Instituto Tecnológico de Aeronáutica – ITA
Inteligência Artificial para Robótica Móvel – CT-213
Professor: Marcos Ricardo Omena de Albuquerque Maximo

Laboratório 7 (Aula 8) – *Imitation Learning* com Keras

1. Introdução

Nesse laboratório, seu objetivo é copiar um movimento de caminhar de um robô humanoide usando uma técnica chamada *imitation learning*. Para isso, você usará o *framework* de Deep Learning Keras. A Figura 1 mostra uma comparação entre o movimento da junta do quadril de arfagem obtida através de "observação" da caminhada de um robô humanoide e o mesmo movimento conforme "copiado" por uma rede neural.

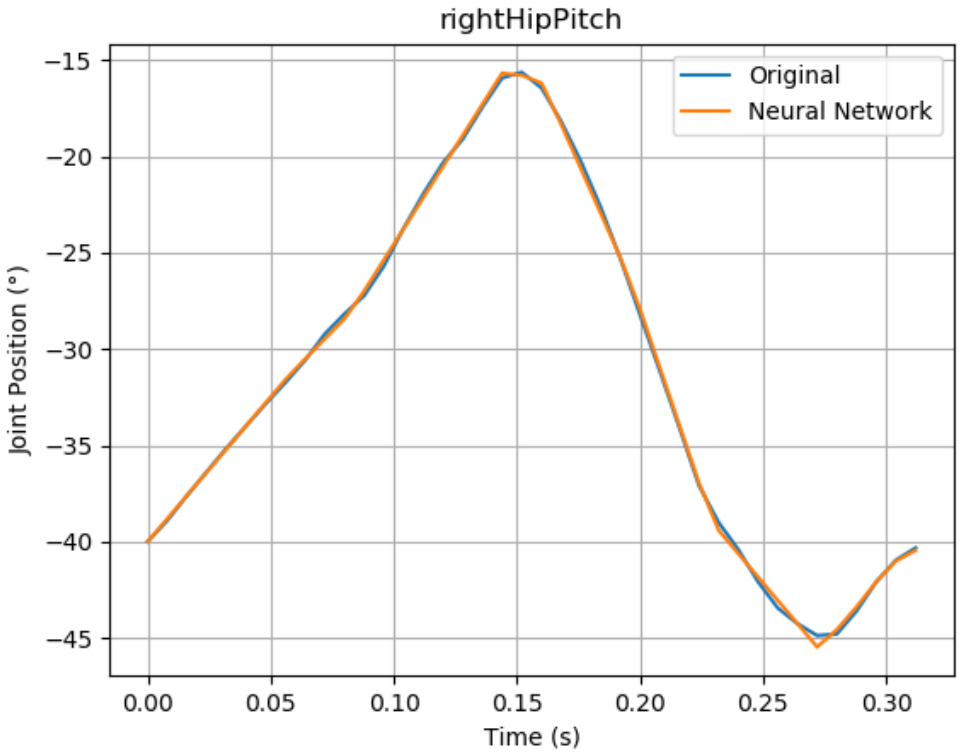

Figura 1: imitação do movimento da junta do quadril de arfagem através de uma rede neural.

2. Descrição do Problema

O problema a ser resolvido nesse laboratório é copiar o movimento de caminhar de um robô humanoide usando uma técnica chamada *imitation learning*. Nessa técnica, copia-se uma política de controle usando aprendizado supervisionado. Para a implementação da rede neural, você utilizará o *framework* Keras, que facilita o uso do *framework* Tensorflow.

O exemplo de treinamento consiste nas posições angulares das 20 juntas do robô durante um ciclo de caminhada, que foram obtidas através da “observação” das juntas do robô enquanto ele caminhava. O algoritmo de caminhada original é baseado em teoria de Controle. A entrada da rede é o tempo dentro do ciclo de caminhada (apenas 1 entrada), enquanto a saída consiste nas posições das 20 juntas. Sugere-se que a rede a ser implementada deve seguir a arquitetura apresentada na Tabela 1.

