Curso de Verão 2023:
“Aprendizado de máquina e Física”

Aula 2: Classificação binária
Portas lógicas AND e OR.

Aula: Classificação binária - Objetivos

Nesta aula, temos o objetivo de:

1) Ilustrar a utilização de redes neurais em problemas de *classificação binária* (0 ou 1).

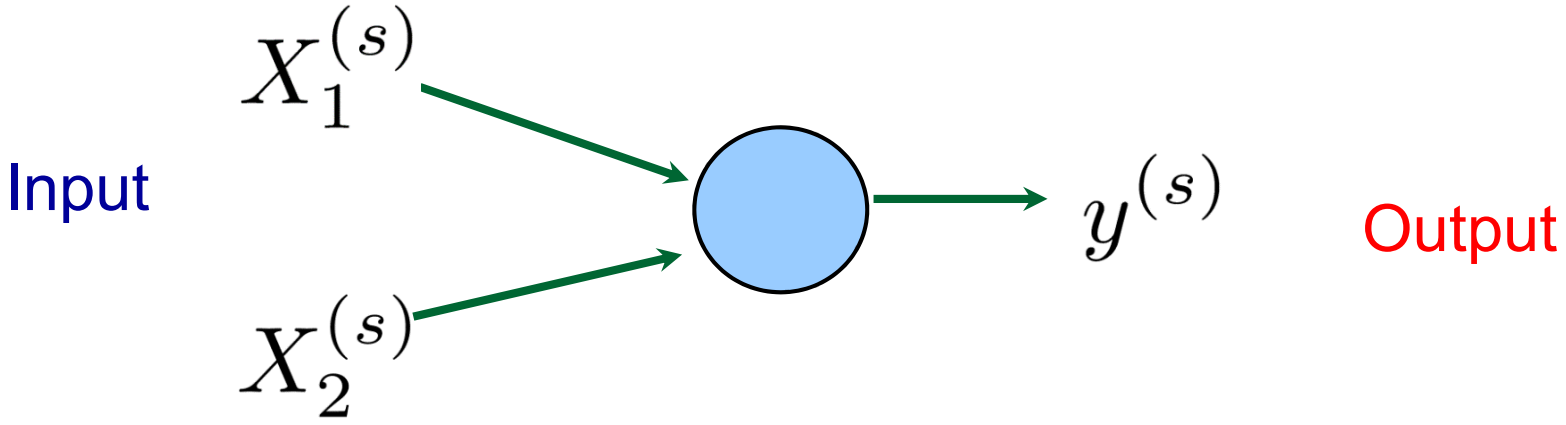
2) Usar uma RN simples (1 neurônio) para simular as portas lógicas AND e OR.

Tarefa: Treinar uma "rede neural de um neurônio" para aprender o comportamento de portas lógicas (AND e OR) e usar a rede convergida para visualizar as regiões de classificação.

Tempo aproximado: 15 a 25 min (lembrando que o *debug* é a maior parte disso!).

Portas lógicas

Uma porta lógica admite dois valores de entrada (vetor de inputs) binários (0 ou 1).
Retorna um valor de saída (output) também binário.



Dois exemplos importantes:

Porta **AND**

s	$X_1^{(s)}$	$X_2^{(s)}$	$y^{(s)}$
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

Porta **OR**

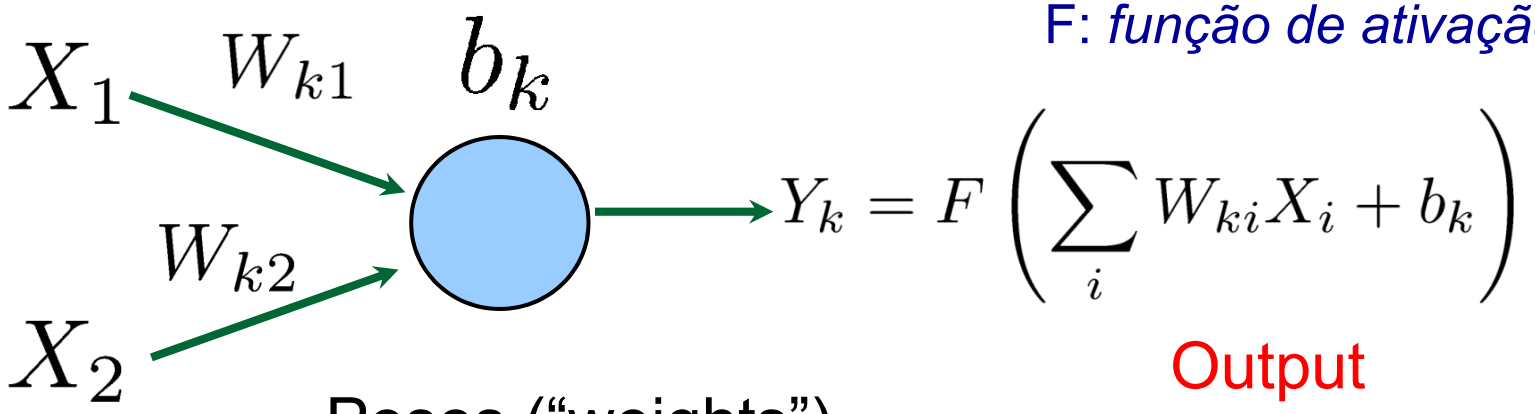
s	$X_1^{(s)}$	$X_2^{(s)}$	$y^{(s)}$
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

