Aprendizaje Profundo por Refuerzo

Puntos totales 15/15

Este test corresponde a la primera parte de la evaluación del curso Aprendizaje Profundo por Refuerzo, celebrado en la Escuela de Ciencias Informáticas 2019. La puntuación máxima que se puede conseguir es 8.5 (sobre 10 de la calificación final).

INSTRUCCIONES
El test consta de 15 preguntas de respuesta múltiple. Cada respuesta correcta suma 0.56 puntos. No hay penalización por las respuestas incorrectas. La duración del test será de 45 minutos. Puede abandonar el aula cuando finalice el test.

No está permitido el uso de teléfonos móviles, mensajería, correo electrónico o cualquier otro medio que permita comunicarse con otras personas. Sí se permite revisar el material del curso. El incumplimiento de estas normas significará la inmediata expulsión del estudiante de la prueba y una calificación de 0 en el curso.

Las respuestas estarán disponibles en este mismo formulario al terminar la prueba. Las calificaciones serán enviadas por correo electrónico. Por favor, no olvide introducir su dirección de correo electrónico.

Puntuación de la sección 15/15

Nombre *

Juan Gómez

Institución / Compañía

Comentarios

✓ 1. Las técnicas de aprendizaje por refuerzo se utilizan para problemas de: 1/1

- [x] Navegación de robots ✓
- [x] Juegos ✓
- [x] Planificación ✓
- [x] Control ✓

Comentarios

Todos los problemas mencionados pueden abordarse utilizando aprendizaje (profundo) por refuerzo. Ejemplos:

- Navegación de robots: En el curso hemos utilizado repetidamente el ejemplo del robot que se mueve por la cuadrícula. Aquí se puede encontrar una aproximación más compleja: https://www.nature.com/articles/nature20101.

- Juegos: El ejemplo motivador del curso ha sido AlphaGo. Hemos visto otros sistemas capaces de vencer a jugadores humanos expertos en otros juegos como el ajedrez (https://science.sciencemag.org/content/362/6419/1140.full?ijkey=XGd77kI6W4rSc&keytype=ref&siteid=sci), póker (https://science.sciencemag.org/content/early/2019/07/10/science.aay2400), DOTA (https://openai.com/five/), Quake (https://science.sciencemag.org/content/364/6443/859), etc.

- Planificación: La planificación puede verse como una generalización de los problemas que requieren descubrir la secuencia de pasos (óptima) que lleva a la solución. Por ejemplo, la navegación de robots o los juegos pueden verse como problemas de planificación. Otro problema que involucra planificación es la conducción automática (https://arxiv.org/abs/1905.02680).

- Control: Hemos estudiado algunos ejemplos de control en los entornos de OpenAI (cartpole, lunar lander, etc.) También hemos visto que los algoritmos de aprendizaje por refuerzo pueden aplicarse para control energético.