Layer	**Neurons**	**Activation Function**
Dense	75	Leaky ReLU ($\alpha = 0,01$)
Dense	50	Leaky ReLU ($\alpha = 0,01$)
Dense	20	Linear

Tabela 1: arquitetura da rede neural usada para o *imitation learning*.

3. Código Base

O código base já implementa um exemplo para familiarização com o Keras, que encontra-se no arquivo `test_keras.py`. Então, você deve implementar a rede neural usada para o *imitation learning* seguindo mais ou menos as mesmas ideias (há dicas no final do roteiro para te ajudar caso ainda tenha dificuldades).

4. Tarefas

4.1. Estudo de Implementação de Rede Neural com Keras

Primeiramente, estude o script `test_keras.py` para aprender como implementar uma rede neural com Keras. Sim, o Keras é muito fácil de usar! Isso que fez essa *framework* ser tão popular.

4.2. Análise do Efeito de Regularização

No script `test_keras.py`, a variável lambda_l2 representa o parâmetro λ da regularização L_2. Treine as redes para as duas funções de classificação diferentes (`sum_gt_zero` e `xor`) e considerando $\lambda = 0$ (sem regularização) e $\lambda = 0,002$. Compare os resultados obtidos e discuta o efeito da regularização. Coloque os gráficos gerados no seu relatório.

4.3. Imitation Learning

Usando Keras, implemente uma rede neural no *script* `imitation_learning.py` de acordo com a arquitetura apresentada na Tabela 1. Leve em consideração o seguinte para o treinamento:

- Não use regularização.

- Use todo o *dataset* em cada iteração do treinamento.
- Use erro quadrático como *loss*.
  Coloque os gráficos gerados no seu relatório e discuta os resultados.

5. Entrega

A entrega consiste do código e de um relatório, submetida através do Google Classroom. Modificações nos arquivos do código base são permitidas, desde que o nome e a interface dos scripts "main" não sejam alterados. A princípio, não há limitação de número de páginas para o relatório, mas pede-se que seja sucinto. O relatório deve conter:

- Breve descrição em alto nível da sua implementação.
- Figuras que comprovem o funcionamento do seu código.

Por limitações do Google Classroom (e por motivo de facilitar a automatização da correção), entregue seu laboratório com todos os arquivos num único arquivo **.zip** (**não** utilize outras tecnologias de compactação de arquivos) com o seguinte padrão de nome: "**<login_email_google_education>_labX.zip**". Por exemplo, no meu caso, meu login Google Education é **marcos.maximo**, logo eu entregaria o lab 7 como "**marcos.maximo_lab7.zip**". **Não** crie subpastas para os arquivos da sua entrega, **deixe todos os arquivos na "raiz" do .zip**. Os relatórios devem ser entregues em formato **.pdf**.

6. Dicas

- Em `keras.activations`, não há função de ativação Leaky ReLU. Para usar Leaky ReLU no Keras, você tem que adicionar uma camada do tipo `LeakyRelu` após ter definido uma camada (usando função de ativação linear). Assim, uma camada com função de ativação Leaky ReLU é definida em Keras da seguinte forma:

```
model.add(layers.Dense(num_neurons, activation=activations.linear))
model.add(layers.LeakyReLU(alpha))  # alpha is the Leaky ReLU parameter
```

- Conforme está mostrado no exemplo, na primeira camada da rede, é importante definir qual é a dimensão da entrada.
- Para usar otimizar os parâmetros da rede usando Adam e função de custo quadrática, faça:

```
model.compile(optimizer=optimizers.Adam(), loss=losses.mean_squared_error)
```

- Para usar todos os dados de treinamento durante o treinamento da rede, coloque o tamanho do *batch* igual ao tamanho do *dataset*.