UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Aula 10: Redes neurais para textos [1]

Prof. Dennis Giovani Balreira

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

INF01221 - Tópicos Especiais em Computação XXXVI:
Processamento de Linguagem Natural

.inf
INSTITUTO DE INFORMÁTICA
UFRGS

Conteúdo

- Redes neurais para textos [1]
  - Redes neurais
  - Perceptron
  - Função de ativação
  - Loss function
  - Backpropagation

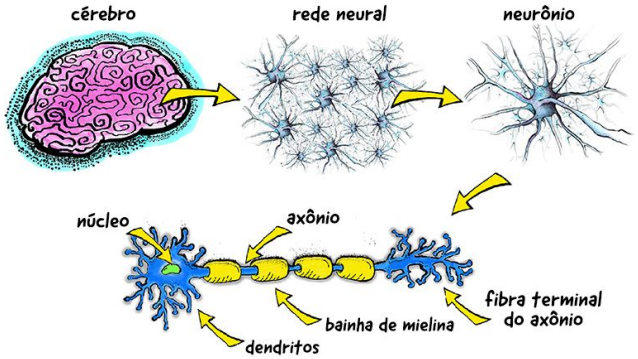
Onde estamos em PLN?

- Algoritmos tradicionais
  - Predominantes entre o final dos anos 1990 até ~2016
  - BoW features + Aprendizado de Máquina

- Embeddings fixas + Deep Learning
  - Predominates de ~2014 até ~2019
  - Word2vec, Glove, FastText + LSTM

- Embeddings contextuais + Large Language Models
  - Estado da arte em diversas tarefas
  - BERT, GPT, etc.

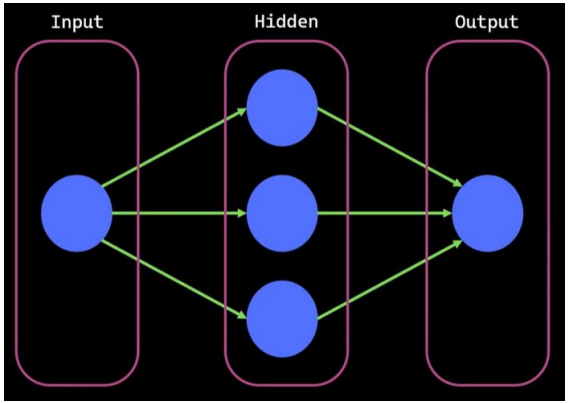
Redes Neurais



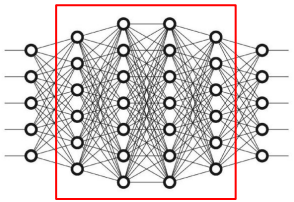
- Modelos computacionais inspirados na estrutura e funcionamento do cérebro humano, compostos por unidades chamadas neurônios artificiais, organizados em camadas

- Neurônio: unidades básicas que realizam operações matemáticas

- Camadas:
  - Input: recebe os dados
  - Hidden: realizam transformações nos dados
  - Output: gera o resultado final da rede



- "Deep learning":
  - Muitas camadas *hidden*!

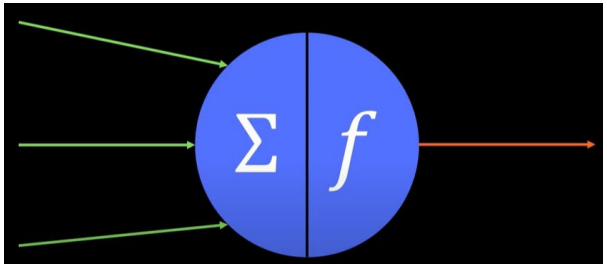


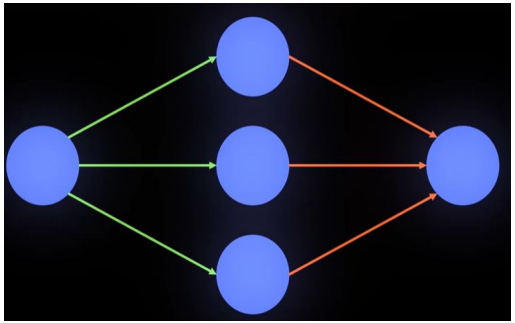
Redes Neurais

- As redes neurais funcionam através de uma sequência de operações em que os dados são processados nas camadas intermediárias (*hidden*) até chegar na saída (*output*)

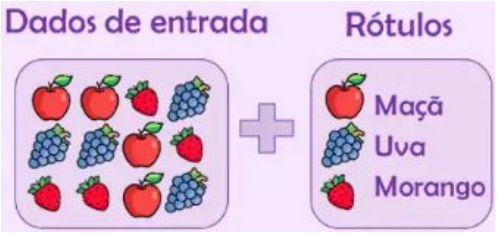
- Cada neurônio é chamado de perceptron
  - Primeira parte soma informações Σ
  - Segunda parte modela de acordo com função f

- Em seguida, a informação é passada para as próximas camadas adiante “para frente” (*feedforward*)



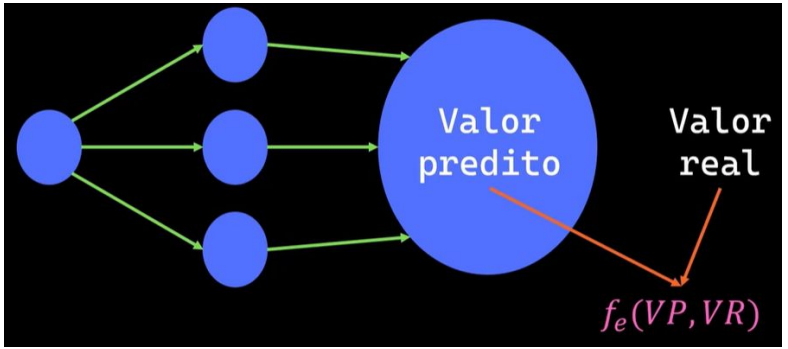


Redes Neurais



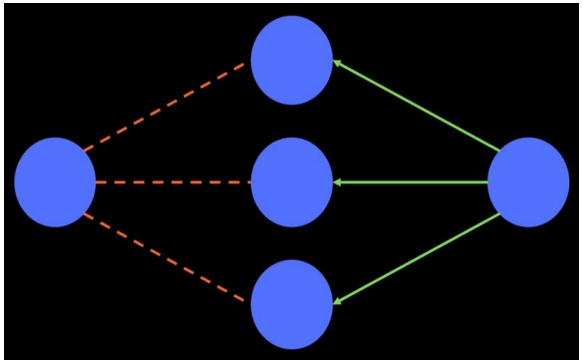
- Faz parte de técnicas de aprendizado supervisionado*
  - Pode ser feito para classificação ou regressão
  - Existem os dados e seus valores “corretos” (*ground truth*)

- Com isso, é possível calcular a *loss function*:
  - Compara a previsão feita pelo modelo com a resposta real (*ground truth*)
  - Gera um valor de “erro” indicando o quão próximo do real o valor previsto foi



*Algumas abordagens usam redes neurais com com aprendizado não supervisionado ou por reforço

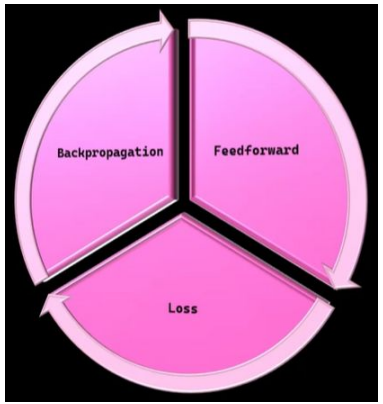
Redes Neurais



- Após saber o erro (via *loss function*), é importante "calibrar" a rede para que ela vá aprendendo
  - Isso é feito pelo algoritmo Backpropagation
  - Ideia é "voltar" na rede, ajustando os pesos com base no erro encontrado

- Ciclo de repetição:
  - Feedforward
  - Loss function
  - Backpropagation

- Ciclo para a partir de:
  - Épocas: número de vezes que o ciclo acontece
  - Tolerância: executa até que uma métrica atinja uma diferença determinada

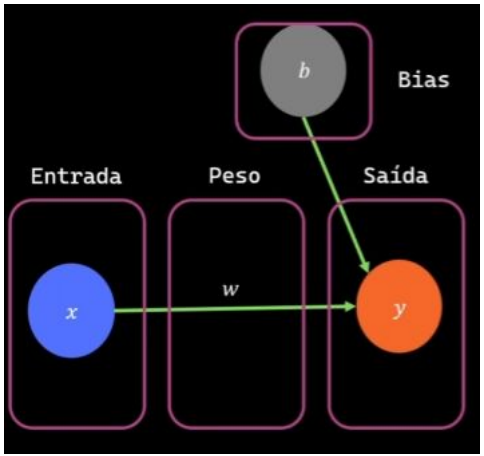


Perceptron

- Modelo de rede neural simples, criado em 1957
- É composto por uma única unidade (neurônio)
  - Cada unidade é multiplicada por pesos

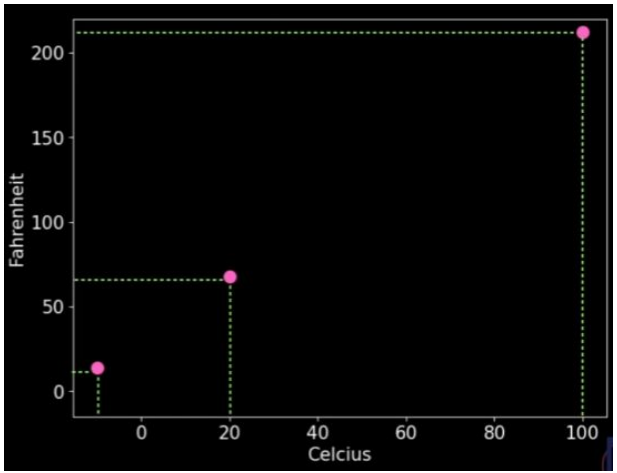
Problema de regressão!

