Apresentação slides conceitos

Slides 1

Basicamente esses são os conceitos que foram aprendidos no desenvolvimento do trabalho. Então vou explicar um pouco sobre aprendizagem por reforço o que é aprendizagem supervisionada e não supervisionada.

Equação de Bellman que é muito utilizado no g-larning. Será explicado o “Plano” que significa como o agente vai tomar as decisões para interagir no ambiente. Markov que basicamente vai adicionar uma camada mas complexa na equação de Bellman.

Politica x Plano a ideia é ver a diferença, então o agente pode ter um plano sair da origem até o objetivo ou ele pode ter uma politica.

Adição de Penalidade

E por fim o Q-larging que é um algoritmo clássico para trabalhar com aprendizagem por reforço

Slides 2

A Aprendizagem Por Reforço é o treinamento de modelos de aprendizado de máquina para tomar uma sequência de decisões. O agente aprende a atingir uma meta em um ambiente incerto e potencialmente complexo. No aprendizado por reforço, o sistema de inteligência artificial enfrenta uma situação.

Slides 3

Basicamente na equação serve pra levar o agente ao seu objetivo final. A equação acima afirma que o valor Q produzido por estar no estado s e selecionar a ação a é a recompensa imediata recebida, r (s, a), mais o valor Q mais alto possível do estado s’ (que é o estado em que chegamos depois de executar a ação a dos estados). Receberemos o valor Q mais alto de s’ escolhendo a ação que maximiza o valor Q. Também apresentamos γ, geralmente chamado de fator de desconto, que controla a importância das recompensas de longo prazo versus as imediatas.

Slides 4

O plano de ações é definido depois que o agente faz a equação de Bellam. O plano de ação é utilizado a equação então temos valores maiores quando está mais próximos do destino do agente(Mas 1 quando chegar as extremidades e -1 quando ficar em outras situações )

Slides 5

Cada estado em que o agente se encontra é uma conseqüência direta do estado anterior e da ação escolhida. O estado anterior também é uma consequência direta do que veio antes dele, e assim por diante até chegarmos ao estado terminal. Cada uma dessas etapas, e sua ordem, retém informações sobre o estado atual – e, portanto, têm efeito direto sobre qual ação o agente deve escolher a seguir. Mas há um problema óbvio aqui: quanto mais avançamos, mais informações o agente precisa para salvar e processar a cada passo necessário. Isso pode facilmente chegar ao ponto em que é simplesmente inviável realizar cálculos.

Então basicamente eu tenho um cenário é o agente vai para cima, então se ele decide ir para cima você vai ter 100% de que efetivamente ele vai fazer o movimento para cima, isso vai acontecer e não existe mas nenhuma opção, isso é busca determinística. E temos a busca não determinística que quando o agente decide ir para cima ele tem 3 opções se ele for pra cima ele tem 80% de chance, se ele ir para a esquerda tem 10% de chance, ou ele vai no fogo e também tem aquela porcentagem. Então isso é chamado de processo estocástico que vai demostrar o que está no ambiente real.

Para resolver isso, assumimos que todos os estados são Markov; isto é – assumimos que qualquer estado depende unicamente do estado que veio antes dele e da transição desse estado para o atual (a ação executada e a recompensa dada)

O MDP é o framework que o agente vai usar para entender o ambiente. Basicamente esse MDP podemos adicionar na equação de bellam.

O objetivo do nosso processo de decisão é encontrar uma política ótima π ∗ que maximiza a recompensa cumulativa descontada:

Slides 6

Politica: Basicamente seria uma estratégia que o agente faz para determinar uma ação, como percepção do seu estado atual, então ele faz o mapeamento dos estados para as ações, e as ações que prometem uma recompensa positiva.

Plano: O plano de ações é definido depois que o agente faz a equação de Bellam. O plano de ação é utilizado a equação então temos valores maiores quando está mais próximos do destino do agente. Então o plano é um caminho que você já sabe passo a passo

Então aqui temos uma versão mais avançada da equação de belman, então basicamente temos o gama que é o fator de desconto multiplicado pelo somatório da multiplicação da probabilidade de estar num estado pelo valor de v.