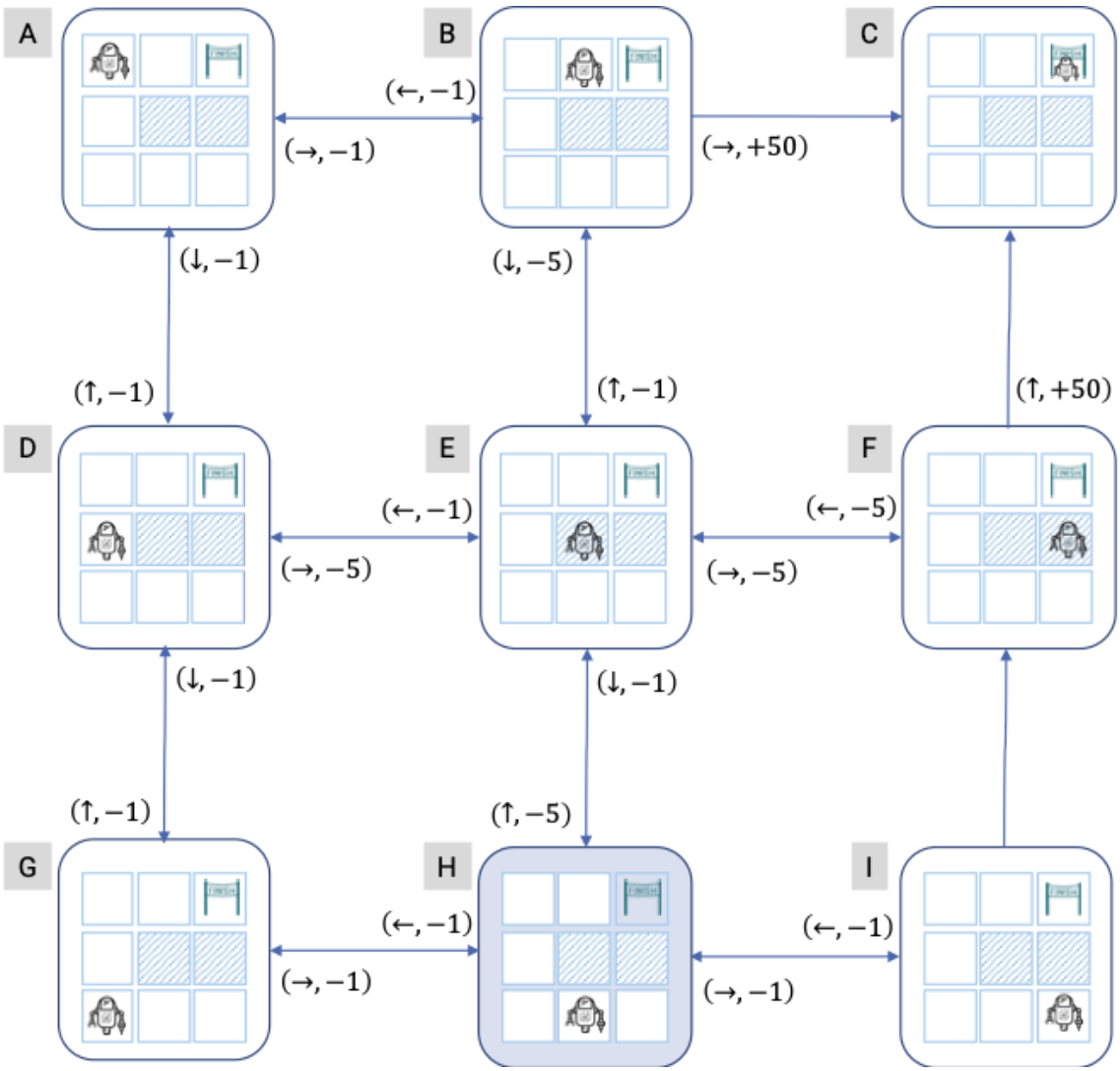
✓ 2. En un problema de decisión de Markov (MDP), la recompensa inmediata Rt+1 recibida por un agente después de realizar una acción At en el instante t: 1/1

- ○ Depende de las últimas acciones At-k, ..., At-2, At-1 (siendo k un parámetro)
- ○ Depende de las últimas acciones At-k, ..., At-2, At-1 (siendo k un parámetro) y del estado actual St del agente
- ○ Depende de las últimas acciones At-k, ..., At-2, At-1 (siendo k un parámetro), de la acción At y del estado actual St del agente
- ◉ Depende del estado actual St del agente y de la acción At ✓

