic gradient descent: solo coge unos pocos ejemplos.

****

Resumen 15

********** enunciado **********

Resumen de la tercera parte del Tema 8.

********** mi resumen **********

La regresión lineal multivariable puede sufrir overfitting. Solución -> usar una regularización.

Esto define varias funciones de regularización. L1 además de reducir el sobreajuste, hace que el modelo tenga menos parámetros.

Para usar regresión lineal en la clasificación lineal debemos usar la regla de aprendizaje del perceptrón, ya que si no, la función umbral no es diferenciable.

Pero es mejor usar la regresión logística. Con ella usamos el descenso de gradiente y vemos los mejores valores de los pesos para minimizar la función de coste.

NEURONAS

Si el impulso eléctrico es suficientemente grande se conecta a la neurona.

La entrada a la neurona es una combinación lineal de los pesos y pasan por una función de activación. Esto nos da un valor y devuelve el nivel de activación de la neurona.

Estructuras:

- Feed-forward networks: va hacia delante. La información la reciben de la unidad anterior. Las neuronas no tienen un estado interno por niveles.
- Recurrent networks: hay unidades que pueden relacionarse hacia atrás, pueden producirse bucles.. Tienen memoria a corto plazo.
- Single-layer perceptrons: cada entrada se conecta con cada salida. Las salidas operan separadamente.
- Multilayer perception: Hay que hacer redes de más niveles (niveles intermedios ocultos). Así se puede representar cualquier función.

****

Resumen 16

********** enunciado **********

Resumen del Tema 9.

********** mi resumen **********

Sistemas basados en reglas. Técnicas basadas en conocimiento -> mediante reglas.

Definimos el comportamiento del sistema en base a reglas. Construimos reglas con las que vamos a decirle al sistema lo que tiene que hacer en cada momento.

Sistemas expertos: Un analista construye las reglas en base al conocimiento obtenido de un experto.

Tipos de programas.

Programas secuenciales: las instrucciones se ejecutan de forma secuencial.
Programas basados en eventos: dependiendo de la información que entre reconoce unos eventos y estos activan unas u otras reglas.
Arquitectura de un sistema de reglas.

Conocimiento : los datos que maneja el sistema de reglas.
- Base de reglas. conocimiento procedimental.
- Base de hechos. conocimiento factual.

Programa : cómo funciona el sistema de reglas.
- Intérprete de reglas.

En el sistema basado en reglas hay un ciclo de operación que ejecuta las reglas.
Reconocimiento. determina las reglas que son aplicables.
Selección de conflictos. decide qué reglas ejecutar mediante una estrategia de control.
Ejecución de la regla.
Se inicializa la agenda y vuelve al punto 1, hasta que se alcanza el objetivo.

Tipos de razonamientos:
- Encadenamiento progresivo: dirigido por los datos. (la que más se usa)
- Encadenamiento regresivo: dirigido por los objetivos.

Fase de reconocimiento (para saber qué reglas se van activan)

mediante el algoritmo RETE el sistema basado en reglas es eficiente.
En cada ciclo los cambios en la memoria de trabajo son los que activan las reglas. Los nuevos hechos generados son los que determinan las nuevas reglas.

****

Resumen 17

********** enunciado **********

Resumen del Tema 9. Drools.

********** mi resumen **********

Drools es un sistema de gestión de reglas.

Proporciona un motor de reglas que procesa hechos y produce resultados como resultado del procesamiento de reglas y hechos.

Drools se divide en dos:
La creación de Reglas archivos (archivos .DRL).
La creación de memoria de trabajo y manejando la activación.
Un sistema de reglas de producción es un motor de inferencia que puede escalar a una gran cantidad de reglas y hechos. El motor de inferencia compara hechos y datos con las reglas de producción para sacar conclusiones que se convierten en acciones.

La primera fase del sistema basado en reglas es Pattern Matching (emparejamiento de patrones): implementa una extensión del algoritmo RETE, el RETEOO.

****

Resumen 18

********** enunciado **********

Resumen del Tema 10.

********** mi resumen **********

Agente Software. Entidad autónoma que interacciona con el entorno

tipos: accesibles, inaccesibles, deterministas, no deterministas, discreto, continuo..

Características:

Entidades autónomas.

Pueden percibir su entorno y responden a cambios que se producen en el mismo.

El agente decide qué acciones realizar para cumplir unos objetivos, dependiendo de la configuración del entorno.

Debe tener inteligencia:

Tiene un razonamiento para decidir cómo actuar para alcanzar un objetivo.
Tienen que ser capaces de adaptarse a cambios.
Sistema multi-agente : Resolución de problemas mediante divide y vencerás. Heterogeneidad. Concurrencia y distribución.

Movilidad. Esos agentes se pueden mover de un nodo a otro en una red.

Comunicación entre agentes. Hace falta un lenguaje de comunicación.

Utilidad:

proporcionan aspectos sociales, lenguajes y protocolos de comunicación entre agentes, distribuyen datos, control, etc.
facilitan la evolución.
tienen un mayor grado de abstracción.

Diseño:

arquitectura BDI. Se basa en el razonamiento práctico.

arquitectura de pizarra. Las fuentes de información se activan de forma 'oportunista'.

JADE. Plataforma de agente más extendida.
ACC. implementa los servicios de mensajería.
DF. donde se registran los agentes.
AMS. gestiona los agentes en una plataforma.
El comportamiento se codifica mediante clases de objetos.
Existe una clasificación de comportamientos.
Existe un hilo de ejecución por agente. Cada agente es responsable de la ejecución de su comportamiento.

****

Resumen 19

********** enunciado **********

Resumen del Tema 10. JADE.

********** mi resumen **********

Implementación de sistemas multiagentes con JADE.

arquitectura distribuida.
contenedores con agentes.
un proceso java por cada contenedor.
Contenedor: instancia ejecutable de JADE.

En el contenedor principal siempre hay dos agentes especiales que siempre se ejecutan, el AMS y el DF.

AMS: sistema de gestión de agentes que se encarga de proveer nombres a los agentes. Se encarga de crear y destruir los agentes
DF: ofrece un servicio donde los agentes se pueden registrar y otros agentes pueden consultar qué agentes se están ejecutando en un determinado momento en el contenedor principal.
La responsabilidad del contenedor principal es manejar las tablas que contienen toda la información referente a los agentes.

Agentes.

El comportamiento de un agente viene determinado por los comportamientos que nosotros definimos en él. Para ello hay que crear comportamientos 'behaviours' y hay que decirle como debe ejecutarlo.

Cada comportamiento define una tarea o subtarea y son elementos java. Se ejecutan según un Round-Robin de igual frecuencia.

Tipos: One-shot, cyclic, conditional.

La comunicación entre agentes se hace mediante un intercambio de mensajes asíncrono. Cada agente tiene un 'mailbox' que funciona como un correo. (los mensajes también son objetos)

Estados: activo, iniciado, suspendido, en espera, en tránsito. Y podemos ir cambiando de estado con su correspondiente función.

****