Slides 7(Penalidade de viver)

Para que a máquina faça o que o programador deseja, a inteligência artificial recebe recompensas ou penalidades pelas ações que executa. Seu objetivo é maximizar a recompensa total. Então basicamente na Adição de Penalidades e adicionar recompensas negativas em todos os estados. Isso quer dizer que quando mais o agente fica andando ele vai ganhando recompensas negativas.

Slides 8

Q-learning é um algoritmo de aprendizado por reforço sem modelo para aprender o valor de uma ação em um estado particular. Não requer um modelo do ambiente e pode lidar com problemas com transições estocásticas e recompensas sem exigir adaptações.

O que acontece quando o número de estados e ações se torna muito grande? Na verdade, isso não é tão raro – mesmo um jogo simples tem centenas de estados diferentes (tente calcular isso) e não se esqueça de multiplicarmos esse número por pelo o número de ações possíveis. Então, como vamos resolver problemas realmente complexos?

Uma solução possível é a Deep Q-Network! Combinamos Q Learning e Deep Learning, o que gera Deep Q-Networks. A ideia é simples: substituiremos a tabela de valores Q por uma rede neural que tente aproximar os valores Q. É geralmente referido como o aproximador ou a função de aproximação e indicado como Q (s, a; θ), em que θ representa os pesos treináveis da rede.

