Pós-Graduação Lato Sensu
Curso de Especialização em Inteligência Artificial

Introdução a Inteligência Artificial

Prof. Dr. **Lucas Dias Hiera Sampaio**

PARTE 04

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CAMPUS CORNÉLIO PROCÓPIO

Bibliotecas para Inteligência Artificial

1. Apresentação

Na última semana, estudamos exemplos de problemas e como algoritmos de busca podem ser utilizados para encontrar soluções. Nesta semana teremos dois principais pontos a serem estudados: de forma teórica, vamos abordar quais são as principais bibliotecas utilizadas na linguagem de programação Python que são utilizadas em aplicações de inteligência artificial; já na prática desta semana vamos aprender a configurar um ambiente de desenvolvimento e realizar alguns experimentos simples utilizando a linguagem Python.

2. Bibliotecas

O conceito de biblioteca na ciência da computação segue a mesma lógica de uma biblioteca de universidade ou escola - todavia, no lugar de livros, artigos e outros trabalhos, tem-se uma coleção de subprogramas que são utilizados no desenvolvimento de *software*. As bibliotecas permitem o reaproveitamento de código e a modularização do sistema: logo, um programa que você irá desenvolver pode utilizar uma ou mais bibliotecas e o código das próprias bibliotecas pode também utilizar outras bibliotecas.

A história das bibliotecas de código começa com Charles Babbage e um dos primeiros computadores mecânicos. Em um artigo publicado em 1888, Charles sugeriu que diferentes operações computacionais poderiam ser gravadas em diferentes cartões perfurados. Os cartões seriam posteriormente reutilizados de tal forma a compor uma biblioteca.

No ramo da inteligência artificial o uso de bibliotecas é muito comum: nas aulas de programação em Python, por exemplo, vocês estudarão bibliotecas utilizadas para manipulação numérica, para realizar o pré-processamento e visualização de dados. Na próxima seção serão descritas as principais bibliotecas utilizadas na atualidade em aplicações de IA.

3. Bibliotecas para IA

A **SciPy** (Scientific Python) é uma biblioteca livre e de código aberto para ciência de dados que é utilizada para computação de alto-nível. Atualmente mais de 600 programadores contribuem para o desenvolvimento da biblioteca que é utilizada de

forma extensiva atualmente visto que ela amplia as funcionalidades já existentes na biblioteca **NumPy**. Entre as diferentes características da biblioteca e aplicações pode-se destacar:

- Comandos de alto nível para manipulação e visualização de dados;
- Processamento de imagens multidimensional;
- Funções para resolução de equações diferenciais;
- Utilizada em algoritmos de otimização;
- Utilizada em álgebra linear;
- Utilizada em aplicações com imagens;

A biblioteca **Matplotlib** é livre e de código aberto e utilizada para a visualização dos dados e resultados. Hoje é mantida por aproximadamente 700 programadores. Ela também possui uma API (*Application Programming Interface*) capaz de tornar os gráficos gerados por meio dela utilizáveis em outras aplicações. Entre as principais características e aplicações desta biblioteca estão:

- Pode ser utilizada como substituta para o *MatLab* - um *software* proprietário;
- Pode ser utilizada em qualquer sistema operacional e gerar diferentes formatos de saída;
- Consome pouca memória;
- Utilizada para avaliar a correlação entre variáveis;
- Utilizada para visualizar o intervalo de confiança dos modelos;
- Utilizada detectar *outliers*;
- Utilizada para observar dados a fim de obter *insights* sobre os mesmos;

A biblioteca **Scikit-learn** contém quase todas as ferramentas e algoritmos de aprendizado de máquina comumente utilizados. De forma geral, funciona de forma contígua a NumPy e a SciPy. Entre as diferentes aplicações (problemas) pode-se citar:

- Clusterização;
- Classificação;
- Regressão;
- Seleção de modelo;
- Redução de dimensionalidade;

Quando se possui uma placa de vídeo dedicada é possível executar boa parte dos algoritmos de IA no *hardware* a fim de poupar tempo uma vez que estas placas possuem unidades de processamento gráfico (GPU) que são compostas por milhares de núcleos e são capazes de executar computação em paralelo. A biblioteca **PyTorch** oferece sub-rotinas capazes de explorar as GPUs provendo simultaneamente flexibilidade e velocidade de processamento. Entre as diferentes aplicações da PyTorch estão:

- Utilização em alto nível de abstração dos *Tensor cores* das GPUs mais modernas;
- Construção de redes neurais para aplicações de **deep learning**;

A biblioteca **Ramp** é livre e de código aberto. Ela é utilizada para construir e avaliar modelos preditivos por meio de aprendizado de máquina. É uma biblioteca modular que permite a utilização de diferentes formatos de dados como CSV (*comma separated values*), planilhas, bancos de dados SQL, etc. Em sua base a Ramp utiliza outras bibliotecas como Pandas e Scikit-learn. Entre as diferentes aplicações, Ramp pode ser utilizada para:

- Modelos de predição do comportamento de ativos em bolsas de valores;
- Modelos de predição de microclima;
- Modelos de predição de comportamento de usuário/cliente;

A biblioteca **Bob** é um conjunto de algoritmos e ferramentas voltados para aprendizado de máquina, visão computacional e processamento de sinais. Sendo assim, a biblioteca é capaz de fazer a leitura e escrita de diferentes tipos de arquivos e formatos como áudio, imagem e vídeo. Adicionalmente, ela inclui algoritmos e modelos capazes de realizar reconhecimento facial, de locutor e de emoções, bem como a autenticação biométrica.

