Trabalho prático 2 - Inteligência Artificial DAS5341

Universidade Federal de Santa Catarina
Departamento de Automação e Sistemas - 2020/01
Prof. Eric Aislan Antonelo

Opção 1 - Treinamento de redes multi-camadas

Neste trabalho, você deverá implementar o método da retropropagação (*backpropagation*) para o cálculo do gradiente da função de custo com relação aos pesos de uma rede neural multi-camadas (número de camadas ocultas $n_o \geq 2$).

(a) Implemente o algoritmo da retropropagação e o método do descenso do gradiente para o treinamento da rede neural, usando a biblioteca *numpy*.

(b) Valide o algoritmo do cálculo do gradiente, realizando uma aproximação numérica do mesmo.

(c) Treine a rede para um conjunto de dados para classificação de padrões (amostras de Gaussianas), onde a rede é mais complexa do que o necessário. Use regularização para suavizar a resposta da rede. O número de variáveis n pode ser 2. Defina o número de exemplos de treinamento (e.g. $m = 30$), de modo que a regularização seja necessária. Encontre o melhor parâmetro de regularização usando validação cruzada e uma métrica de avaliação.

(d) Crie um conjunto de dados altamente não-linear e treine uma rede neural para classificar os padrões do mesmo. Use validação cruzada e uma métrica de avaliação para escolher a melhor arquitetura da rede (número de camadas, e neurônios).

Opção 2 - Clonagem comportamental

Neste trabalho, você usará o simulador CARLA para Navegação Autônoma de Veículos em Ambientes Urbanos.

Aprendizagem por imitação é uma abordagem pela qual um modelo caixa-preta (rede neural) é treinado para imitar um especialista usando um conjunto fixo de amostras de pares observação-ação (ou trajetórias) obtidas daquele especialista. A clonagem comportamental (CC) é um tipo de aprendizagem por imitação baseada em um processo de treinamento supervisionado de um modelo (rede neural) usando um grande conjunto de dados rotulados. A CC tem sido utilizada para a obtenção de políticas de condução autônoma para veículos, onde as amostras de treinamento são geradas por motoristas humanos: a entrada da rede neural é a imagem da câmera do carro, enquanto a saída desejada corresponde ao atuador (ação do motorista).

Mais detalhes sobre CARLA, rede neural e aprendizado por imitação (ver seções 3.2 Imitation learning e S.2.2 Imitation Learning): `http://proceedings.mlr.press/v78/dosovitskiy17a/dosovitskiy17a.pdf`

Requisitos: deverá ter um computador potente o suficiente para treinar uma rede neural convolucional com um conjunto de imagens obtidas do simulador CARLA.

(a) Projete e treine uma rede convolucional, usando *pytorch* ou *tensorflow/keras*, somente para seguir (em frente) a pista atual no simulador.

(b) Treine uma rede convolucional para dar a volta em um quarteirão (bloco) infinitamente.

(c) Crie uma outro tipo de comportamento desejado ou outra arquitetura de rede neural e apresente os resultados.

Explique sua abordagem, métodos utilizados, arquitetura da rede, e apresente os gráficos da função de custo no conjunto de treinamento. Use uma métrica para avaliar os agentes (rede neural) durante a navegação (colisões, tempo de condução sem colisões, etc.).

Para gerar exemplos de treinamento, poderá usar *auto-pilots* já existentes no simulador ou algum repositório.

Há muitos códigos no github relacionados a este assunto que poderão ser consultados para inspiração no desenvolvimento desse trabalho.

Links:
- `https://github.com/carla-simulator/carla`
- `https://github.com/sunnahwalker/Carla-Behavioral-Cloning`