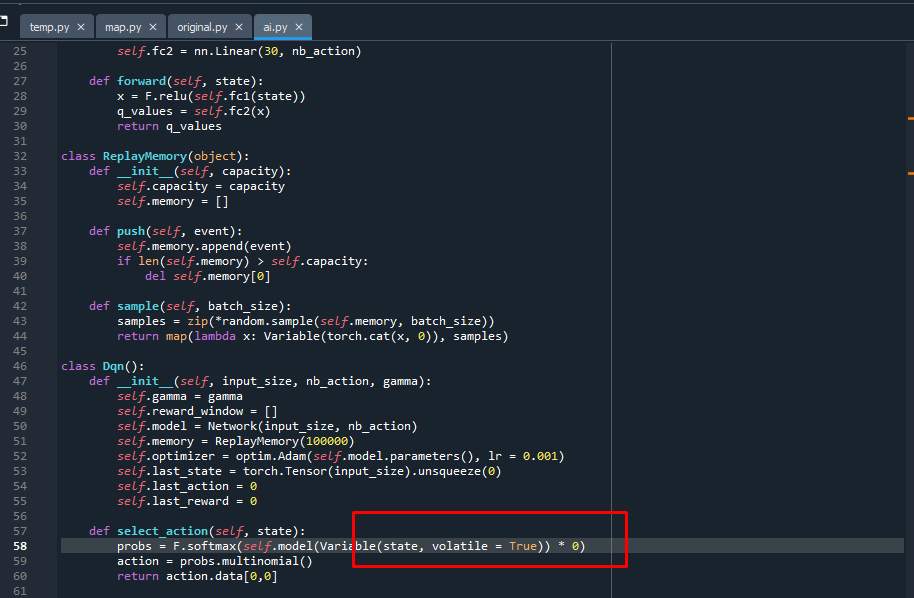
Apresentação dos codigos

Apresentação dos resultados

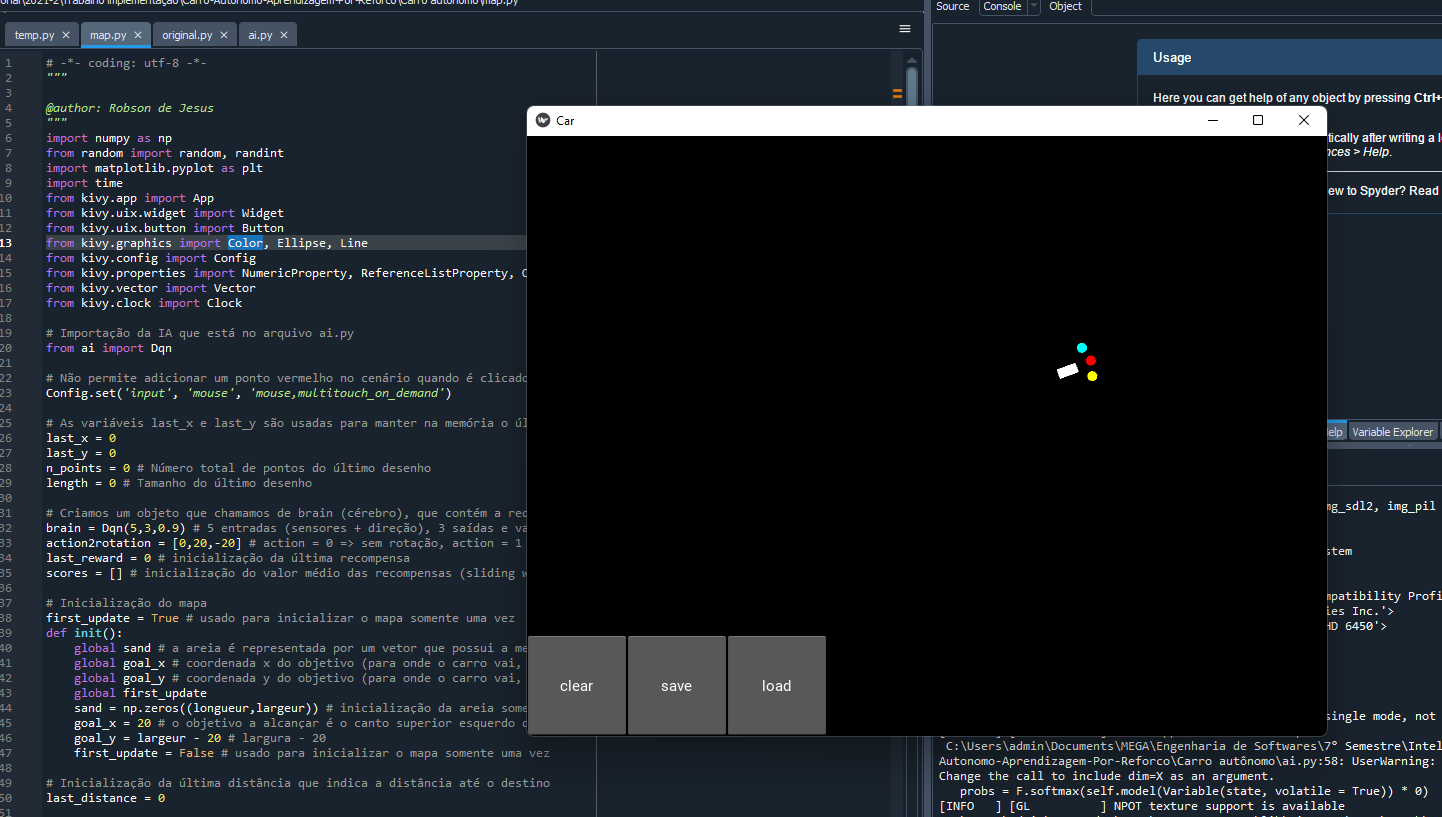
Será utilizado 4 níveis de dificuldade

1. Nível 1 não será colocado nenhum obstáculo.
2. Nível 2 desenhar uma estrada e esse carro deve se manter dentro da estrada.
3. Nível 3 desenhar objetos no mapa.
4. Nível 4 desenhar estrada mais complexa.
5. Nível 1

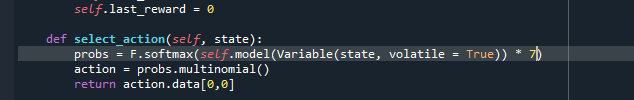
Primeiro teste sem utilizar a inteligência artificial. Mudança do valor da temperatura para zero



