|  |
| --- |
|  |
| Laboration 2 |
| Rapport |
|  |
|  |
|  |
|  |

**Innehåll**

1. Instruktioner 1
2. BridgeGrid (Value Iteration) 1
3. DiscountGrid (Value Iteration) 2
4. BridgeGrid (Q-Learning) 3

# Instruktioner

Denna rapporten hör till laboration 2 (MDPer och RL). I laborationsdokumnetet ombeds ni fylla i ett antal uppgifter i denna rapporten. När ni lämnar in laborationen skall denna ifyllda rapporten finnas med i arkivfilen.

# BridgeGrid (Value Iteration)

Byt ut värdet för antingen *discount* **eller** *noise* parametern i nedanstående tabell. Med det modifierade värdet skall value iteration agenten kunna följa den genererade policyn för att korsa bron och lämna Bridgeworld brädet via terminaltillståndet +10.00. Förklara också varför parameterinställning leder till önskad policy.

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Värde** | **Förklaring** |
| discount | 0.9 | Vi justerade endast noise parametern och då till ett mycket litet värde eftersom noise avgör hur ofta en agent hamnar i ett oavsiktligt tillstånd. |
| noise | 0.00001 |
|  | |

Uppdateringsformeln för value iteration återges nedan§

och är ekvivalent med

där

# DiscountGrid (Value Iteration)

Fyll i de tre parametervärdena *discount*, *noise* och *livingReward* (*reward*) som leder till önskad policy och förklara även varför dessa parameterinställningar leder till önskad policy.

|  |  |  |  |
| --- | --- | --- | --- |
| **Uppgift** | **Parameter** | **Värde** | **Förklaring** |
| a | discount | 0.3 | Med en låg discount (<0.3) blir agenten närsynt och tar ett tidigt positivt terminaltillstånd. Men också om man har en hög negativ living reward (-5) så vill agenten hitta ett tidigt postitivt terminaltillstånd, detta kräver också att discounten har ett värde (>0), annars kommer inte algoritmen att konvergera. |
| noise | 0.0 |
| reward | 0.5 |
|  | |
| b | discount | 0.3 | En låg discount (0.3) som gör att agenten väljer ett tidigt positivt terminaltillstånd före ett senare. Samt ett lågt värde av noise (0.3) för att agenten skall ta en säkrare väg för att undvika de negativa terminaltillstånden. |
| noise | 0.3 |
| reward | 0.0 |
|  | |
| c | discount | 0.4 | Här sätter vi discount till medel->högt värde (=>0.4) för att agenten skall välja det höga, långsiktiga positiva terminaltillståndet. |
| noise | 0.0 |
| reward | 0.0 |
|  | |
| d | discount | 0.0 | Här har vi samma argument som i uppgift C samt att vi sätter noise till 0.1 för att agenten skall ta den säkra vägen för att undvika att hamna i de negativa terminaltillstånden. |
| noise | 0.0 |
| reward | 0.0 |
|  | |
| e | discount | 0.0 |  |
| noise | 0.0 |
| reward | 0.0 |
|  | |

# BridgeGrid (Q-Learning)

Fyll i de två parametervärdena *epsilon* och *alpha* som leder till den optimala policyn under 50 träningsepisoder. Om den optimala policyn inte kan hittas med någon parameterkombination under 50 träningsepisoder, ange värdet *GÅR EJ* för de båda parametrarna. Motivera också erat svar, dvs varför hittas den optimala policyn med en viss parameterkombination eller varför går det inte att hitta den optimala policyn med någon parameterkombination?

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Värde** | **Förklaring** |
| epsilon | 0.0 | Om vi ställer epsilon högt (1) så väljs handling slumpmässigt och bron är helt enkelt tillräckligt lång för att agenten inte ska hitta till den högre belöningen (det är för många coinflips som måste bli rätt). Detta påverkas inte av learning rate. Väljer vi ett lågt (0) epsilon slutar agenten att utforska snabbt och väljer att casha in en liten belöning många ggr, detta oavsett hur låg learning rate vi sätter. Någon balans mellan dessa två parametrar som lyckas gå till den högre belöningen finns inte. |
| alpha | 0.0 |
| **GÅR EJ** | |