Temperaturas	
°C	°F
-10	14
0	
20	68
30	
100	212



$$y = x \cdot w + b$$

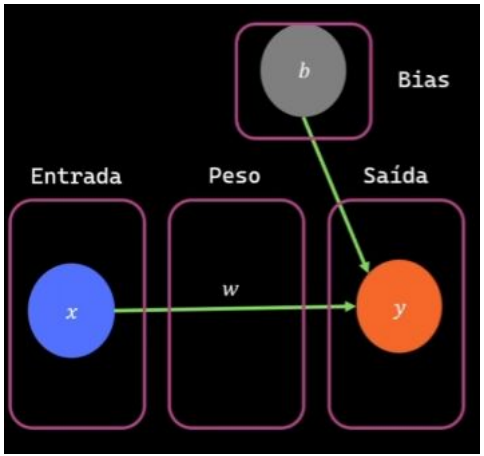
O objetivo é achar “valores” bons de w e b para conseguir “representar” bem quaisquer outros pontos que surjam posteriormente



Perceptron

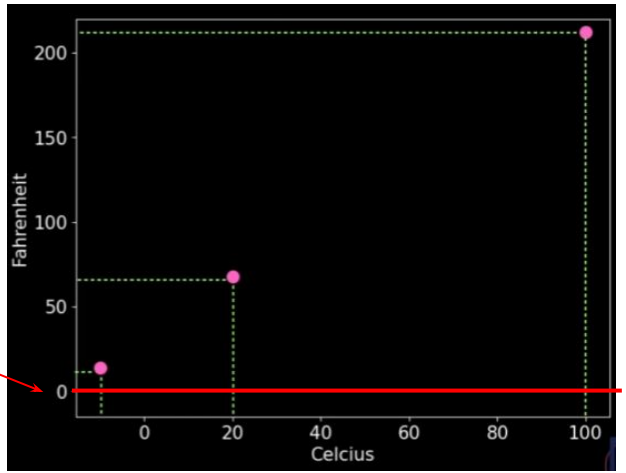
Temperaturas	
°C	°F
-10	14
0	
20	68
30	
100	212

- Modelo de rede neural simples, criado em 1957
- É composto por uma única unidade (neurônio)
  - Cada unidade é multiplicada por pesos



$$y = x \cdot w + b$$

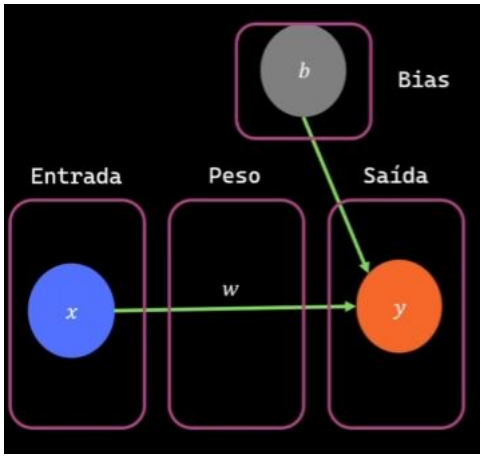
w=0.0
b=0.0



Perceptron

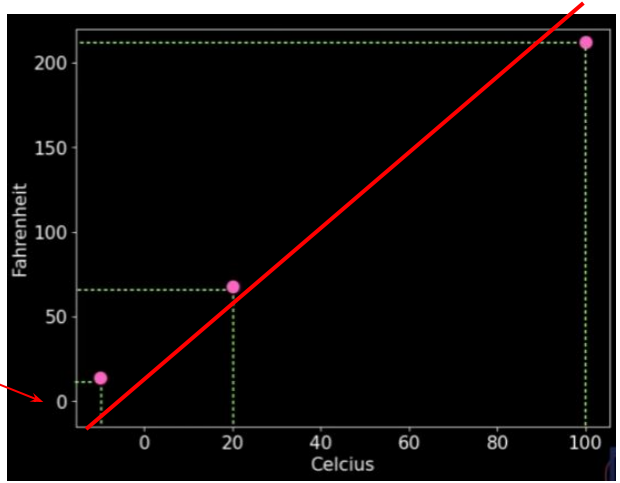
- Modelo de rede neural simples, criado em 1957
- É composto por uma única unidade (neurônio)
  - Cada unidade é multiplicada por pesos

Temperaturas	
°C	°F
-10	14
0	
20	68
30	
100	212



$$y = x \cdot w + b$$

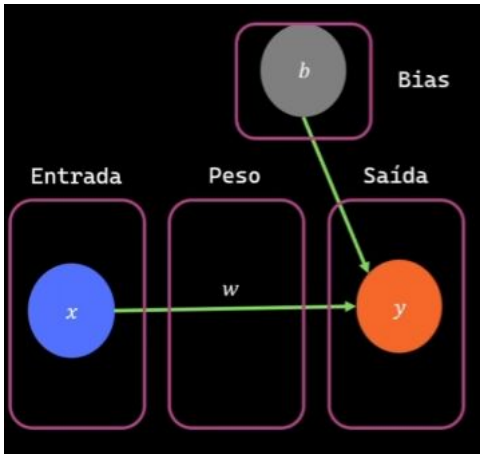
w=2.14
b=12.86



Perceptron

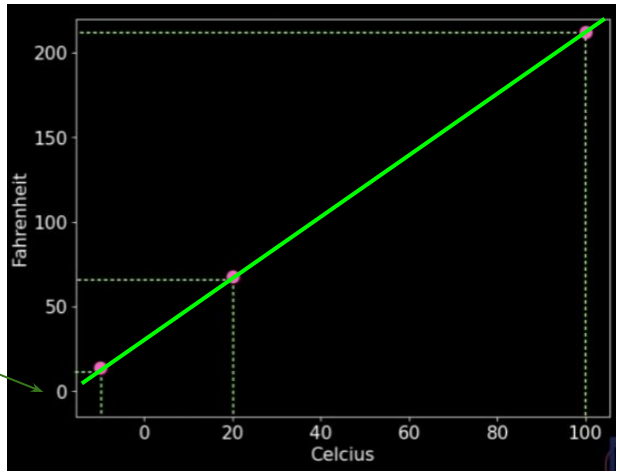
Temperaturas	
°C	°F
-10	14
0	
20	68
30	
100	212

- Modelo de rede neural simples, criado em 1957
- É composto por uma única unidade (neurônio)
  - Cada unidade é multiplicada por pesos



$$y = x \cdot w + b$$

w=1.80
b=32.00

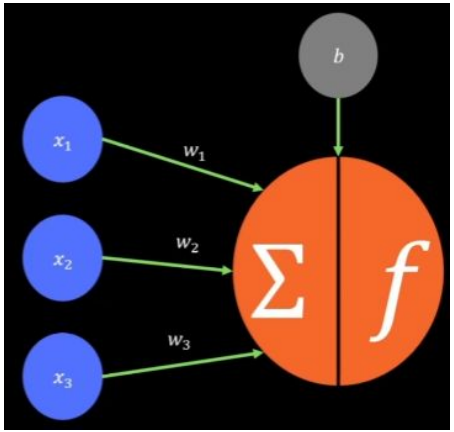


Perceptron: múltiplas entradas

- Pode ser estendido para suportar múltiplos valores de entrada
  - Cada valor $\mathbf{x}_i$ representa uma “feature”

Problema de classificação!

Estudou	Conhecimento prévio	Facilidade de aprendizado	Nome	Resultado
0.8	0.5	0.7	Miguel	1
0.3	0.2	0.8	Bruno	0



$$y = w_1x_1 + w_2x_2 + w_3x_3 + b$$

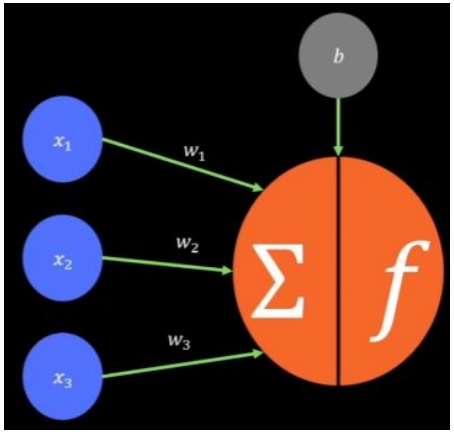
$$y = \sum_{i=1}^{n} w_i x_i + b$$

Perceptron: múltiplas entradas

- Pode ser estendido para suportar múltiplos valores de entrada
  - Cada valor $\mathbf{x}_i$ representa uma “feature”

Problema de classificação!

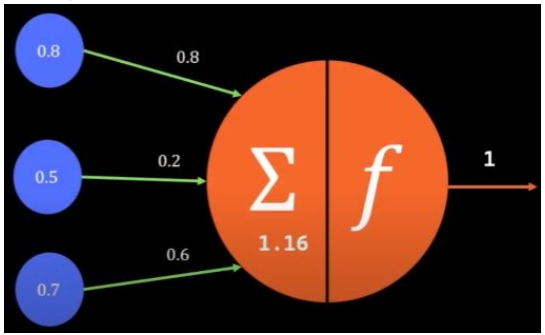
Estudou	Conhecimento prévio	Facilidade de aprendizado	Nome	Resultado
0.8	0.5	0.7	Miguel	1
0.3	0.2	0.8	Bruno	0



Para o Miguel:

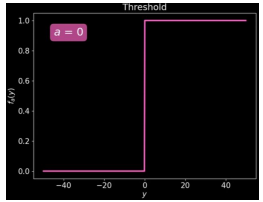
$$y = w_1x_1 + w_2x_2 + w_3x_3 + b$$

$$y = \sum_{i=1}^{n} w_i x_i + b$$



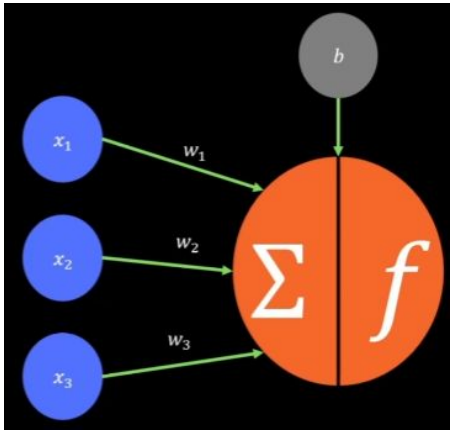
$$y_M = 0.8 * 0.8 + 0.2 * 0.5 + 0.6 * 0.7$$

$$f_a(y) \begin{cases} 1 \text{ se } y \geq 1 \\ 0 \text{ se } y < 1 \end{cases}$$