Comentarios

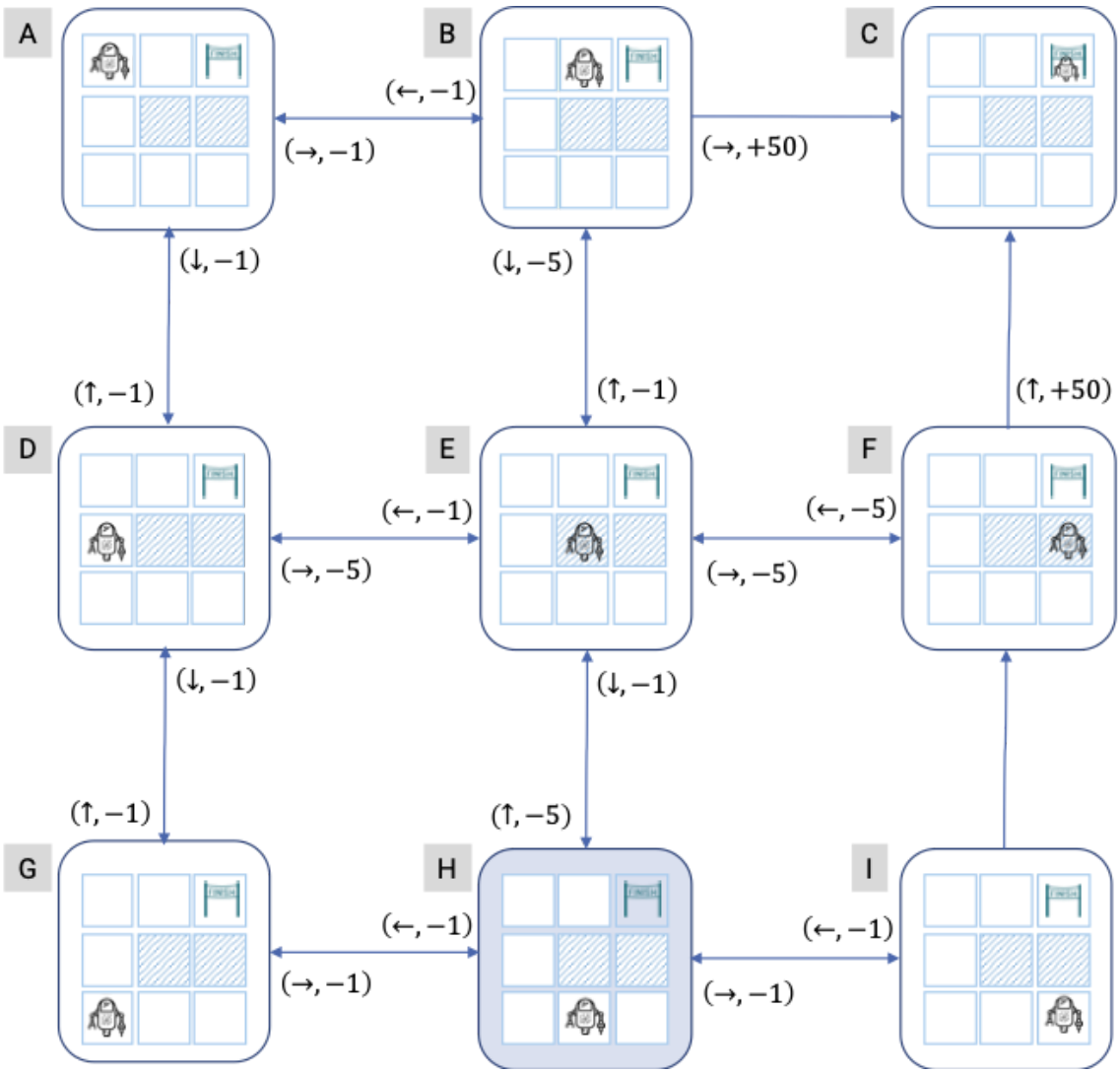
Una característica fundamental de un MDP es que la recompensa inmediata de un agente (Rt+1) solo depende del estado actual y de la acción realizada por el agente. La dinámica del MDP (Tema 3, p. 6) --o la visión simplificada en forma de diagrama de estados y transiciones-- reflejan precisamente este aspecto.

3. Dado el siguiente problema de decisión de Markov (MDP), ¿cuál sería la estrategia óptima? *



	Arriba	Derecha	Abajo	Izquierda	Puntuación	
A	○	◉	○	○	0/0	✓
B	○	◉	○	○	0/0	✓
D	◉	○	○	○	0/0	✓
E	◉	○	○	○	0/0	✓
F	◉	○	○	○	0/0	✓
G	◉	○	○	○	0/0	✓
H	○	○	○	◉	0/0	✓
I	◉	○	○	○	1/1	✓

✓ 4. Dado el siguiente problema de decisión de Markov (MDP), ¿cuál sería el valor de v(D) con la siguiente política? (Escriba solo el número) Política: El agente elige la primera acción posible de la lista: {Derecha, Arriba, Izquierda, Abajo}. Suponga un valor de descuento gamma = 1. * 1/1