Portas lógicas = classificação

Podemos pensar em uma rede neural que *simule* uma dada porta lógica.

Para isso, precisamos determinar o pesos e bias através de “treino”.

F: *função de ativação*.



Input

Output

Pesos (“weights”)

“Treino” **AND**

s	$X_1^{(s)}$	$X_2^{(s)}$	$y^{(s)}$
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

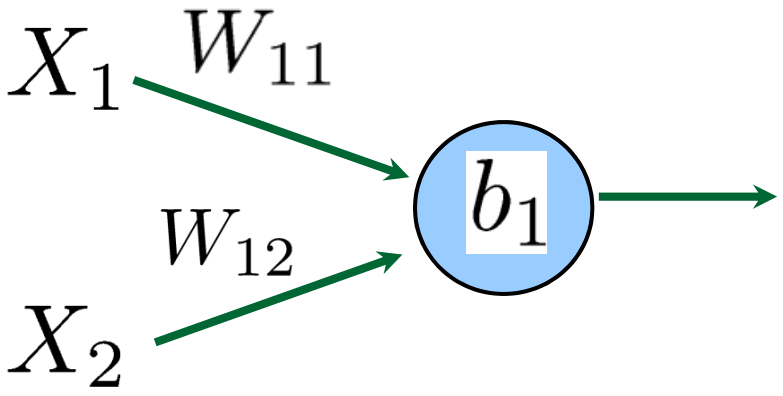
“Treino” **OR**

s	$X_1^{(s)}$	$X_2^{(s)}$	$y^{(s)}$
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

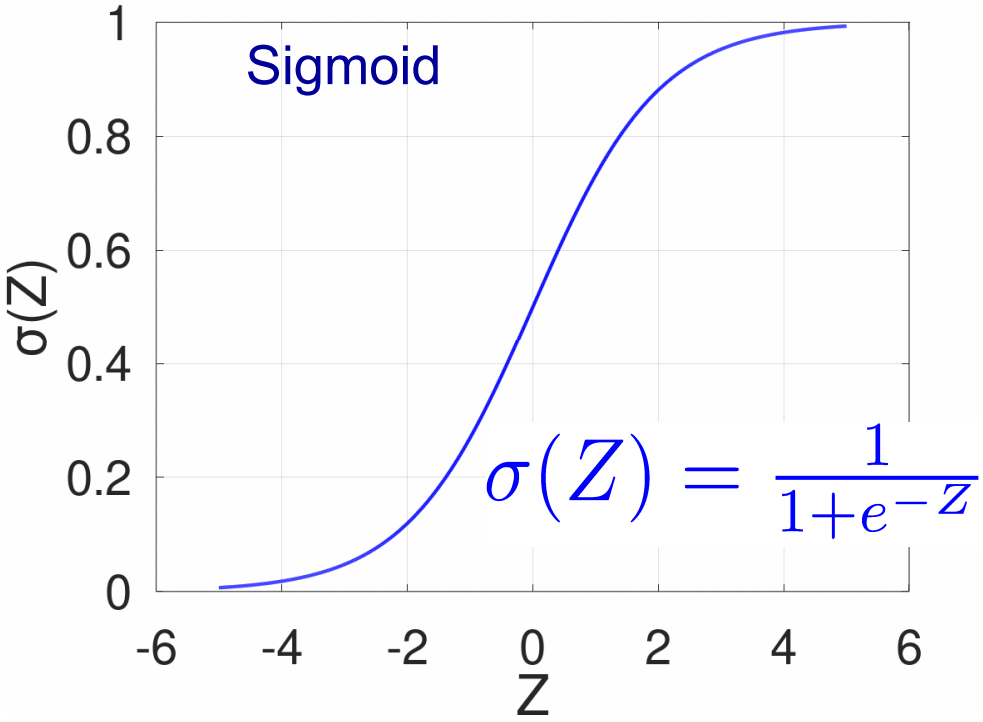
Classificação binária: quais funções usar?