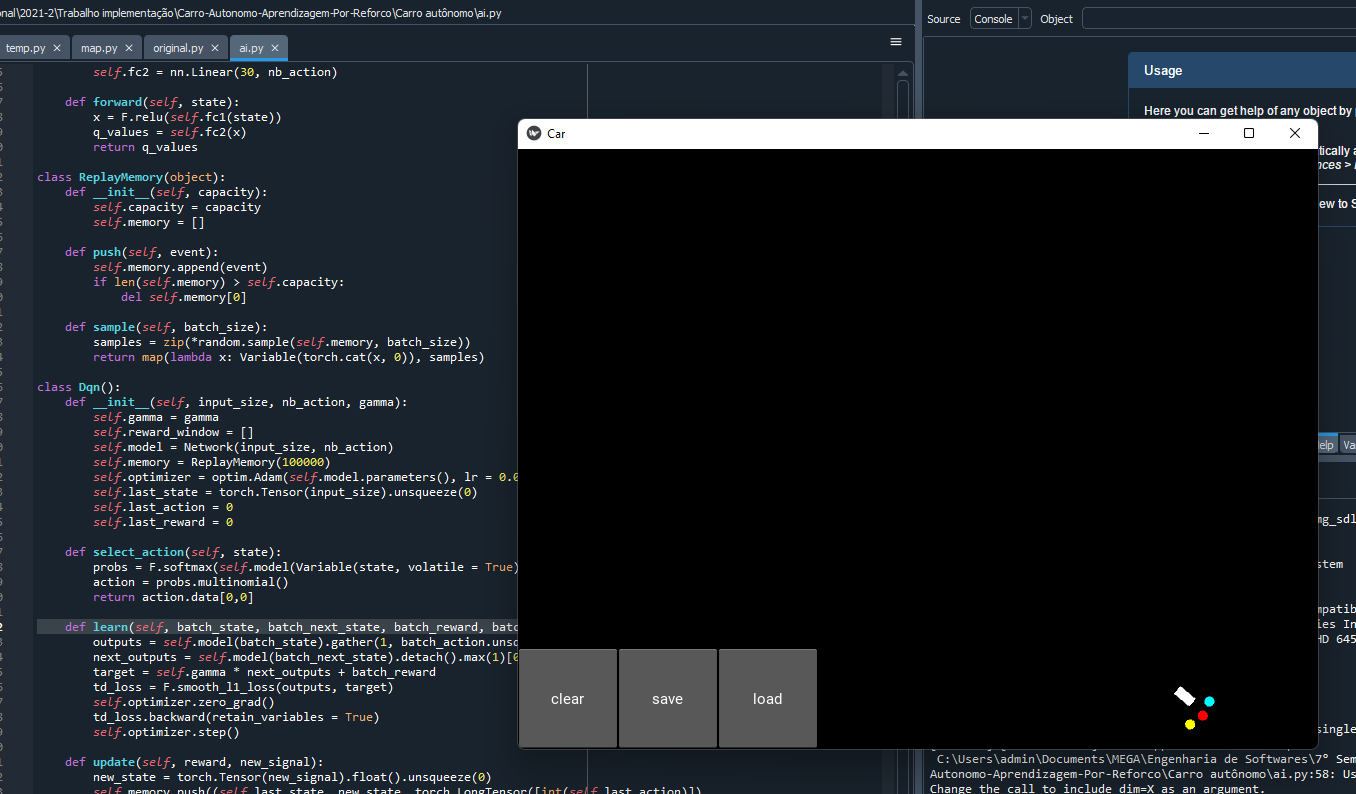
* Nesse tipo de simulação ele esta sem nenhum objetivo definido então ele fica andando para todos os lados como se fosse um inseto. Basicamente o objetivo principal é ir de uma lado para o outro.



Para realizar as viagens de uma ponta da extremidade para outra tem que mudar o valor do parâmetro para 7

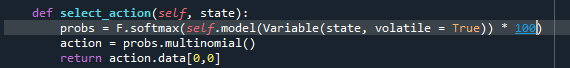


Assim podemos notar na simulação que ele tenta chegar de uma extremidade para outra quando é mudado esse parâmetro.