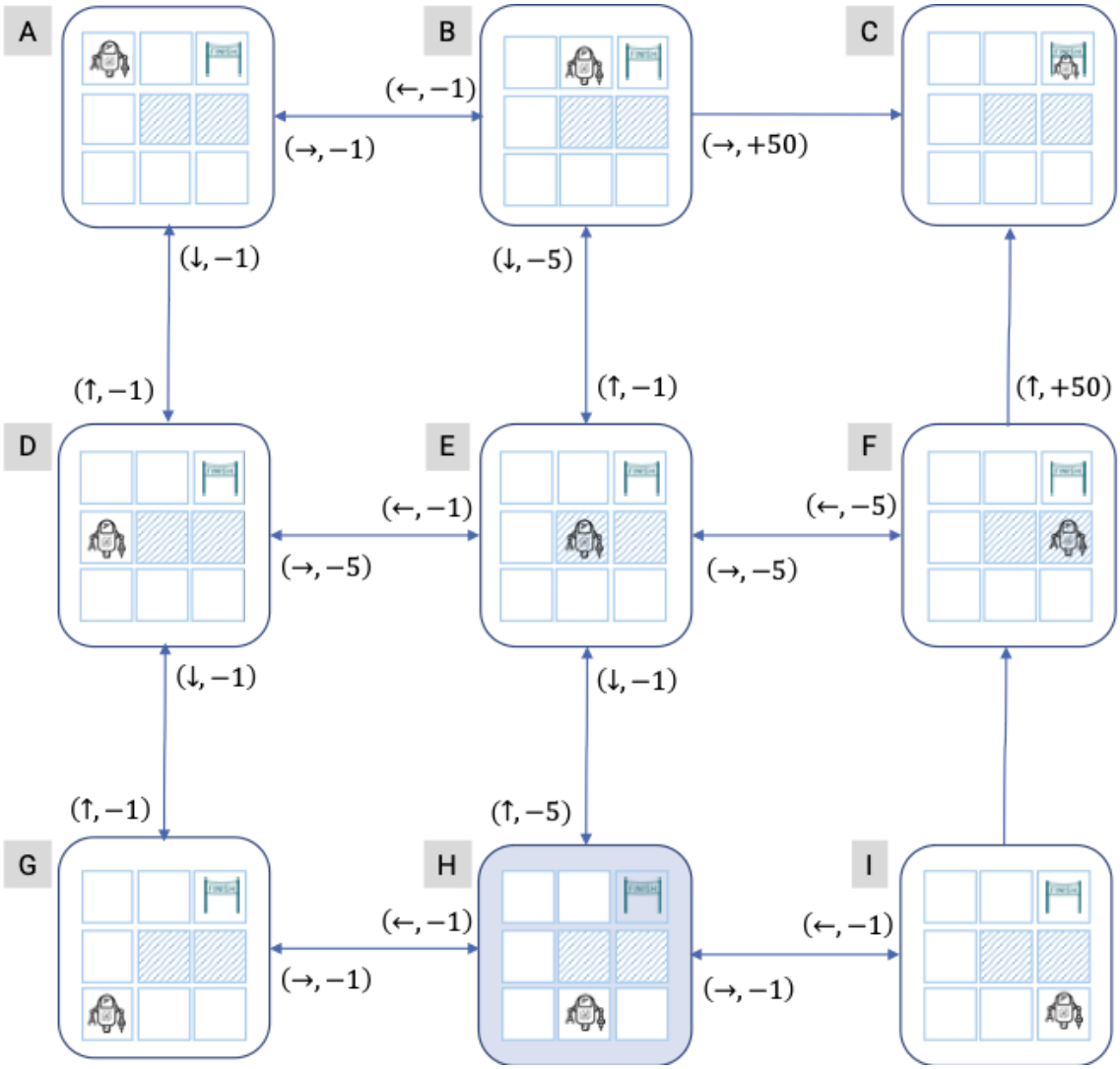


40

Comentarios

*Tema 3, p. 8. En este caso, v(D) = 1 * -5 + 1^1 * -5 + 1^2 * 50 = -5 -5 + 50 = 40*

✓ 5. Dado el siguiente problema de decisión de Markov (MDP), ¿cuál sería la recompensa acumulada Gt de un agente que, de acuerdo a su política de actuación, realiza el recorrido {D, E, B, C} --i.e. St = D? (Escriba solo el número) Suponga un valor de descuento gamma = 0.1. * 1/1

