TRAIN A SMARTCAB TO DRIVE

Módulo de Aprendizagem por Reforço
Nanodegree Engenheiro de Machine Learning Udacity

Tarefas

TAREFA 1:

Implementar um Agente Condutor Básico

Para começar, sua única tarefa é fazer o táxi inteligente movimentar-se ao redor do ambiente. Nesse momento, você não deverá se preocupar com nenhum política de optimização de condução. Note que o agente condutor está dando as seguintes informações a cada intersecção:

- O próximo ponto de navegação é relativo a sua própria localização e direção.
- O estado do semáforo na intersecção e a presença de veículos em direção contrária vindo de outras direções.
- O atual tempo que sobrou do prazo estipulado. Para completar essa tarefa, simplesmente faça com que seu agente condutor escolha uma ação qualquer do conjunto de ações possíveis (None , 'forward' ,'left' , 'direita') a cada intersecção, desconsiderando a informação de entrada acima. Defina a simulação de execução do prazo final, enforce_deadline para False e observa como ele executa

TAREFA 2:

Informar o Agente condutor

Agora que seu agente condutor é capaz de se movimentar pelo ambiente, sua próxima tarefa é identificar um conjunto de estados que são apropriados para a modelagem do táxi inteligente e o ambiente. A fonte principal de variáveis de estado são as atuais entradas na intersecção, mas nem todas podem precisar de representação. Você pode optar por definir explicitamente os estados ou usar algumas combinações de entrada como um estado implícito. A cada passo de tempo, processe as entradas e atualize o estado atual do agente utilizando a variável self.state . Continue com a obrigação do prazo da simulação enforce_deadline sendo ajustado para False e observe como seu agente condutor agora reporta a mudança de estado ao progredir da simulação.

TAREFA 3:

Implementar um Agente Condutor Q-Learning

Com o seu agente condutor bem capacitado para interpretar as informações de entrada e ter um mapeamento dos estados do ambiente, sua próxima tarefa é implementar um algoritmo QLearning para o seu agente condutor escolher a melhor ação a cada passo, baseado nos Qvalues do estado e ação atuais. Cada ação tomada pelo táxi inteligente vai produzir uma recompensa que dependerá do estado do ambiente. O agente condutor QLearning vai precisar considerar essas recompensas ao atualizar os Qvalues. Uma vez implementado, ajuste a obrigação do prazo da simulação enforce_deadline para True . Execute a simulação e observe como os táxis inteligentes se movimentam no ambiente em cada tentativa.

TAREFA 4:

Melhorar o Agente Condutor Q-Learning

Sua última tarefa para este projeto é melhorar seu agente condutor até que,depois de treinado o suficiente, o táxi inteligente seja capaz de alcançar o destino dentro do tempo alocado de maneira segura e eficiente. Parâmetros do algoritmo do QLearning, como a taxa de aprendizagem (alpha), o fator de desconto (gamma) e a taxa de exploração (epsilon), todos contribuem para a habilidade do agente condutor de aprender a melhor ação para cada estado. Para melhorar o sucesso do seu táxi inteligente:

- Ajuste o número de tentativas, n_trials , na simulação para 100.
- Execute a simulação com obrigação do prazo enforce_deadline , ajuste para True (você vai precisar reduzir o atraso da atualização update_delay e ajuste o display para Falso).
- Observe a aprendizagem do agente condutor e a taxa de sucesso do táxi inteligente particularmente durante os últimos testes.
- Ajuste um ou vários dos parâmetros acima e itere esse processo. Esta tarefa estará completa uma vez que você tenha chegado naquilo que você determinou como melhor combinação de parâmetros requisitados para que o agente condutor aprenda com sucesso.

PERGUNTAS

PERGUNTA 1:

Observe o que você vê do comportamento do agente ao realizar ações aleatórias. O táxi inteligente eventualmente chega ao seu destino? Há outras observações interessantes a serem feitas?

Resposta:

Percebi que o táxi parece ser dirigido por uma pessoa sem habilitação, já que ele não chega ao seu destino, onde ele não chega ao seu destino por se envolver em algum tipo de acidente ou cometer infrações. Percebi que para as 100 tentativas nenhuma delas chega ao seu destino.

Uma curiosidade que achei interessante e que se você reduzir o delay do modelo ele roda mais rápido, o que possibilita fazer melhores testes no modelo, já que o delay em 0.5 demora muito para rodar e você fica dependendo de esperar.

PERGUNTA 2:

Quais estados definidos por você são apropriados para modelar o táxi inteligente e o ambiente? Por que você acredita que cada um desses estados são apropriados para esse problema?

Resposta:

Acredito que a direção da viagem seja um dado muito importante sobre esse modelo. A direção da viagem (next_waypoint) é uma variável muito importante para esse modelo, a direção do próximo waypoint informa a melhor maneira para o smartcab se comportar. Sem esse dado seria impossível ensinar para o modelo se o smartcab deve virar a direita ou esquerda.