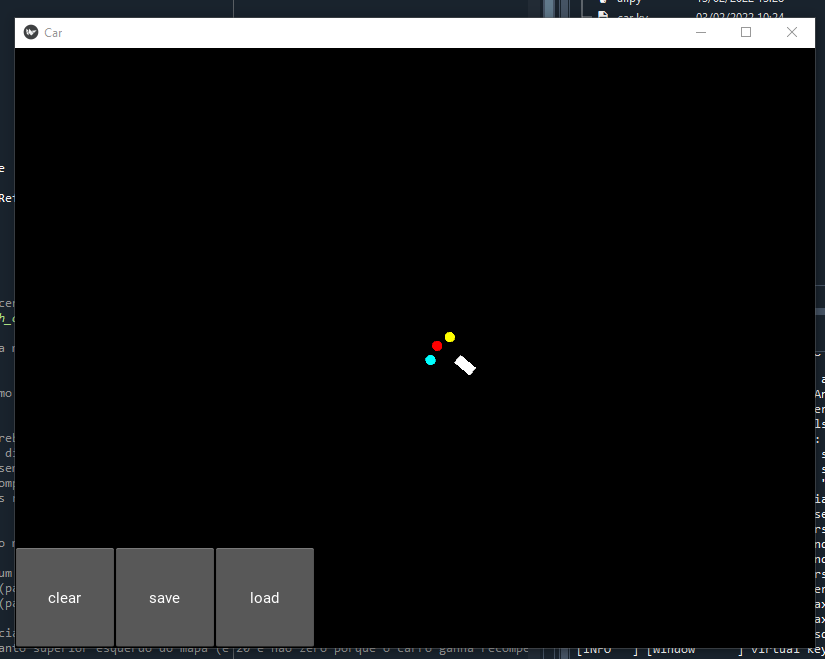


Então basicamente ele nessa situação já está no nível 2. Nesse caso como não a obstáculos de areia na pista as únicas punições é quando ele chega nas bordas e também ele tem a punição quando ele está longe do objetivo.

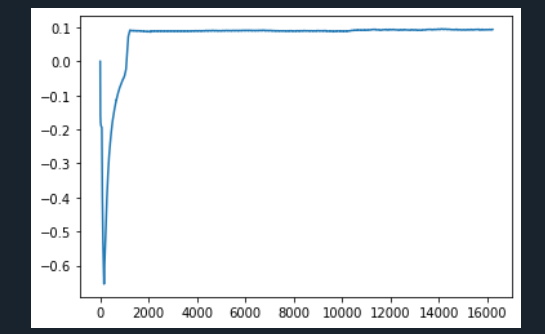
Realizando outro teste caso você queira deixar o carro com menos movimentos. Por que atualmente com o parâmetro 7 ele parece que não está muito confiante nos movimentos que ele esta tomando. Altere o valor da temperatura para 100



Basicamente agora o carro está mas confiante nesse movimentos, isso acontece por que o parâmetro da temperatura ele é um parâmetro para a função softmax retornar os dados com mais confiabilidade ou seja a inteligência artificial será mais confiança sobre a ação a ser tomada. É importante também dizer se esse parâmetro for muito aumentado pode atrapalhar o carro de explorar o ambiente.



Salve o modelo

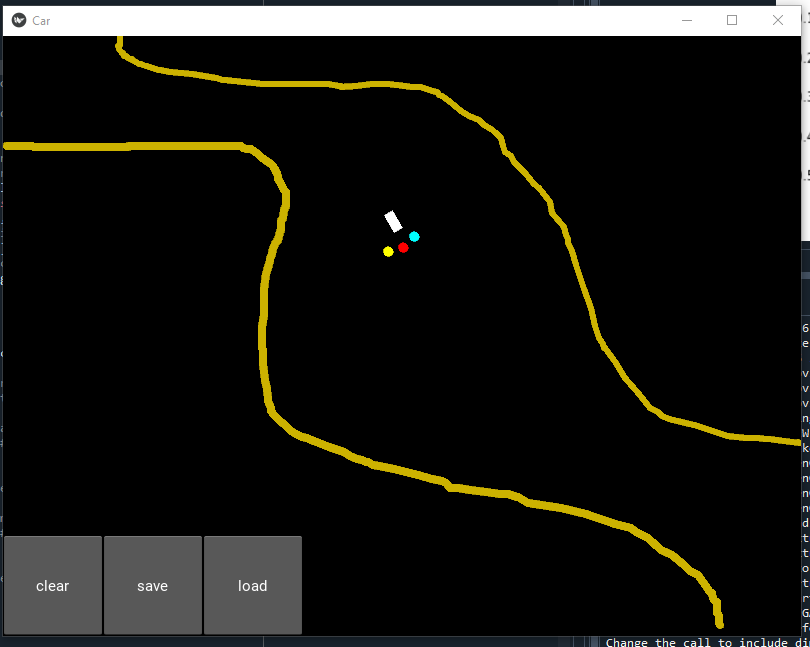


Se for analisado na função save ele faz um plot na variável scores. Lembrando que está variável score que vai calcular os valores da recompensa, que é a media das 1000 ultimas recompensas.