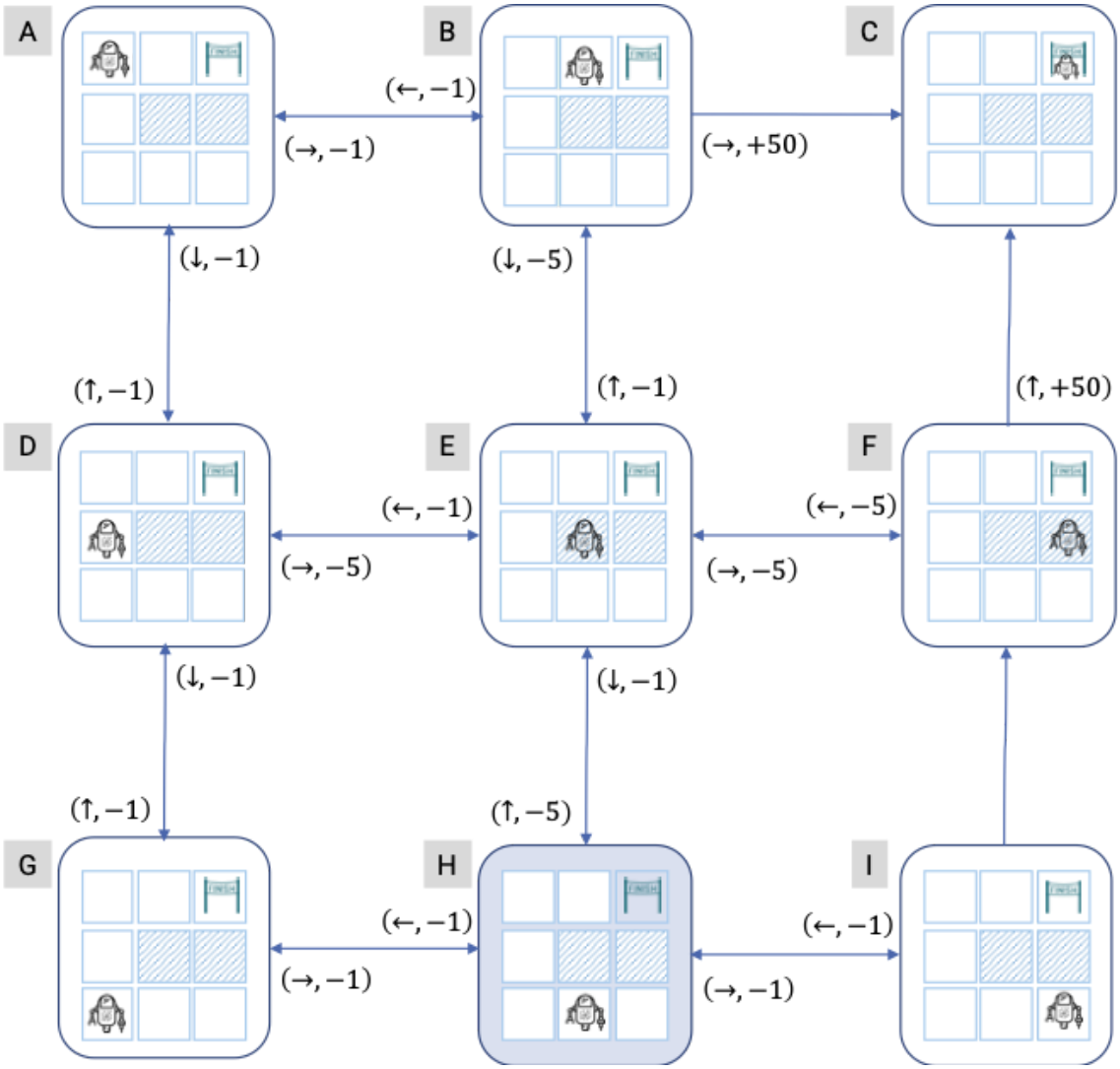


-4.6

Comentarios

*Tema 3, p. 7. En este caso, Gt = 0.1^0 * -5 + 0.1^1 * -1 + 0.1^2 * 50 = -5 + 0.1 * -1 + 0.01 * 50 = -5 -0.1 + 0.5 = -4.6*

✓ 6. Dado el siguiente problema de decisión de Markov (MDP), ¿cuál sería el valor de q(B, Abajo) con la siguiente política? Política: El agente elige la primera acción posible de la lista: {Derecha, Arriba, Izquierda, Abajo}. (Escriba solo el número) Suponga un valor de descuento gamma = 1. * 1/1



40

Comentarios

Tema 3, p. 8. En este caso, q(B, Abajo) = -5 -5 + 50 = 40. (El primer movimiento es "Abajo", a continuación se continúa utilizando la política.)

