# camila-roteiro-aula-15-2

August 31, 2023

---

---

**Tópico I:**

Para que seja possivel avançar ainda mais no campo de IA, a tecnologia atual deverá ultrapassar limites e paradigmas ainda vigentes. Na divulgação presente no link 1 (abaixo), do dia 10 de Agosto de 2023, a IBM apresentou uma proposta de nova geração para chips voltados ao mercado de IA. Note que estamos presenciando tal revolução.

- Desenvolva um parágrafo, com aproximadamente 5 linhas, discutindo o que você entendeu sobre esta reportagem e também considere, neste parágrafo, a sua visão dos possíveis efeitos danosos à sociedade devido ao avanço rápido e não planejado das tecnologias de IA.

Link 1: https://research.ibm.com/blog/analog-ai-chip-inference

O conteúdo do link acima aborda sobre a pesquisa da IBM que explora a computação analógica em memória, baseando-se no funcionamento neural biológico. Com o intuito de superar os gargalos de transferência de dados e alcançar eficiência e velocidade, o desenvolvimento de um chip analógico da IBM mostra promissora eficiência energética e precisão em tarefas de reconhecimento de imagem, mas desafios de precisão e integração persistem. Contudo, é importante avaliar as implicações sociais do rápido avanço da IA, como a consequência de diversas pessoas desempregadas e despreparadas para essa revolução.

---

---

**Tópico II:**

Acesse o link 2 e o link 3, disponibilizados abaixo, e escreva um parágrafo (com aproximadamente 4 linhas) sobre a tecnologia desenvolvida e como ela pode impactar positivamente na sociedade:

Link 2: https://research.ibm.com/blog/chi-linechaser-app

Link 3: https://youtu.be/HJTam8_Yisw?t=4

O LineChaser é um aplicativo para smartphones que utiliza a IA com o intuito de auxiliar pessoas com deficiência visual a navegar em filas públicas. A aplicação faz uso da câmera RGB do telefone e um sensor de profundidade infravermelho embutido, no qual, detecta pedestres próximos e estima suas posições. Essa aplicação evidencia como a tecnologia pode ter impactos positivos na sociedade, visto que essa inovação tem o potencial de melhorar a vida de pessoas com deficiência visual, permitindo-lhes uma participação com mais autonomia na sociedade.

---

---

**Tópico III:**

- Acesse o link 4 e responda: em qual base de dados os pesquisadores alcançaram 97% de acurácia?
- Ainda neste tópico, escreva um parágrafo (com aproximadamente 4 linhas), discutindo as vantagens que o tipo de tecnologia (presente na reportagem) pode oferecer:

Link 4: https://research.ibm.com/blog/why-we-need-analog-AI-hardware

**Os pesquisadores alcançaram 97% de acurácia usando uma base de dados MNIST.**

A tecnologia de chips analógicos em pesquisa pela IBM oferece vantagens notáveis no campo de IA, visto que ao realizar cálculos diretamente na memória, eles economizam energia, permitindo eficiência energética superior. Além disso, a capacidade de executar tarefas de inferência e treinamento em hardware analógico pode acelerar consideravelmente o processo de desenvolvimento e ajuste de modelos de IA.

---

---

**Tópico IV:**

A figura abaixo apresenta a estrutura padrão do neurônio artificial:

**Responda:** o bias pode afetar o funcionamento do neurônio artificial de que modo?

Nesse contexto, o bias afeta principalmente como o neurônio responde às entradas. Ele ajusta a sensibilidade do neurônio a diferentes padrões nos dados e pode influenciar se o neurônio se ativa com facilidade ou não. Além disso, o bias ajuda a mover a linha de separação entre diferentes classes em problemas de classificação. Ademais, é importante ter em mente que se o bias for configurado de forma inadequada, o neurônio pode ter dificuldade em aprender e não conseguirá representar bem os padrões nos dados. Portanto, o ajuste correto do bias é importante para que o neurônio funcione de forma efetiva.

---

---

# 1 Tópico V: DESAFIO (Não é obrigatório responder este tópico)

Vimos em aula que o aprendizado profundo advém do aprendizado de máquina. Neste contexto, existem redes neurais que apresentam uma certa simplicidade em suas características construtivas. A Máquina de Aprendizado Extremo, do inglês *Extreme Learning Machine* (ELM), é uma dessas arquiteturas. Na figura abaixo, nota-se que a ELM é composta por apenas três camadas: uma camada de entrada, uma camada oculta e uma camada de saída.