Perceptron: múltiplas entradas

- Pode ser estendido para suportar múltiplos valores de entrada
  - Cada valor $\mathbf{x}_i$ representa uma “feature”



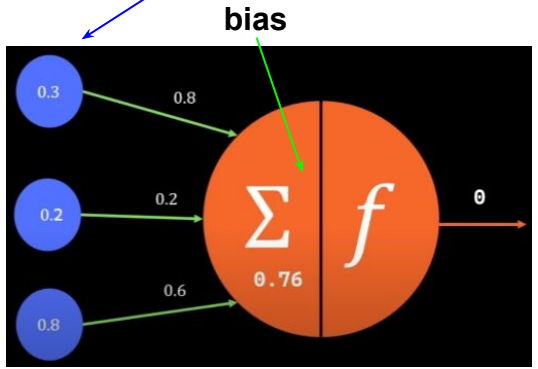
Para o Bruno:

$$y = w_1x_1 + w_2x_2 + w_3x_3 + b$$

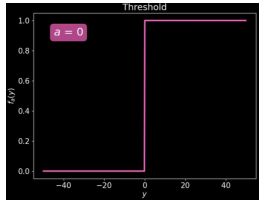
$$y = \sum_{i=1}^{n} w_i x_i + b$$

Problema de classificação!

Estudou	Conhecimento prévio	Facilidade de aprendizado	Nome	Resultado
0.8	0.5	0.7	Miguel	1
0.3	0.2	0.8	Bruno	0



$$f_a(y) \begin{cases} 1 \text{ se } y \geq 1 \\ 0 \text{ se } y < 1 \end{cases}$$



$$y_B = 0.8 * 0.3 + 0.2 * 0.2 + 0.6 * 0.8 + 0.4$$

Função de ativação

- São aplicadas às somas ponderadas das entradas em cada neurônio
  - Funções de ativação devem ser não-lineares (geram “curvas”)

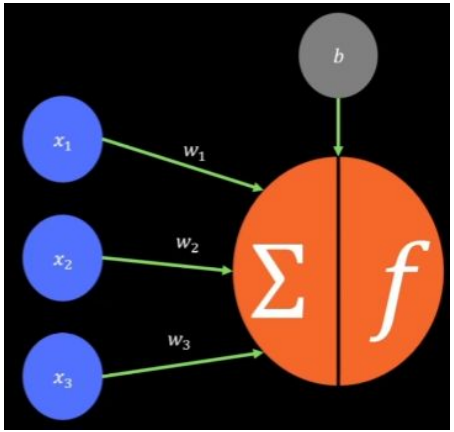
Funções de ativação:
- Step function
- Sigmoid
- ReLU
- Softmax

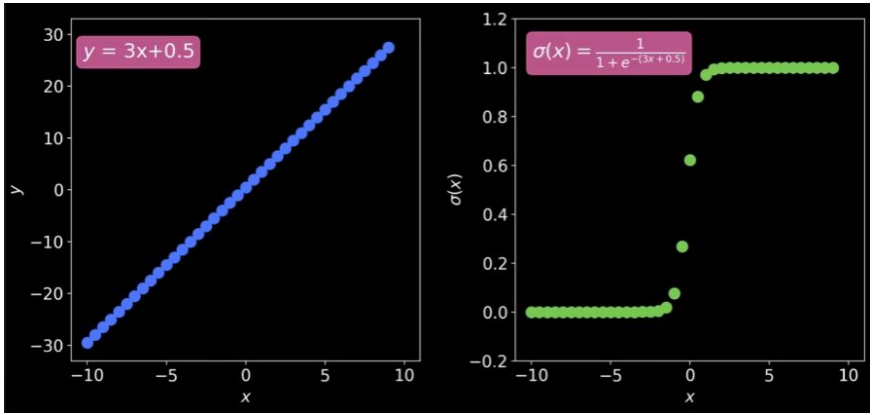
$$f(z) = \max(0, z)$$

$$f(z) = \begin{cases} 1 & \text{se } z \geq 0 \\ 0 & \text{se } z < 0 \end{cases}$$

$$f(z) = \frac{1}{1 + e^{-z}}$$

$$f(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}$$





Função de ativação

- São aplicadas às somas ponderadas das entradas em cada neurônio
  - Funções de ativação devem ser não-lineares (geram “curvas”)

Funções de ativação:
- Sigmoid

$$f(z) = \frac{1}{1 + e^{-z}}$$

Estudou	Conhecimento prévio	Facilidade de aprendizado	Nome	Resultado
0.8	0.5	0.7	Miguel	1
0.3	0.2	0.8	Bruno	0

Para o Bruno:

x = [0.3, 0.2, 0.8]

z = (0.6)(0.3) + (0.3)(0.2) + (0.5)(0.8) - 0.4

z = 0.18 + 0.06 + 0.4 - 0.4 = **0.24**

$$\hat{y}_{Bruno} = \frac{1}{1 + e^{-0.24}} \approx 0.560$$

Pesos:
w1=0.6
w2=0.3
w3=0.5

Bias:
b=0.4

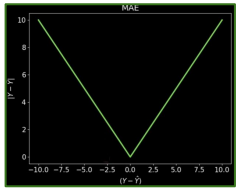
Loss function

Loss functions (regressão):
- MAE (L1) (Mean Absolute Error)
- MSE (Mean Squared Error)

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- Função que mede o erro entre a saída prevista por um modelo e o valor real esperado
- O objetivo do processo é minimizar a loss function
- Processo
  - Detectar erro
  - Modelar erro
  - Aprender

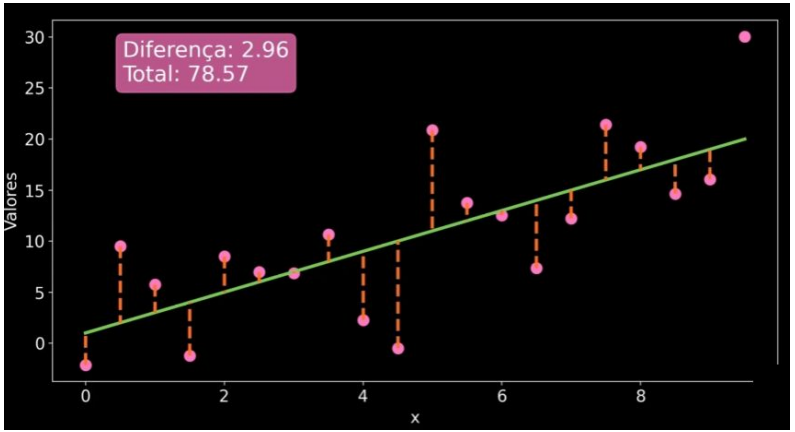
$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$



Y	$\hat{Y}$	$Y - \hat{Y}$	MAE
-0.3	0.2	-0.5	0.5
-0.2	-0.8	0.6	0.6
5.1	4.4	0.7	0.7
50	35	15	15

$$MAE = \frac{1}{4}(0.5 + 0.6 + 0.7 + 15)$$

$$MAE = \frac{1}{4}(16.8) = 4.2$$



Loss function

Loss functions (classificação):
- Binary Cross Entropy Loss
- Cross Entropy Loss

Na verdade é ln (natural)

- Função que mede o erro entre a saída prevista por um modelo e o valor real esperado
- O objetivo do processo é minimizar a loss function
- Processo
  - Detectar erro
  - Modelar erro
  - Aprender

$$\text{Loss} = -[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})]$$

y é o **rótulo real** (0 ou 1),

$\hat{y}$ é a **previsão do modelo** (probabilidade entre 0 e 1).

Estudou	Conhecimento prévio	Facilidade de aprendizado	Nome	Resultado
0.8	0.5	0.7	Miguel	1
0.3	0.2	0.8	Bruno	0

$$\hat{y}_{Bruno} = \frac{1}{1 + e^{-0.24}} \approx 0.560$$

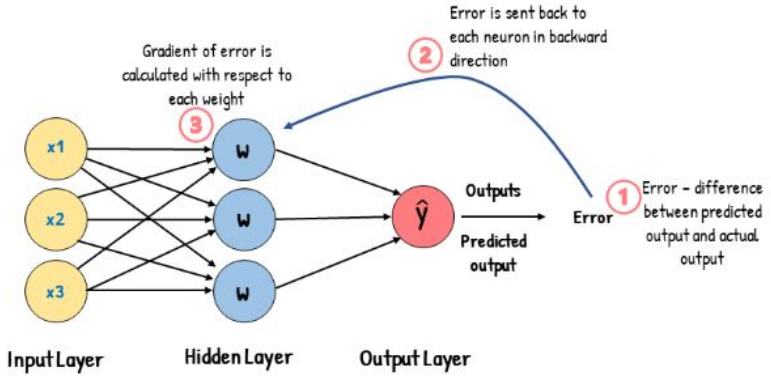
Função de ativação

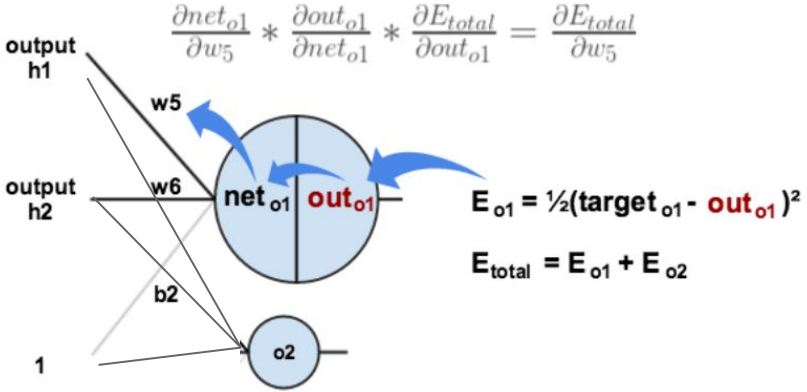
Quanto mais perto da classe errada, muito maior é a perda!