✓ 7. ¿En qué consiste el método de selección épsilon-greedy? 1/1

- [ ] Se selecciona siempre la mejor acción según la estimación épsilon
- [ ] Se selecciona siempre la mejor acción según la política actual
- [x] Se selecciona una acción aleatoria con probabilidad épsilon ✓
- [x] Se selecciona la mejor acción según la política actual con probabilidad (1-épsilon) ✓

Comentarios

El método ϵ-greedy (Tema 3, p. 19) establece que la acción a tomar desde un estado S data una tabla Q es:
- con probabilidad ϵ se toma una acción aleatoria
- con probabilidad $1-\epsilon$ se toma la acción de π (la que maximiza Q(S, a))

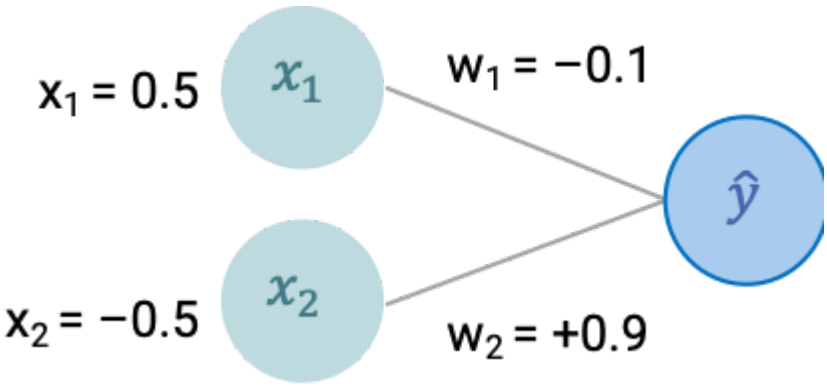
✓ 8. ¿Para qué sirve el método de selección épsilon-greedy? 1/1

- (●) Para aumentar el espacio de acciones explorado por el agente ✓
- ( ) Para reproducir el comportamiento de un agente humano
- ( ) Para asegurarse una recompensa mayor o igual a épsilon
- ( ) Para descontar la importancia de las recompensas estimadas futuras

Comentarios

El método ϵ-greedy (Tema 3, p. 19) se utiliza para diversificar la búsqueda durante el proceso de aprendizaje; esto es, para explorar secciones del espacio de búsqueda que utilizando la política greedy dada por Q no serían visitadas.

✓ 9. ¿Cuál sería la salida de la siguiente red neuronal, suponiendo función de salida ReLU en todas las neuronas (capa de entrada y capa de salida)? (Escriba solo el número) 1/1



0

Comentarios

*Tema 2, p. 5. La salida de la red neuronal sería 0: y = ReLU(ReLU(x1)*w1 + ReLU(x2)*w2) = ReLU(0.5*-0.1 + 0*0.9) = ReLU(-0.05) = 0.*

✓ 10. El método del gradiente descendente, utilizado para entrenar una red neuronal: 1/1

- [x] Modifica los pesos de la red en la dirección del gradiente de la función de error ✓
- [x] Utiliza la derivada de la función de error ✓
- [ ] Evita problemas de sobreaprendizaje
- [x] Es la base de algoritmos de optimización como RMSProp o Adam ✓

Comentarios

Tema 2, p. 14-15.

El método del gradiente descendente es un método de optimización iterativo que modifica sucesivamente los parámetros del modelo --los pesos de la red, en nuestro caso-- según el gradiente de la función de pérdida --el error entre la salida de la red y la salida esperada, en nuestro caso.

Los pesos se modifican en la dirección del gradiente (en sentido negativo, para descender en la función); o lo que es lo mismo, según la primera derivada de la función de error.