Para manter o output entre 0 e 1, usaremos uma ***função sigmoid***

F(z)=1/(1+exp(-z)) como função de ativação:



$$Y_1 = \sigma(W_{11}X_1 + W_{12}X_2 + b_1)$$



Bônus: forma simples da derivada.

$$\frac{d\sigma(Z)}{dZ} = \sigma(Z)\,(1-\sigma(Z)) = Y_1\,(1-Y_1)$$

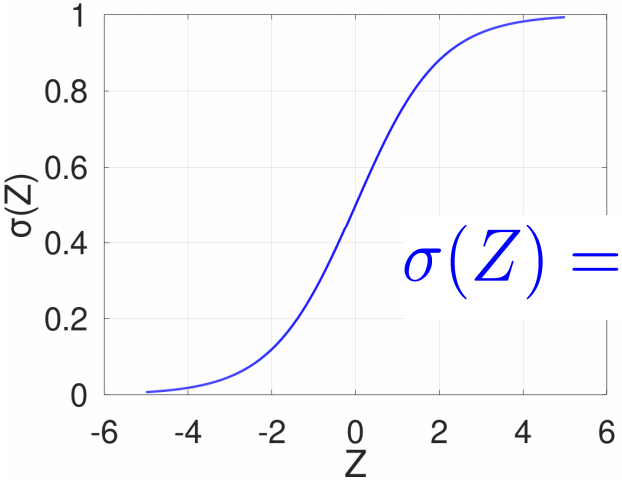
Classificação binária: quais funções usar?

Para o caso de portas lógicas AND e OR, é suficiente um único neurônio.

O problema de classificação binária é essencialmente uma *regressão logística.*

Nesses modelos, é adequado utilizar uma **função log-loss** como função-custo;

$$\mathcal{C}(W_{1i}, b_1) = \frac{1}{N_s} \sum_{s=1}^{N_s} \left[\underbrace{-y^{(s)} \ln\left(Y_1^{(s)}\right)}_{\text{Custo para } y^{(s)}=1} - \underbrace{\left(1 - y^{(s)}\right) \ln\left(1 - Y_1^{(s)}\right)}_{\text{Custo para } y^{(s)}=0} \right]$$



$$\sigma(Z) = \frac{1}{1+e^{-Z}}$$

$$\underbrace{Y_1^{(s)} = \sigma\left(\sum_{i=1}^{2} W_{1i} X_i^{(s)} + b_1\right)}_{0<Y_1^{(s)}<1}$$

O problema: encontrar W_{11} , W_{12} e b_1 que minimizem $\mathcal{C}(W_{1i}, b_1)$!

Minimizando a função-custo.

$$\mathcal{C}(W_{1i}, b_1) = \frac{1}{N_s} \sum_{s=1}^{N_s} \left[-y^{(s)} \ln\left(Y_1^{(s)}\right) - \left(1 - y^{(s)}\right) \ln\left(1 - Y_1^{(s)}\right) \right]$$

Tarefa: derive as relações abaixo,

$$\begin{cases} \dfrac{\partial \mathcal{C}}{\partial W_{1i}} = -\dfrac{1}{N_s} \displaystyle\sum_{s=1}^{N_s} \left[y^{(s)} - Y_1^{(s)} \right] X_i^{(s)} \qquad Y_1^{(s)} = \sigma\left(\displaystyle\sum_{i=1}^{2} W_{1i} X_i^{(s)} + b_1 \right) \\ \dfrac{\partial \mathcal{C}}{\partial b_1} = -\dfrac{1}{N_s} \displaystyle\sum_{s=1}^{N_s} \left[y^{(s)} - Y_1^{(s)} \right] \qquad \sigma(Z) = \dfrac{1}{1 + e^{-Z}} \end{cases}$$

Método do gradiente descendente (gradient descent) para encontrar o mínimo.

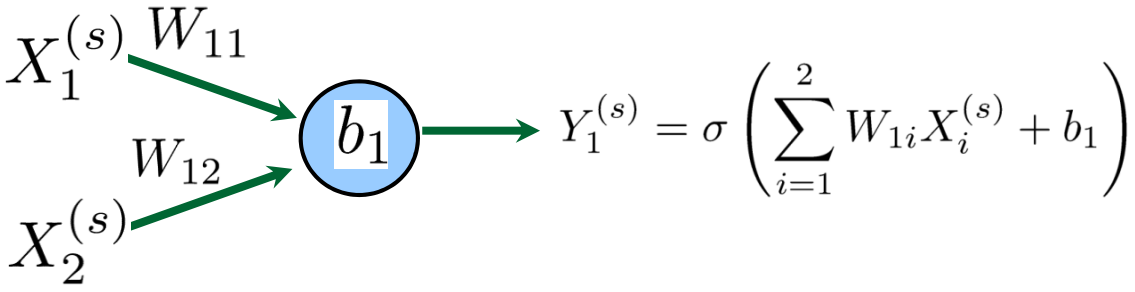
Escolhemos $W^{(0)}_{1i}$ e $b^{(0)}_1$ iniciais e atualizamos na forma:

$$\begin{cases} W_{1i}^{(1)} = W_{1i}^{(0)} + \Delta W_{1i} \\ b_1^{(1)} = b_1^{(0)} + \Delta b_1 \end{cases} \Rightarrow \text{onde} \quad \begin{cases} \Delta W_{11} = -\eta \dfrac{\partial \mathcal{C}}{\partial W_{11}} \\ \Delta b_1 = -\eta \dfrac{\partial \mathcal{C}}{\partial b_1} \end{cases}$$