A cor do sinal é muito importante porque ela define se um smartcab deve seguir na vida ou não, saber diferenciar se a luz e verde ou vermelha vai ser uma implementação muito importante para esse modelo. O smartcab deve estar ciente que avançar sinais vermelhos é uma infração grave que pode levar ao acidente fazendo com que dois smartcabs não cheguem ao seu destino.

Uma coisa interessante sobre esse modelo é que não deve ser levado em consideração o prazo para a chegada no destino, afinal, independente de quanto tempo houver a ideia é fazer o smartcab se mover respeitando as regras de trânsito. Um smartcab que quebre regras do trânsito chegar rápido ao seu destino se assemelha aos seres humanos é a ideia de um carro autônomo e ser melhor.

É necessário saber o status de todos os carros na interseção. O principal e saber sobre os carros da esquerda. Não consigo pensar em um cenário em que nos preocupamos com carros à direita para esse modelo específico.

É necessário saber o status de todos os carros nos cruzamentos porque podemos definir ações para esse smartcab, assim como quando fazemos auto escola aprendemos que alguns veículos têm prioridade com relação a outros, também existem regras quando um veículo está na esquerda ou na direita e sobre qual decisão deverá tomar.

PERGUNTA 3:

Reporte os valores diferente para os parâmetros sintonizados em sua implementação básica de QLearning. Para quais conjuntos de parâmetros o agente melhor se desempenha? Quão bom é o último desempenho do agente condutor?

Resposta:

Durante o processo de criação do modelo, eu fiz diversos testes, tirando e colocando variáveis, aumentando e diminuindo os valores, e após executando o agent.py para perceber quais caracteristicas melhor se enquandram para o modelo. Abaixo vou registrar todas as tentativas após eu ter chegado a um modelo ideal de variáveis para esse modelo, separei aqui os principais testes.

Primeiro report do projeto.

- self.state = {}
- self.learning_rate = 0.5
- self.exploration_rate = 0.8
- self.exploration_degradation_rate = 0.0
- self.discount_rate = 0.8
- self.q_values = {}
- self.valid_actions = [None, 'forward', 'left', 'right']
- self.total_wins = 0
- self.trial_infractions = 0
- self.infractions_record = []
- self.trial_count = 0
- self.epsilon_annealing_rate = .01
- self.episilon_reset_trials = 200

Primary agent has reached destination! = 69 | Primary agent ran out of time! Trial aborted = 31

Segundo report do projeto.

- self.state = {}
- self.learning_rate = 0.5

- self.exploration_rate = 0.8
- self.exploration_degradation_rate = 0.0
- self.discount_rate = 0.8
- self.q_values = {}
- self.valid_actions = [None, 'forward', 'left', 'right']
- self.total_wins = 0
- self.trial_infractions = 0
- self.infractions_record = []
- self.trial_count = 0
- self.epsilon_annealing_rate = .01
- self.episilon_reset_trials = 200

Primary agent has reached destination! = 25 | Primary agent ran out of time! Trial aborted = 74

Terceiro report do projeto.

- self.state = {}
- self.learning_rate = 0.6
- self.exploration_rate = 0.1
- self.exploration_degradation_rate = 0.001
- self.discount_rate = 0.4
- self.q_values = {}
- self.valid_actions = [None, 'forward', 'left', 'right']
- self.total_wins = 0
- self.trial_infractions = 0
- self.infractions_record = []
- self.trial_count = 0
- self.epsilon_annealing_rate = .01
- self.episilon_reset_trials = 200

Primary agent has reached destination! = 95 | Primary agent ran out of time! Trial aborted = 5

Reportação final do projeto.

- self.state = {}
- self.learning_rate = 0.4
- self.exploration_rate = 0.1
- self.exploration_degradation_rate = 0.001
- self.discount_rate = 0.2
- self.q_values = {}
- self.valid_actions = [None, 'forward', 'left', 'right']
- self.total_wins = 0

- self.trial_infractions = 0
- self.infractions_record = []
- self.trial_count = 0
- self.epsilon_annealing_rate = .01
- self.episilon_reset_trials = 200
- trials = 100

Primary agent has reached destination! = 98 | Primary agent ran out of time! Trial aborted = 2

PERGUNTA 4:

Seu agente se aproxima de encontrar uma política ótima, por exemplo,chegar ao destino no tempo mínimo possível e sem incorrer nenhuma penalidade? Como você descreveria uma política ótima para este problema?

Resposta:

Uma política ótima para esse problemas seria um modelo onde o agente não se envolve em acidentes e chegue no destino no tempo correto, para criar um agente sem o uso de técnicas de aprendizagem por reforço seria basicamente criar um modelo ensinado por um instrutor de auto escola, onde o agente teria opções de movimento pré definidas. Dessa forma acredito que cheguei a um modelo com uma política interessante.

A maior preocupação ao se treinar um smartcab e não violar a lei, se envolver em acidentes e chegar na hora em seu objetivo, se o modelo conseguir chegar 98% das vezes no tempo certo sem infringir leis de trânsito e se acidentar acredito que o modelo seja adequado. Acredito que a melhor política para esse modelo seria um acerto de 100%, é claro se for possível de chegar a perfeição para um modelo de carro autônomo.