$$\text{Loss}_{Bruno} = -[0 \cdot \log(0.560) + 1 \cdot \log(1 - 0.560)] = -\log(0.440) \approx 0.820$$

Backpropagation

- Principal algoritmo para treinar redes neurais
  - Ajusta os pesos para minimizar o erro entre a previsão e o valor real
  - Usa gradiente descendente para calcular os gradientes dos pesos em relação à perda





Gradiente descendente

- Principal algoritmo de otimização para minimizar funções de perda ajustando os parâmetros do modelo (pesos e bias)

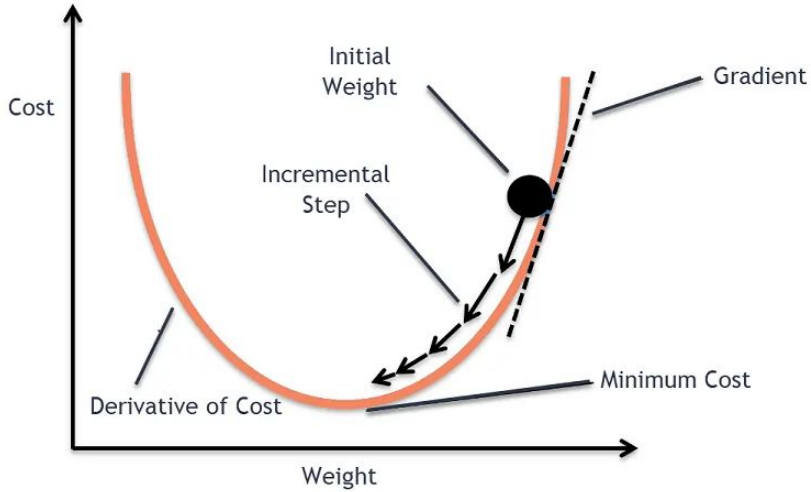
Peso anterior

$$w_{t+1} = w_t - \alpha \cdot \nabla L(w_t)$$

Peso novo

Gradiente da função de perda L em relação aos pesos w no passo t

Taxa de aprendizagem (*learning rate*)

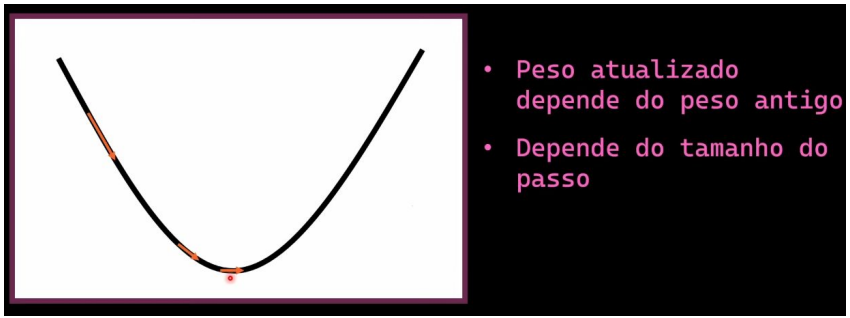


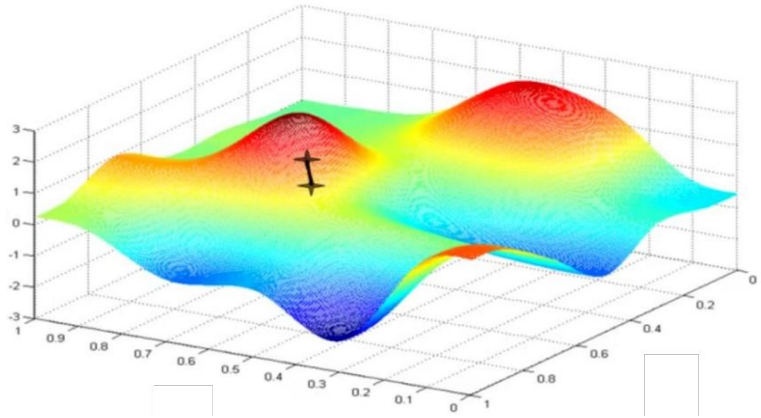
Gradiente descendente

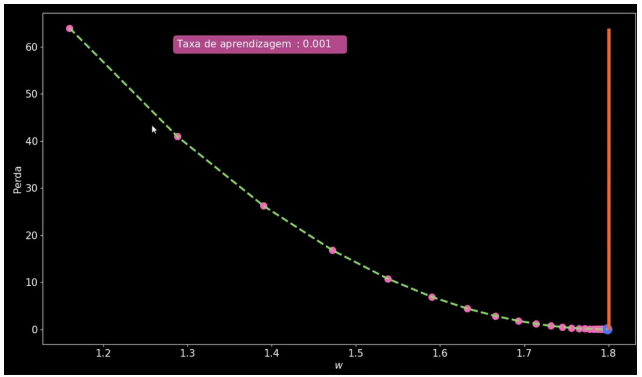
- Principal algoritmo de otimização para minimizar funções de perda ajustando os parâmetros do modelo (pesos e bias)

$$w_{t+1} = w_t - \alpha \cdot \nabla L(w_t)$$

Taxa de aprendizagem (*learning rate*)



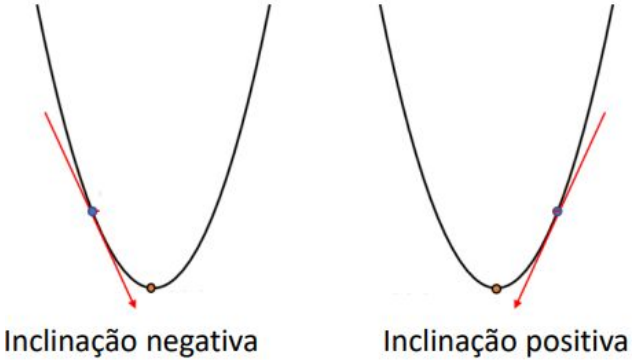




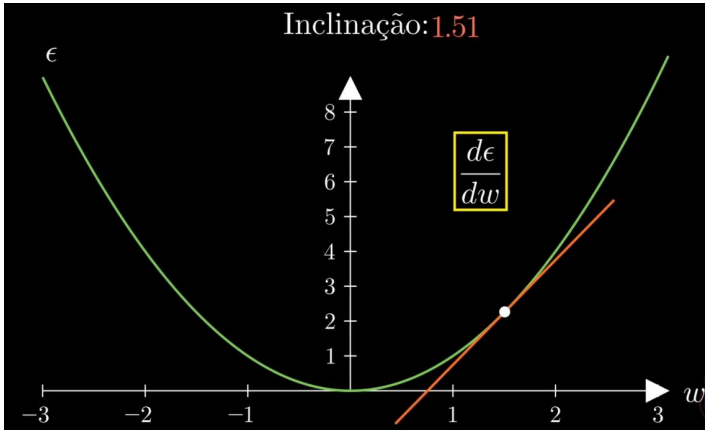
Gradiente descendente: derivada

- Principal algoritmo de otimização para minimizar funções de perda ajustando os parâmetros do modelo (pesos e bias)

$$w_{t+1} = w_t - \alpha \cdot \nabla L(w_t)$$



Derivada do erro com relação ao peso mede quanto o erro varia conforme o peso muda (inclinação em determinado ponto)



Gradiente descendente: regra da cadeia

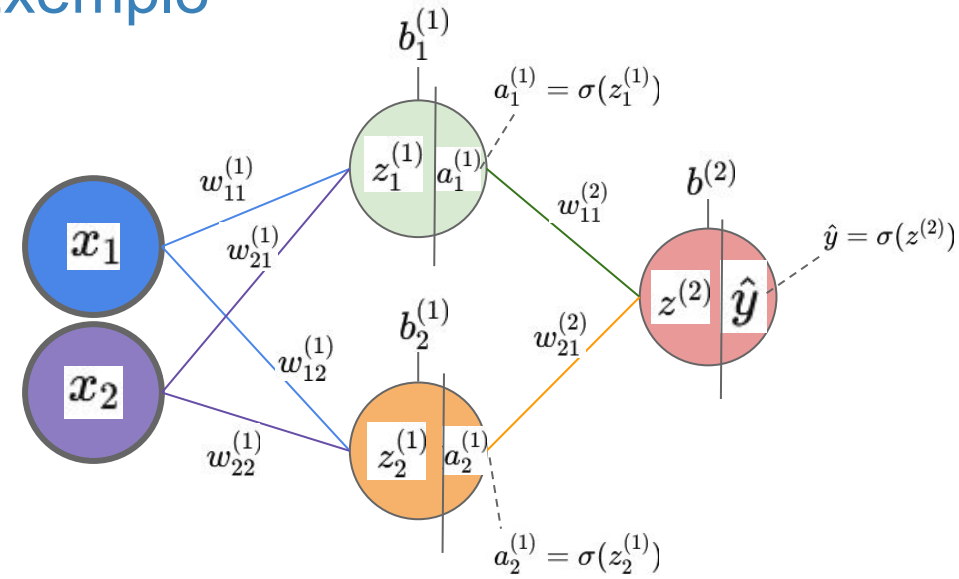
- Principal algoritmo de otimização para minimizar funções de perda ajustando os parâmetros do modelo (pesos e bias)

$$w_{t+1} = w_t - \alpha \cdot \nabla L(w_t)$$

Regra da cadeia: permite calcular a derivada de funções compostas (funções que dependem de outras funções)