O código abaixo apresenta uma implementação da ELM em Python:

[60]:
```python
# Importação necessária para definir matematicamente a ELM
import numpy as np

# Definindo a classe ELM
class ELM:

    # Inicializa a classe ELM com o número de entradas e neurônios na camada oculta
    def __init__(self, n_inputs: int, n_hidden=100):

        # Inicializa os pesos aleatórios da camada oculta
        self.random_weights = np.random.uniform(low=-.1, high=.1, size=[n_inputs, n_hidden])

    # Método para treinar o ELM
    def learn(self, X: np.ndarray, Y: np.ndarray):

        # Calcula as ativações da camada oculta usando o método _hidden_layer
        H = self._hidden_layer(X)

        # Calcula os pesos da camada de saída usando a pseudo-inversa de H
        self.output_weights = np.linalg.pinv(H) @ Y


    def _f(self, x: np.ndarray):

        # Função de ativação sigmoidal
        return 1. / (1. + np.exp(-x))


    # Calcula as ativações da camada oculta
    def _hidden_layer(self, inputs: np.ndarray):

        # Multiplicação matricial das entradas pelos pesos da camada oculta,
        return self._f(inputs @ self.random_weights)


    # Calcula as saídas do modelo
    def _output_layer(self, hidden: np.ndarray):

        # Multiplicação matricial das ativações da camada oculta pelos pesos da camada de saída
        return hidden @ self.output_weights

    # Método que permite chamar a instância da classe como uma função
```

```
    def __call__(self, inputs: np.ndarray):

        # Calcula as saídas finais do modelo ELM passando as entradas pelas␣
↪camadas oculta e de saída
        return self._output_layer(self._hidden_layer(inputs))
```

Para mais informações sobre a formulação matemática da ELM acesse:

- Link 1: https://erdem.pl/2020/05/introduction-to-extreme-learning-machines
- Link 2: https://link.springer.com/article/10.1007/s11042-021-11007-7

Inicialmente, vamos aplicar a ELM em um problema de classificação empregando o dataset Iris:

Para mais informações sobre o dataset Iris acesse o link: https://scikit-learn.org/stable/auto_examples/datasets/plot_iris_dataset.html

[61]:
```
# Importações para a aplicação
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer

# Carregar a base de dados Iris
iris = load_iris()
X = iris.data
y = iris.target

# Dividir os dados em conjuntos de treinamento e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[62]:
```
# Observe uma amostra do label original
y_train[0]
```

[62]: 0

[63]:
```
# Pré-processamento dos rótulos para classificação multiclasse
label_binarizer = LabelBinarizer()
y_train_bin = label_binarizer.fit_transform(y_train)
```

[64]:
```
# Observe uma amostra do label binarizado
y_train_bin[0]
```

[64]: array([1, 0, 0])

[65]:
```
# Pré-processamento dos rótulos do conjunto de teste para classificação␣
↪multiclasse
y_test_bin = label_binarizer.transform(y_test)
```

[66]:
```
# Criar o modelo ELM
n_inputs = X.shape[1]
n_hidden = 10
elm_model = ELM(n_inputs, n_hidden)

# Treinar o modelo ELM
elm_model.learn(X_train, y_train_bin)

# Fazer predições nos dados de teste
y_pred_bin = elm_model(X_test)

# Recuperar os label originais
y_pred = label_binarizer.inverse_transform(y_pred_bin)

# Calcular a acurácia das predições
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_pred, y_test)
print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

### 1.1 Agora: adapte o código de forma que a ELM execute a predição das classes da base de dados Fashion MNIST.

- Dica: Lembre-se de converter as imagens para vetores.

[67]:
```
# Importações
import tensorflow as tf
from sklearn.metrics import accuracy_score

# Carregar a base de dados Fashion MNIST
(X_train, y_train), (X_test, y_test) = tf.keras.datasets.fashion_mnist.
 ↪load_data()
```

[68]:
```
# Normalizar os valores dos pixels para o intervalo [0, 1]
X_train = X_train.astype(float) / 255.0
X_test = X_test.astype(float) / 255.0
```

[69]:
```
# Redimensionar as imagens para vetores
X_train = X_train.reshape(X_train.shape[0], -1)
X_test = X_test.reshape(X_test.shape[0], -1)
```

[70]:
```
# Pré-processamento dos rótulos para classificação multiclasse
label_binarizer = LabelBinarizer()
y_train_bin = label_binarizer.fit_transform(y_train)
y_test_bin = label_binarizer.transform(y_test)
```

[71]:
```
# Definindo a classe ELM adaptada para imagens
class ELM:

    def __init__(self, n_inputs: int, n_hidden=100):
        self.random_weights = np.random.uniform(low=-.1, high=.1,␣
 ↪size=[n_inputs, n_hidden])

    def learn(self, X: np.ndarray, Y: np.ndarray):
        H = self._hidden_layer(X)
        self.output_weights = np.linalg.pinv(H) @ Y

    def _f(self, x: np.ndarray):
        return 1. / (1. + np.exp(-x))

    def _hidden_layer(self, inputs: np.ndarray):
        return self._f(inputs @ self.random_weights)

    def _output_layer(self, hidden: np.ndarray):
        return hidden @ self.output_weights

    def predict(self, inputs: np.ndarray):
        hidden_activations = self._hidden_layer(inputs)
        output_activations = self._output_layer(hidden_activations)
        return label_binarizer.inverse_transform(output_activations)
```

[72]:
```
# Criar o modelo ELM
n_inputs = X_train.shape[1]
n_hidden = 100  # Você pode ajustar o número de neurônios na camada oculta
elm_model = ELM(n_inputs, n_hidden)

# Treinar o modelo ELM
elm_model.learn(X_train, y_train_bin)

# Fazer predições nos dados de teste
y_pred = elm_model.predict(X_test)

# Calcular a acurácia das predições
accuracy = accuracy_score(y_pred, y_test)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7749
```