$\eta \rightarrow$ “hiperparâmetro”

Algoritmo para minimizar a função-custo.

1) Escolhemos $W^{(0)}_{1i}$ e $b^{(0)}_1$ iniciais e, para cada par $X^{(s)}_i$, calculamos $Y^{(s)}_1$



2) Calculamos ΔW_{1i} e Δb_1 usando os N outputs $Y^{(s)}_1$ e os N dados $(X^{(s)}, y^{(s)})$:

$$\begin{cases} \Delta W_{1i} &= +\frac{\eta}{N_s} \sum_{s=1}^{N_s} \left[y^{(s)} - Y_1^{(s)} \right] X_i^{(s)} \\ \Delta b_1 &= +\frac{\eta}{N} \sum_{s=1}^{N_s} \left[y^{(s)} - Y_1^{(s)} \right] \end{cases}$$

3) Fazermos o update em W_{11} e b_1 e re-calculamos $\mathcal{C}(W_{1i}, b_1)$

$$\begin{cases} W_{1i}^{(1)} &= W_{1i}^{(0)} + \Delta W_{1i} \\ b_1^{(1)} &= b_1^{(0)} + \Delta b_1 \end{cases} \Rightarrow \mathcal{C}(W_{1i}, b_1) = \frac{1}{N_s} \sum_{s=1}^{N_s} \left[-y^{(s)} \ln\left(Y_1^{(s)}\right) - \left(1 - y^{(s)}\right) \ln\left(1 - Y_1^{(s)}\right) \right]$$

4) Repete-se o processo até convergir.

Classificação binária – Tarefa 2

"Treine" uma rede de um neurônio para simular portas lógicas usando o método do gradiente descendente.

"Treino" **AND**

s	$X_1^{(s)}$	$X_2^{(s)}$	$y^{(s)}$
1	0	0	0
2	0	1	0
3	1	0	0
4	1	1	1

"Treino" **OR**

s	$X_1^{(s)}$	$X_2^{(s)}$	$y^{(s)}$
1	0	0	0
2	0	1	1
3	1	0	1
4	1	1	1

- Use o algoritmo anterior de minimização, escolhendo $W^{(0)}_{1i}$ e $b^{(0)}_1$ iniciais (use valores entre zero e um).
- Use valores de η "grandes" (por exemplo, η=1).
- Rode o algoritmo por 2000 iterações OU até que

$$\left| \mathcal{C}^{(n+1)} - \mathcal{C}^{(n)} \right| < 10^{-5}$$

- Treine primeiramente para a porta AND.
- Após a convergência, faça um contour plot de pontos $(X_1, X_2) \rightarrow Y$ com X_i variando entre 0 e 1.
- Note que o plot é claramente dividido em duas regiões: uma com $Y \approx 1$ e outra com $Y \approx 0$.
- Repita para a porta OR.

Classificação binária – Debug/Dicas

- Exemplo: porta AND com "chutes iniciais" $W^{(0)}_{11}$ =0.4, $W^{(0)}_{12}$ =0.6 e $b^{(0)}_{1}$ =0.7, η=1

```
n+1 = 1324, W1= (7.98328 7.98328) , b1= -12.14642,
 Cost(n+1) = 0.01314270, Cost(n) = 0.01315270
is = 1, X1=0.00 X2=0.00 Ytrue = 0.00000 Ypredict = 0.00001
is = 2, X1=0.00 X2=1.00 Ytrue = 0.00000 Ypredict = 0.01532
is = 3, X1=1.00 X2=0.00 Ytrue = 0.00000 Ypredict = 0.01532
is = 4, X1=1.00 X2=1.00 Ytrue = 1.00000 Ypredict = 0.97855
```

- Use funções externas para a sigmoid, função custo, e para o output:

$$Y_1 = \sigma\left(\sum_{i=1}^{2} W_{1i}X_i + b_1\right)$$

- Use np.log(Y+np.finfo(float).eps) para evitar erros se Y=0.
- Notação padrão: W →"matriz"; X → "vetor"
- O contour plot mostra as "regiões" de classificação (Y≈1 e Y≈0).