Então nota-se no gráfico que ele começa com o valor de recompensa negativo e depois ele vai chegando numa recompensa positiva de 0.10 que é o maior valor de recompensa positiva definido.

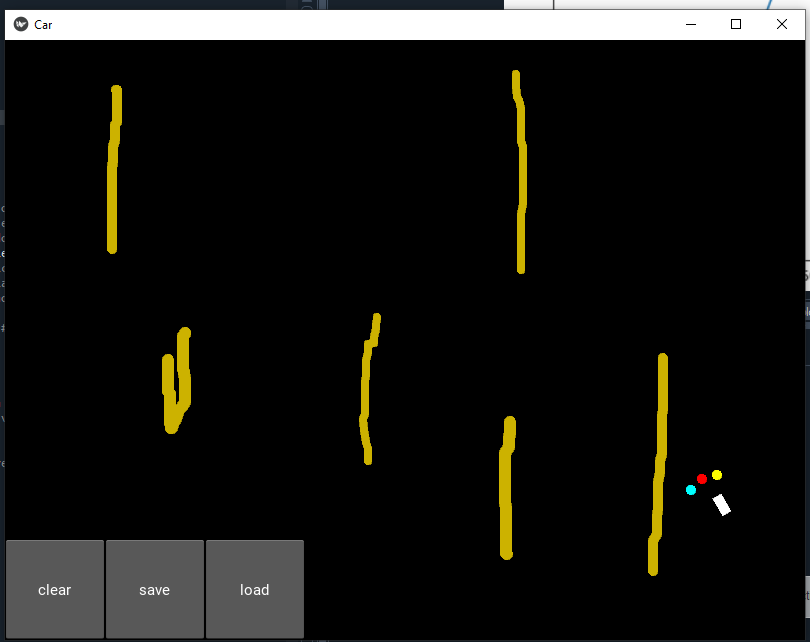
1. Nível 2

Execute o mapa e aperte em load para carregar o modelo anterior, seja que quando você faz esse carregamento ele já consegue fazer esta roda do nível 1. Agora o objetivo no nível 2 e verificar se ele vai conseguir andar dentro de uma estrada. Desenho a estrada. Facilmente podemos notar que esse agente conseguiu passar para o nível 2 dentro dessa estrada.



1. Nível 3

Nesse teste vai ser colocado alguns obstáculos, no meio do mapa. Pode-se notar que ele também consegue se adaptar a esse novo ambiente. Então ele já passou para o nível 3.



1. Para o agente conseguir passar no nível 4 tem que mudar a estrutura da rede neural

Pode-se notar que o agente não se adapta muito bem ao ambiente colocado na pista

Então basicamente para o agente conseguir se adaptar a esse tipo de ambiente deve-se mudar a estrutura da rede neural. Então tem que ter uma melhoria do código fonte tem que mudar a estrutura da rede. Temos uma rede neural bastante simples que tem somente uma camada oculta. Então se buscar na teoria de deep larning é considerado uma deep larning quando tem pelo menos duas camadas ocultas. Então basicamente pode-se adicionar mas camadas ocultas, adicionar mais a quantidade de neurônios e verificar como vai ficar esse desempenho.

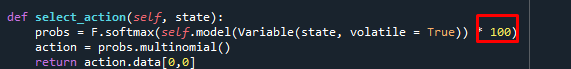
Coisas para se alterar:

1 – Adicionar mais camadas



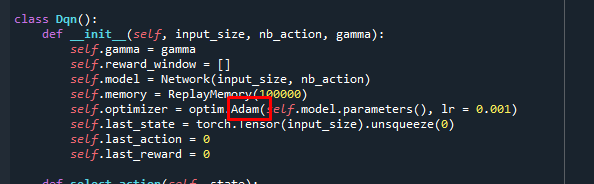
2 – Pode aumentar a quantidade de neurônios

3 – Pode-se alterar o parâmetro da temperatura



4 – Pode usar outras funções de erro

5 – Pode utilizar outros tipos de utilizadores (é utilizado o Adan, mas também tem o rms)



6- Com relação ao mapa pode dentar diminuir os valores das recompensas.