Adam y RMSProp son versiones avanzadas del algoritmo de gradiente descendente que utilizan tasas de aprendizaje adaptativas, entre otras mejoras.

✓ 11. En aprendizaje profundo podemos solucionar el sobreaprendizaje: 1/1

- ○ Aumentando el numero de capas de la red
- ◉ Seleccionando una muestra de datos para entrenamiento significativa ✓
- ○ Seleccionando una muestra de datos para entrenamiento ordenada por número de iteración
- ○ Aumentando el tamaño del conjunto de validación

Comentarios

El sobreaprendizaje es el fenómeno que se da cuando una red es incapaz de generalizar más allá de los datos que se han utilizado para el entrenamiento (Tema 2, p. 17-20). Entre las técnicas para limitar este fenómeno se incluye la selección adecuada y el aumento de la cantidad de datos disponibles para entrenamiento.

✓ 12. El método de aprendizaje Deep Q-Learning: 1/1

- ◉ Estima q(s, a) utilizando redes neuronales ✓
- ○ Estima p(a) utilizando redes neuronales
- ○ Estima q(s, a) y p(a) utilizando redes neuronales
- ○ Estima épsilon utilizando redes neuronales

Comentarios

Tema 4, p. 7. Deep Q-Learning es una extensión de Q-Learning. Utiliza redes neuronales para estimar la función estado-valor q (mediante aproximaciones Q modeladas mediante una red neuronal).

✓ 13. Los métodos de aprendizaje profundo por refuerzo basados en optimización de políticas, como REINFORCE: 1/1

- [x] Permiten trabajar con espacios de estados continuos ✓
- [x] Permiten trabajar con espacios de acciones continuos ✓
- [ ] Utilizan una red neuronal para estimar q(s, a)
- [ ] Utilizan una red neuronal para estimar v(s)

Comentarios

Tema 4, p. 17. Los métodos basados en optimización de políticas (policy-based) calculan directamente qué acción es mejor a partir de un estado, sin modelar explícitamente una aproximación a la función de acción-valor (q) o a la función de estado valor (v). REINFORCE, en concreto, permite además trabajar con espacios de estados y acciones continuos.

✓ 14. El entrenamiento de la red Q (action - valor de recompensa de acciones) en Deep Q-Learning se realiza: 1/1

- ○ Minimizando la diferencia entre la acción que devuelve la red y la mejor acción
- ○ Maximizando la diferencia entre la acción que devuelve la red y la peor acción
- ◉ Minimizando la diferencia entre la recompensa acumulada estimada por la red y la recompensa acumulada estimada para una muestra de la memoria de experiencias ✓
- ○ Maximizando la diferencia entre la recompensa acumulada estimada por la red y la recompensa acumulada calculada para una muestra de la memoria de experiencias

Comentarios

Tema 4, p. 14. Deep Q-Learning optimiza (mediante gradiente descendent) la función de pérdida [yj - Q(Sj, Aj; Theta)]^2, donde:

- yj es la recompensa acumulada estimada para un mini-batch de experiencias (basada en Rt + estimación de la red "target")

- Q(Sj, Aj; Theta) es la estimación de recompensa estimada por la red "action" (salida de la red)

✓ 15. La propuesta original del algoritmo Deep Q-Learning incluye los siguientes mecanismos para mejorar la convergencia de la red y encontrar antes soluciones: 1/1

- [x] Memoria de experiencias ✓
- [x] Actualización diferida de la red "target" Q ✓
- [ ] Soporte para espacios de estados continuos
- [ ] Soporte para espacios de acciones continuos

Comentarios

Tema 4, p. 14. El artículo original de Deep Q-Learning propone utilizar memoria de experiencias y actualización diferida de la red "target" (mediante "fusión" de pesos según el parámetro tau).

V. Mnih et al. (2015) Human-level control through deep reinforcement learning. Nature 518, 529-533

Posteriormente se han propuesto otras mejoras, como las enunciadas en Tema 4, p. 15. NOTA: El uso de dos redes ya aparece en el artículo original. Las aproximaciones Double DQN posteriores utilizan también dos redes Q, pero modifican la forma en que se utilizan en los pasos train/update del algoritmo.