$$f(g(x)) \quad \frac{df}{dx} = \frac{df}{dg} \cdot \frac{dg}{dx}$$

Exemplo

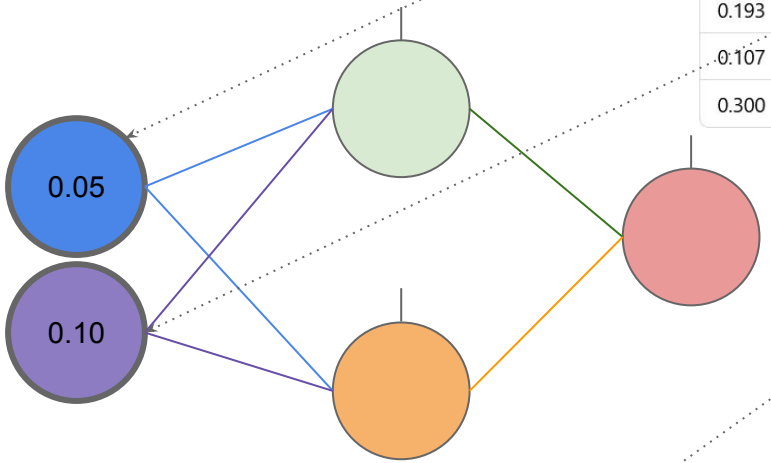


Função de ativação sigmoide: $\sigma(z) = \frac{1}{1 + e^{-z}}$

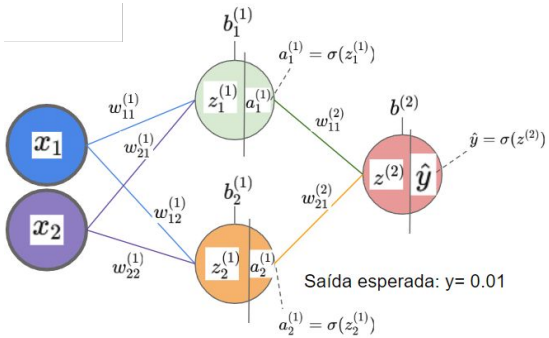
Saída esperada: y= 0.01

Exemplo

Idade Normalizada	Percentual de Renda Comprometido	Inadimplência Provável (Saída Esperada)
0.050	0.10	0.01
0.121	0.30	0.25
0.079	0.05	0.00
0.193	0.50	0.60
0.107	0.80	0.85
0.300	0.20	0.10



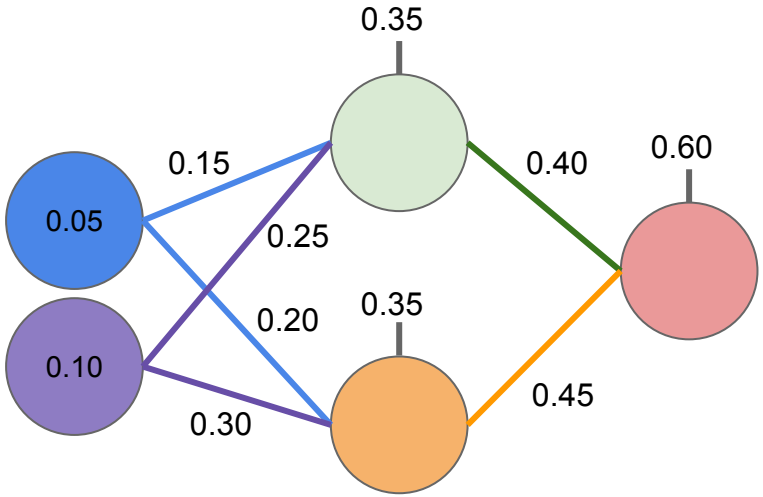
Tarefa de regressão!



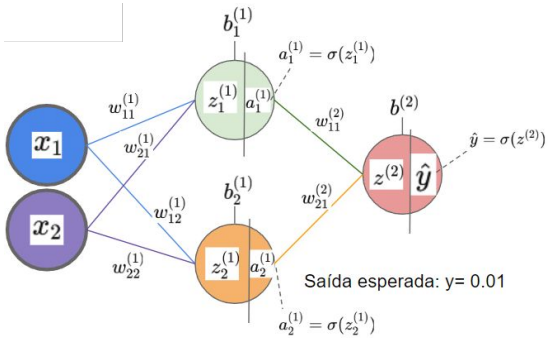
Função de ativação sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$

Saída esperada: y= 0.01

Exemplo



1. Inicialização dos pesos e biases (aleatório)

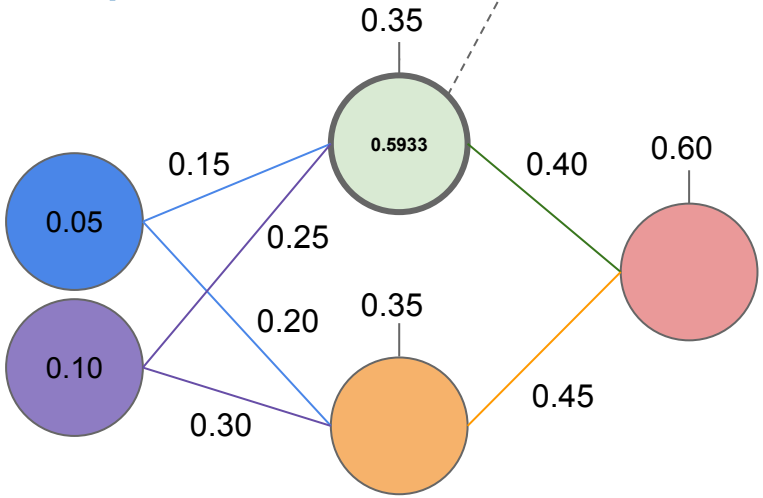


Função de ativação sigmoide: $\sigma(z) = \frac{1}{1 + e^{-z}}$

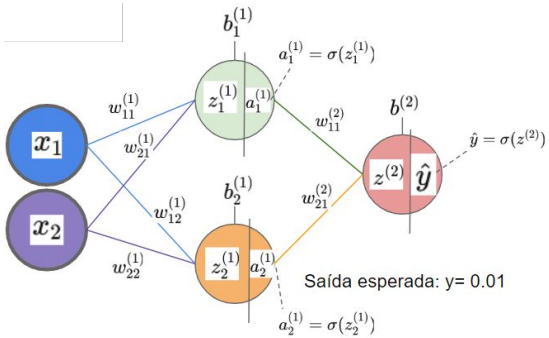
Saída esperada: y= 0.01

Exemplo

f(0.05*0.15 + 0.10*0.25 + 0.35) = **f**(0.3775)
~ 0.5933



1. Inicialização dos pesos e biases (aleatório)
2. Feedforward

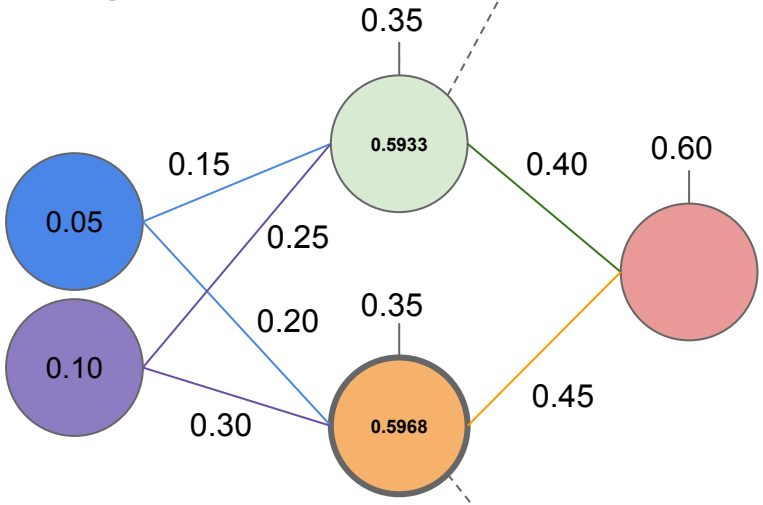


Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1 + e^{-z}}$

Saída esperada: y= 0.01

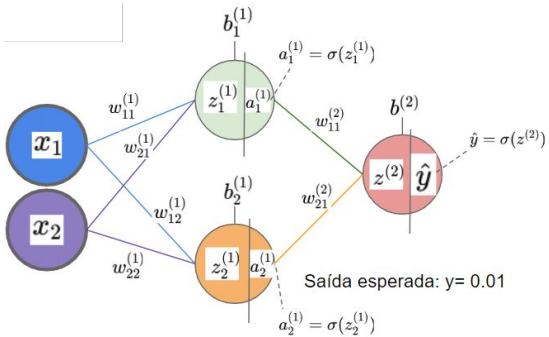
Exemplo

f(0.05*0.15 + 0.10*0.25 + 0.35) = **f**(0.3775)
~ 0.5933



1. Inicialização dos pesos e biases (aleatório)
2. Feedforward

f(0.05*0.20 + 0.10*0.30 + 0.35) = **f**(0.392)
~ 0.5968

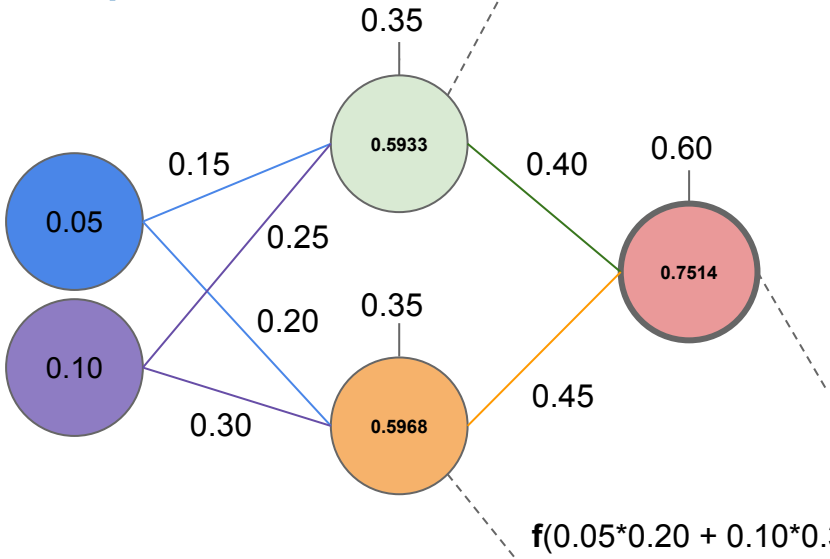


Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$

Saída esperada: y= 0.01

Exemplo

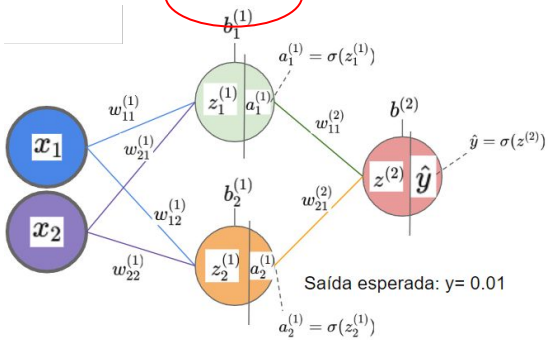
f(0.05*0.15 + 0.10*0.25 + 0.35) = **f**(0.3775)
~ 0.5933



