



# Tensorflow 2.0: Deep Learning and Artificial Intelligence

**Anotações do curso:** <https://www.udemy.com/course/deep-learning-tensorflow-2/>

**Códigos do curso:** <https://deeplearningcourses.com/notebooks/JhnnzH3atbHGlhWYYwfCog>

**Exercícios:** [https://github.com/lazyprogrammer/machine\\_learning\\_examples/blob/master/tf2.0/exercises.txt](https://github.com/lazyprogrammer/machine_learning_examples/blob/master/tf2.0/exercises.txt)

**Github:** [https://github.com/lazyprogrammer/machine\\_learning\\_examples/tree/master/tf2.0](https://github.com/lazyprogrammer/machine_learning_examples/tree/master/tf2.0)

Obs: Como o github não suporta o uso de expressões matemáticas utilizando LaTeX, algumas partes do README estarão legíveis apenas na pré visualização no VS Code, foi convertido o markdown para pdf para que se possa ler as fórmulas porem há erros de formatação.

## Index

1. [Datasets usados no curso.](#)
2. [O que é Aprendizado de Máquina?](#)
  1. [Aprendizado de Máquina não é nada mais do que um problema geométrico](#)
3. [Teoria de Classificação Linear](#)

## Datasets usados no curso

**Obs:** Retirados do link -> <https://docs.google.com/document/d/1S7fAvk-MTUymxVB-FpG-fwlx6qR0ziNmK2Wp1BQbpzE/edit>

- Colab Basics:
  - Arrhythmia: <https://archive.ics.uci.edu/ml/datasets/Arrhythmia>
  - Auto MPG: <https://archive.ics.uci.edu/ml/datasets/Auto+MPG>
  - Daily minimum temperatures: [https://raw.githubusercontent.com/lazyprogrammer/machine\\_learning\\_examples/master/tf2.0/daily-minimum-temperatures-in-me.csv](https://raw.githubusercontent.com/lazyprogrammer/machine_learning_examples/master/tf2.0/daily-minimum-temperatures-in-me.csv)
- Machine Learning Basics:
  - Linear Regression: <https://raw.githubusercontent.com/lazyprogrammer/>

[machine\\_learning\\_examples/master/tf2.0/moore.csv](https://raw.githubusercontent.com/lazyprogrammer/machine_learning_examples/master/tf2.0/moore.csv)

- RNN:
  - Stock Returns: [https://raw.githubusercontent.com/lazyprogrammer/machine\\_learning\\_examples/master/tf2.0/sbox.csv](https://raw.githubusercontent.com/lazyprogrammer/machine_learning_examples/master/tf2.0/sbox.csv)
- Natural Language Processing:
  - Spam Detection: [https://lazyprogrammer.me/course\\_files/spam.csv](https://lazyprogrammer.me/course_files/spam.csv)
- Recommender Systems:
  - \_: <http://files.grouplens.org/datasets/movielens/ml-20m.zip>
- Transfer Learning:
  - \_: [https://lazyprogrammer.me/course\\_files/Food-5K.zip](https://lazyprogrammer.me/course_files/Food-5K.zip)

## O que é Aprendizado de Máquina?

Aprendizagem de máquina é um subcampo da Engenharia e da ciência da computação que evoluiu do estudo de reconhecimento de padrões e da teoria do aprendizado computacional em inteligência artificial. Em 1959, Arthur Samuel definiu aprendizado de máquina como o "campo de estudo que dá aos computadores a habilidade de aprender sem serem explicitamente programados"

## Aprendizado de Máquina não é nada mais do que um problema geométrico

- Um estatístico diria: Aprendizado de Máquina é apenas um ajuste de curva glorificado.
  - Ajuste de curva: método que consiste em encontrar uma curva que se ajuste a uma série de pontos.
- Tanto regressão quanto classificação são exemplos de aprendizado supervisionado
  - Regressão: prever um número. Em regressão, nós tentamos achar uma curva que mais se aproxime aos pontos passados.
  - Classificação: prever uma categoria. Em classificação, nós procuramos uma curva que separe os pontos em grupos.

## Teoria de Classificação Linear

Obs: Para facilitar o exemplo lida só com classificação linear binária.

A que separa um conjunto de pontos em um plano cartesiano pode ser escrita através da equação: (1)  $y = m * x + b$

Também podendo ser escrita como sendo: (2)  $w_1 * x_1 + w_2 * x_2 + b = 0$ , com  $x_1$  sendo o eixo horizontal e  $x_2$  sendo o eixo vertical.

Como podemos usar essa linha para classificar pontos?

$$(3) a = w_1 * x_1 + w_2 * x_2 + b$$

Usando a equação (3), podemos tomar uma decisão:

Se  $a \geq 0 \implies$  Classe 1

Se  $a < 0 \implies$  Classe 0

Matematicamente podemos encapsular esse processo de decisão em uma função de Heaviside (função degrau: assume valor 0 ou 1):

$$(4) \hat{y} = u(a), a = w_1 * x_1 + w_2 * x_2 + b$$

Como em Deep Learn nós preferimos utilizar funções mais suaves e diferenciáveis, é utilizada uma função Sigmoid (função logística: assume valores entre 0 e 1):

$$(5) \hat{y} = \sigma(a), a = w_1 * x_1 + w_2 * x_2 + b$$

com  $\hat{y}$  da equação (5) podendo ser interpretado como a probabilidade de que:

$$p(y = 1|x) = \sigma(w_1 * x_1 + w_2 * x_2 + b), \text{ ou seja, } y = 1 \text{ dado } x$$

Com o resultado da probabilidade podemos:

- Se  $p(y = 1|x) \geq 0 \implies$  prever 1, se não 0

Quando aplicamos uma função Sigmoid em cima de uma função linear, nós chamamos o modelo de Regressão Logística, e o argumento de uma função Sigmoid é chamado de ativação.

Usando o que sabemos até agora, pode-se notar um problema de notação: e se tivermos mais de duas entradas? ( $w_1 * x_1, w_2 * x_2, \dots, w_n * x_n$ )

R: Sem problemas, considerando  $w$  como um vetor de pesos e  $x$  como um vetor de

características :

$$(6) p(y = 1|x) = \sigma(w^T * x + b) = \sigma\left(\sum_{d=1}^D w_d * x_d + b\right)$$

Mas como podemos implementar, o que vimos até agora no Tensorflow?

- A expressão  $w^T * x + b$  é implementada através da função:  
`tf.keras.layers.Dense(output_size)`
- Para podermos implementar esse modelo utilizaremos dois layers:

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Input(shape=(D,)), #basicamente falando para o keras o tamanho do vetor de entrada x
    tf.keras.layers.Dense(1, activation='sigmoid') #especificar o tamanho de saida e função de ativação
])

model.compile(
    optimizer='adam', #será visto depois
    loss='binary_crossentropy', #será visto depois
    metrics=['accuracy'] #lista de métricas usadas, accuracy = correct/total
)

r = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=100 #numero de iterações para calculo dos pesos (vetor w)
)

# Visualiar a função de perda e outras métricas ao passar das iterações
# Com isso podemos avaliar se precisamos de mais epochs ou alterar outros hiper parametros
plt.plot(r.history['loss'], label='loss')
plt.plot(r.history['val_loss'], label='val_loss')
```