A biblioteca **PySwarms** traz uma implementação de uma heurística do contexto de inteligência de enxames (*swarm intelligence*) - o PSO ou *particle swarm optimization* - um algoritmo capaz de utilizar a inteligência de um grupo de animais para realizar a busca da solução para diferentes problemas de otimização. Entre as diferentes aplicações pode-se citar:

- Solucionar problemas de otimização de domínio contínuo;
- Realizar o treinamento de redes neurais e definição dos pesos sinápticos;

A biblioteca **PyGAD** é livre e de código aberto que implementa diferentes ferramentas para o uso de algoritmos genéticos na solução de problemas e para o aprendizado de máquina. Os algoritmos genéticos são algoritmos bioinspirados e uma das ferramentas da computação evolutiva. A biblioteca PyGAD traz diferentes tipos de operadores *crossover*, mutação e de seleção. São aplicações - mas não se restringem a - da biblioteca:

- Solucionar problemas de otimização (de domínio contínuo ou discreto);
- Treinar modelos de aprendizado de máquina;

Ao longo do curso de especialização os diferentes professores e as diferentes disciplinas irão abordar em detalhes algumas dessas bibliotecas. É importante entender que o uso de bibliotecas tem como objetivo facilitar o uso destas

ferramentas computacionais de tal forma a tornar a programação um aspecto secundário e não o principal objetivo das aulas.

4. A Biblioteca matplotlib

Nesta seção discutiremos os aspectos básicos de uso da matplotlib, exemplos de uso e documentação da biblioteca para que vocês tenham acesso mais rapidamente do que nas videoaulas para consulta no caso de dúvidas.

A biblioteca é utilizada para gerar figuras que são muitas vezes utilizadas para análise e apresentação de dados bem como para análise e apresentação dos resultados da aplicação de inteligência artificial na solução de problemas.

De forma geral, o processo de uso do matplotlib inicia-se pela seguinte linha:

```
import matplotlib.pyplot as plt
```

O módulo **pyplot** é responsável por oferecer uma interface para criação de figuras que segue a mesma lógica do software **MatLab**. Para criar uma figura utilizando o **pyplot** após a linha acima basta utilizar o comando:

```
fig, axs = plt.subplots()
```

O objeto **fig** será responsável gerenciar a figura enquanto o vetor **axs** representa os eixos de cada uma das figuras ou subfiguras. Quando há apenas 1 gráfico na figura, podemos acessar suas propriedades utilizando o **axs** diretamente. Já quando há múltiplas figuras faz-se necessário utilizar um indexador, veja:

```
axs[0].plot(x1, y1) #Primeira Figura
axs[1].plot(x2, y2) #Segunda Figura
```

é importante lembrar que o indexador da variável **axs** sempre iniciará em zero (0). Com estes comandos é possível criar os gráficos com os dados gravados nas variáveis **x1, y1, x2, y2**. Sem essas informações não é possível criar as figuras.

A figura criada na videoaula desta semana utiliza o código abaixo:

```
import matplotlib.pyplot as plt
import numpy as np

# O comando np.arange(início, fim, passo) cria um vetor cujo primeiro elemento é o número
```

```
# início e o último elemento é fim, os elementos entre estes dois são espaçados de passo em
# passo

t = np.arange(0.01, 5.0, 0.01)

# O comando np.exp() calcula a exponencial do número entre parênteses. Se o parâmetro for
# um vetor, então calcula o valor para cada elemento do vetor.

s = np.exp(-t)
# Instanciando a figura e os eixos
fig, ax = plt.subplots()

# Desenha a figura com base nos pontos t (para o eixo x) e s (para o eixo y)
ax.plot(t, s)

# Altera o label (descrição) do eixo x
ax.set_xlabel('Tempo')

# Altera o label (descrição) do eixo y
ax.set_ylabel('Quantidade')

# Altera o título da figura
ax.set_title('Should be growing...')

# Exibe as linhas auxiliares no gráfico
ax.grid(True)

# Apresenta a imagem na tela (ATENÇÃO: se ao executar o script nada estiver aparecendo
# verifique se não esqueceu este comando)
plt.show()
```

A documentação da biblioteca está disponível em: Documentação Matplotlib.