1. Inicialização dos pesos e biases (aleatório)
2. Feedforward

f(0.5933*0.40 +0.5968*0.45 + 0.60) = **f**(1.1059)
~ 0.7514

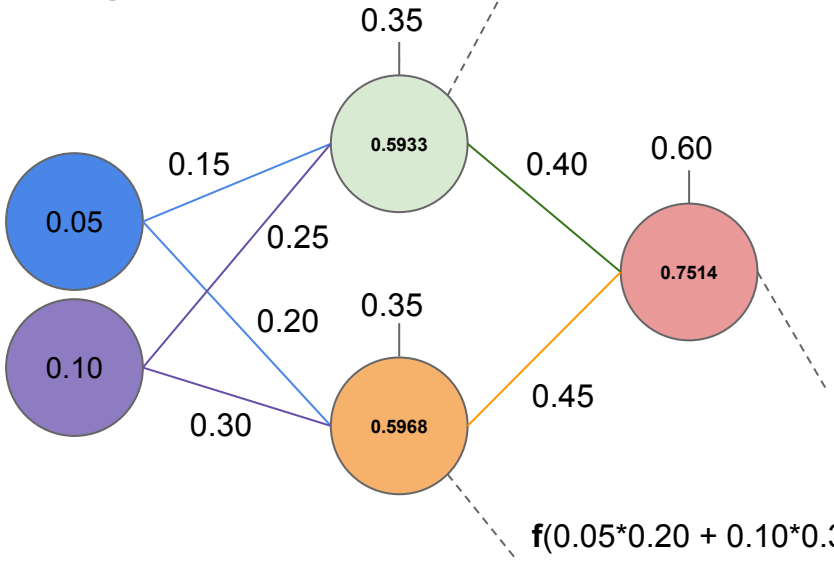
f(0.05*0.20 + 0.10*0.30 + 0.35) = **f**(0.392)
~ 0.5968



Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$

Saída esperada: y= 0.01

Exemplo



f(0.05*0.15 + 0.10*0.25 + 0.35) = **f**(0.3775)
~ 0.5933

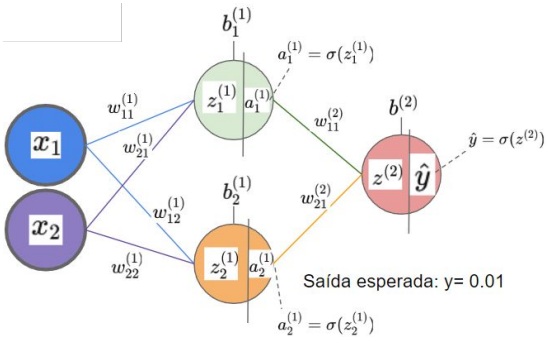
f(0.5933*0.40 +0.5968*0.45 + 0.60) = **f**(1.1059)
~ 0.7514

f(0.05*0.20 + 0.10*0.30 + 0.35) = **f**(0.392)
~ 0.5968

MAE (Mean Absolute Error)

$$L = |y - \hat{y}| = |0.01 - 0.7514| = 0.7414$$

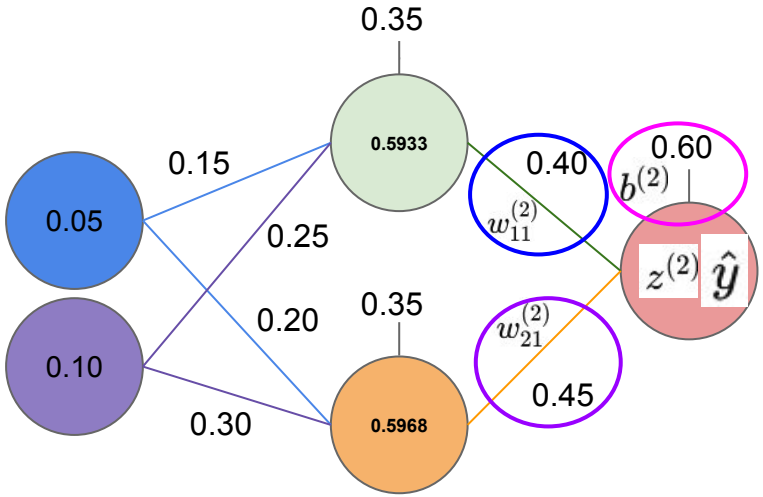
1. Inicialização dos pesos e biases (aleatório)
2. Feedforward
3. **Cálculo do erro (loss)**



Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$

Saída esperada: y= 0.01

Exemplo



Objetivo é calcular:

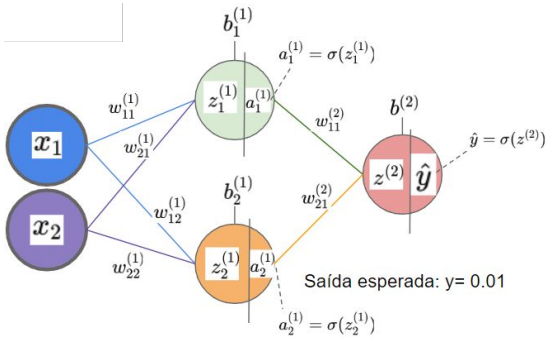
$\frac{\partial \mathcal{L}}{\partial w_{11}^{(2)}}$ $\frac{\partial \mathcal{L}}{\partial w_{21}^{(2)}}$ $\frac{\partial \mathcal{L}}{\partial b^{(2)}}$

1. Inicialização dos pesos e biases (aleatório)
2. Feedforward
3. Cálculo do erro (loss) MSE:
4. **Backpropagation**

$$\frac{\partial L}{\partial z^{(2)}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{(2)}}$$

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$$

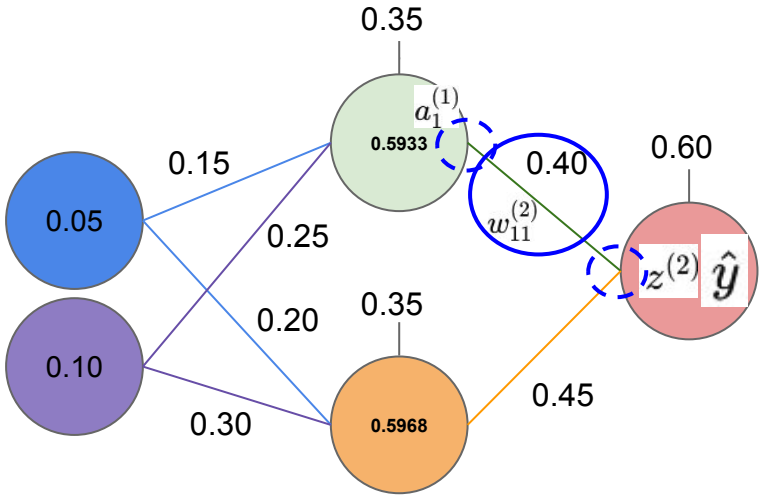
$$\frac{\partial \hat{y}}{\partial z^{(2)}} = \hat{y} \cdot (1 - \hat{y})$$



Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1 + e^{-z}}$

Saída esperada: y= 0.01

Exemplo



Objetivo é calcular:

$\frac{\partial \mathcal{L}}{\partial w_{11}^{(2)}}$ $\frac{\partial \mathcal{L}}{\partial w_{21}^{(2)}}$ $\frac{\partial \mathcal{L}}{\partial b^{(2)}}$

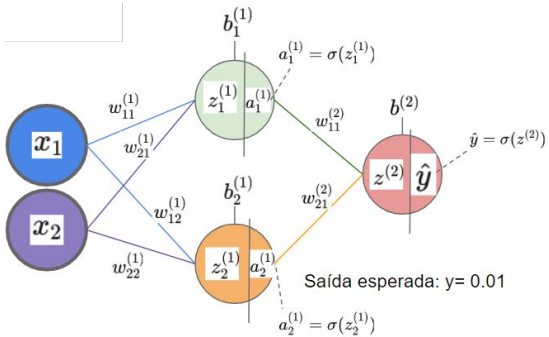
1. Inicialização dos pesos e biases (aleatório)
2. Feedforward
3. Cálculo do erro (loss) MSE:
4. **Backpropagation**

$$\frac{\partial \mathcal{L}}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \cdot a_1^{(1)}$$

$$\frac{\partial L}{\partial z^{(2)}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{(2)}}$$

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$$

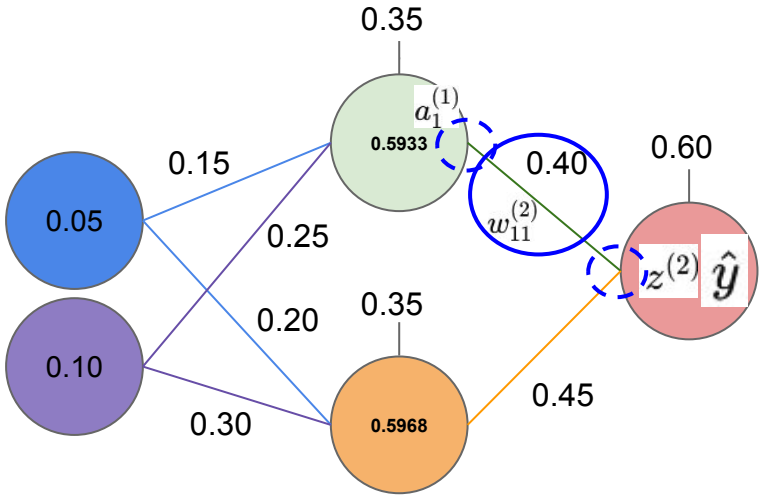
$$\frac{\partial \hat{y}}{\partial z^{(2)}} = \hat{y} \cdot (1 - \hat{y})$$



Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1 + e^{-z}}$

Saída esperada: y= 0.01

Exemplo



Objetivo é calcular:

$\frac{\partial \mathcal{L}}{\partial w_{11}^{(2)}}$ $\frac{\partial \mathcal{L}}{\partial w_{21}^{(2)}}$ $\frac{\partial \mathcal{L}}{\partial b^{(2)}}$

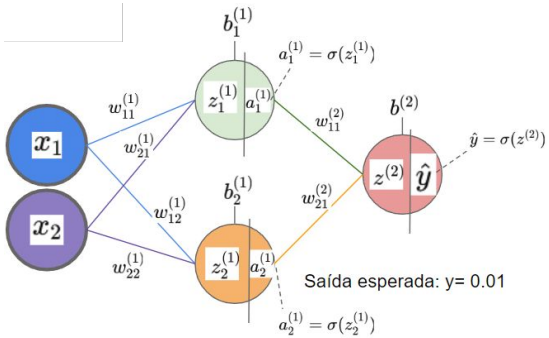
1. Inicialização dos pesos e biases (aleatório)
2. Feedforward
3. Cálculo do erro (loss) MSE:
4. **Backpropagation**

$$\frac{\partial \mathcal{L}}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \cdot a_1^{(1)}$$

$$\frac{\partial L}{\partial z^{(2)}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{(2)}}$$

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y$$

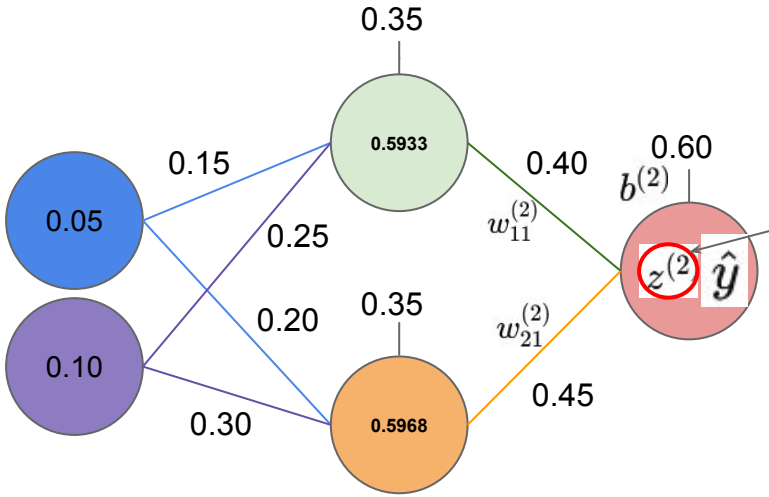
$$\frac{\partial \hat{y}}{\partial z^{(2)}} = \hat{y} \cdot (1 - \hat{y})$$



Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1 + e^{-z}}$

Saída esperada: y= 0.01

Exemplo



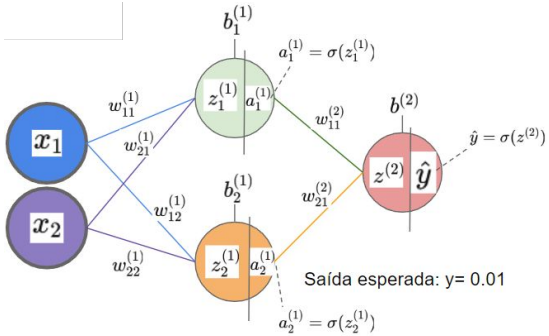
1. Inicialização dos pesos e biases (aleatório)
2. Feedforward
3. Cálculo do erro (loss) MSE:
4. **Backpropagation**

$$\frac{\partial L}{\partial z^{(2)}} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial z^{(2)}} = 0.7414 \cdot 0.1860 \approx 0.1378$$

Regra da cadeia

$$\frac{\partial L}{\partial \hat{y}} = \hat{y} - y = 0.7514 - 0.01 = 0.7414$$

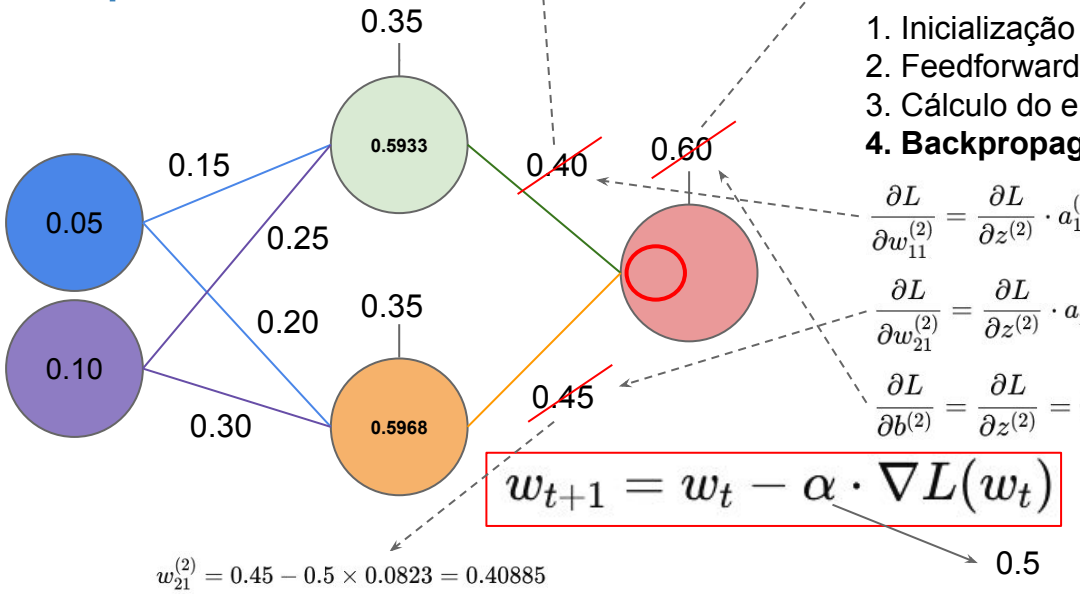
$$\frac{\partial \hat{y}}{\partial z^{(2)}} = \hat{y} \cdot (1 - \hat{y}) = 0.7514 \cdot (1 - 0.7514) \approx 0.1860$$



Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1 + e^{-z}}$

Saída esperada: y= 0.01

Exemplo



$w_{11}^{(2)} = 0.40 - 0.5 \times 0.0818 = 0.3591$

$b^{(2)} = 0.60 - 0.5 \times 0.1378 = 0.5311$

$w_{21}^{(2)} = 0.45 - 0.5 \times 0.0823 = 0.40885$

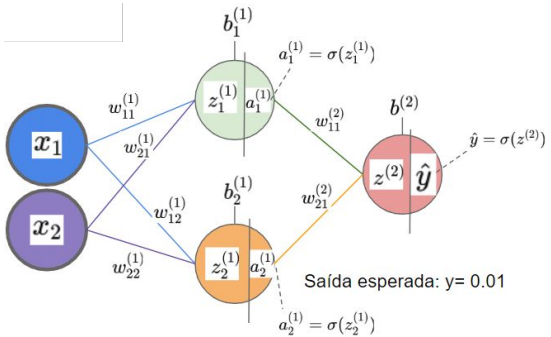
1. Inicialização dos pesos e biases (aleatório)
2. Feedforward
3. Cálculo do erro (loss) MSE:
4. **Backpropagation**

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \cdot a_1^{(1)} = 0.1378 \cdot 0.5933 \approx 0.0818 \qquad \frac{\partial L}{\partial z^{(2)}} = 0.1378$$

$$\frac{\partial L}{\partial w_{21}^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \cdot a_2^{(1)} = 0.1378 \cdot 0.5968 \approx 0.0823$$

$$\frac{\partial L}{\partial b^{(2)}} = \frac{\partial L}{\partial z^{(2)}} = 0.1378$$

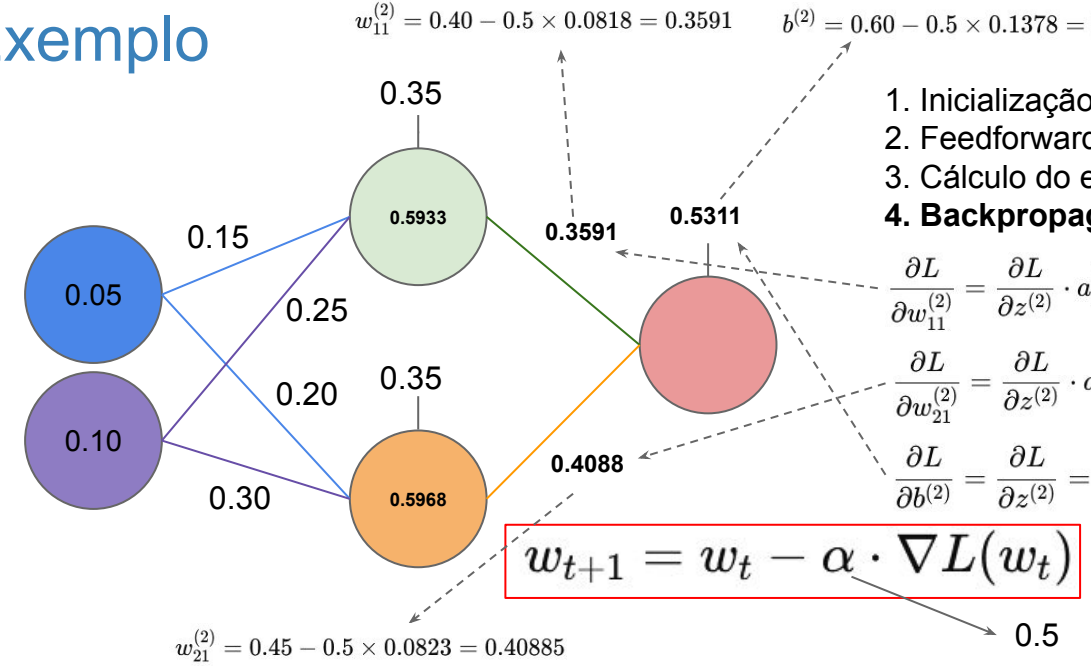
$$w_{t+1} = w_t - \alpha \cdot \nabla L(w_t)$$



Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1 + e^{-z}}$

Saída esperada: y= 0.01

Exemplo



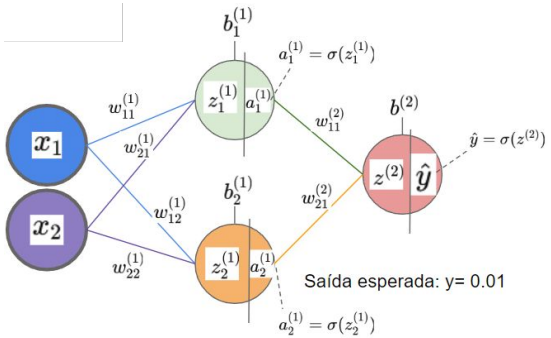
1. Inicialização dos pesos e biases (aleatório)
2. Feedforward
3. Cálculo do erro (loss) MSE:
4. **Backpropagation**

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \cdot a_1^{(1)} = 0.1378 \cdot 0.5933 \approx 0.0818 \qquad \frac{\partial L}{\partial z^{(2)}} = 0.1378$$

$$\frac{\partial L}{\partial w_{21}^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \cdot a_2^{(1)} = 0.1378 \cdot 0.5968 \approx 0.0823$$

$$\frac{\partial L}{\partial b^{(2)}} = \frac{\partial L}{\partial z^{(2)}} = 0.1378$$

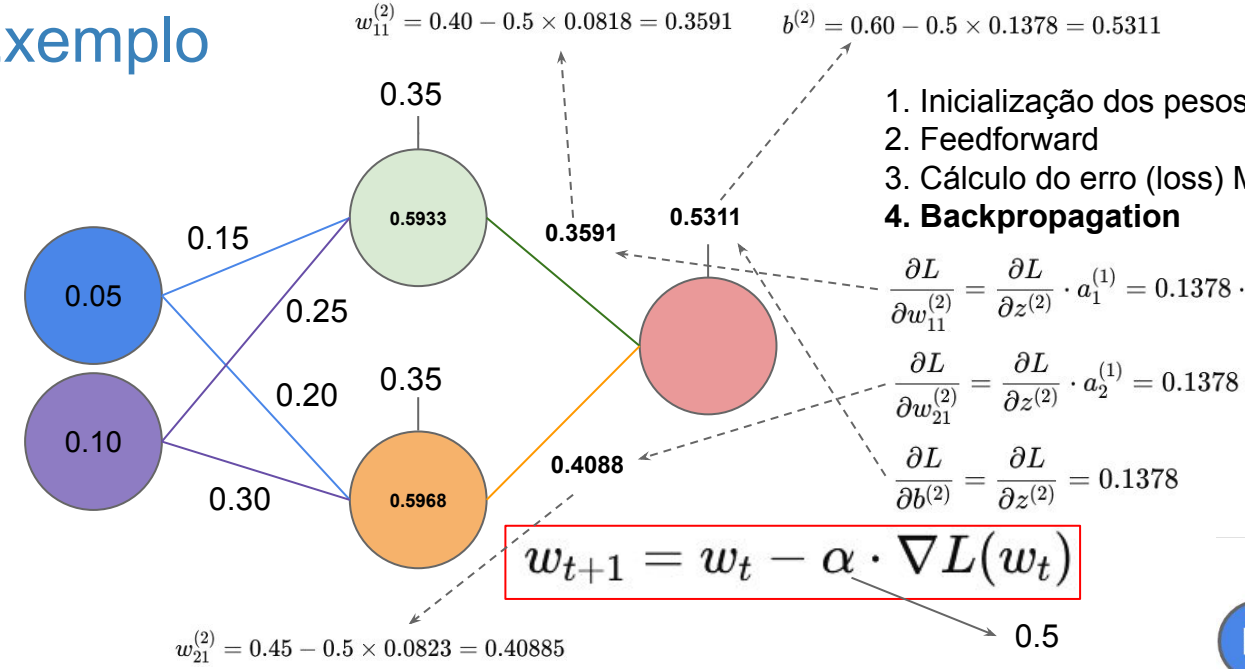
$$w_{t+1} = w_t - \alpha \cdot \nabla L(w_t)$$



Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$

Saída esperada: y= 0.01

Exemplo



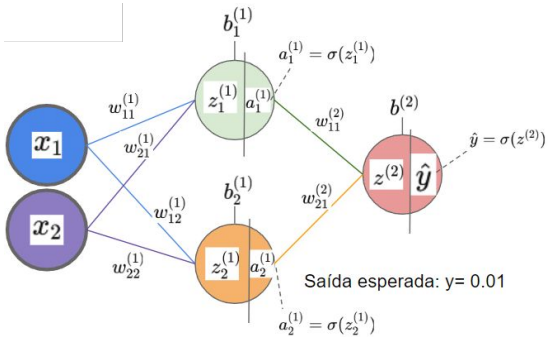
1. Inicialização dos pesos e biases (aleatório)
2. Feedforward
3. Cálculo do erro (loss) MSE:
4. **Backpropagation**

$$\frac{\partial L}{\partial w_{11}^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \cdot a_1^{(1)} = 0.1378 \cdot 0.5933 \approx 0.0818 \qquad \frac{\partial L}{\partial z^{(2)}} = 0.1378$$

$$\frac{\partial L}{\partial w_{21}^{(2)}} = \frac{\partial L}{\partial z^{(2)}} \cdot a_2^{(1)} = 0.1378 \cdot 0.5968 \approx 0.0823$$

$$\frac{\partial L}{\partial b^{(2)}} = \frac{\partial L}{\partial z^{(2)}} = 0.1378$$

$$w_{t+1} = w_t - \alpha \cdot \nabla L(w_t)$$



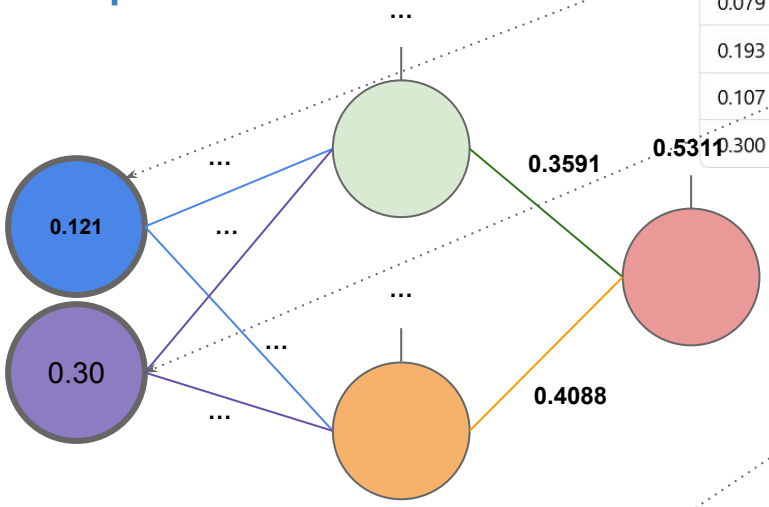
Função **f** de ativação sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$

Saída esperada: y= 0.01

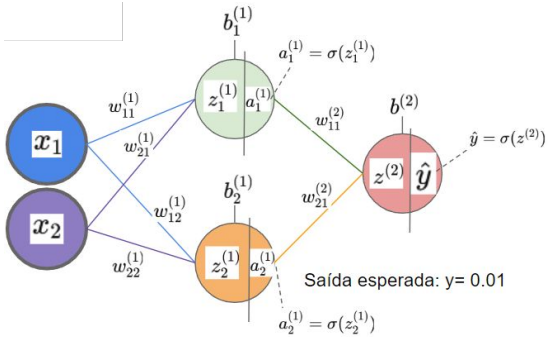
Os pesos continuam sendo ajustados por backpropagation até o início da rede!

Exemplo

Idade Normalizada	Percentual de Renda Comprometido	Inadimplência Provável (Saída Esperada)
0.050	0.10	0.01
0.121	0.30	0.25
0.079	0.05	0.00
0.193	0.50	0.60
0.107	0.80	0.85
0.300	0.20	0.10



Agora o processo deve ser repetido para todos os dados disponíveis (mas os pesos/biases são mantidos!)

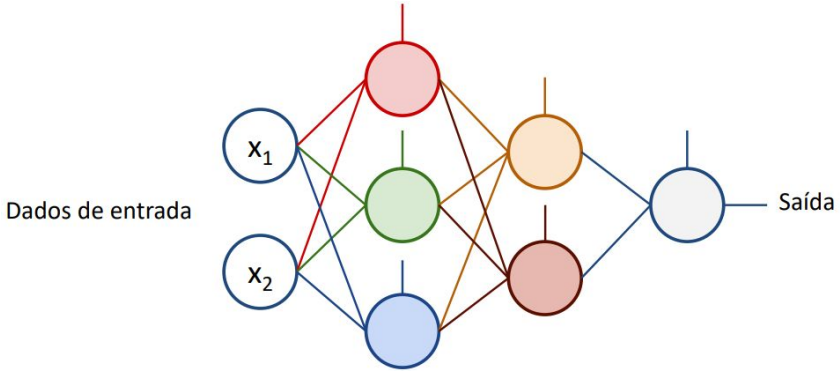


Função de ativação sigmoide: $\sigma(z) = \frac{1}{1+e^{-z}}$

Saída esperada: y= 0.25

Resumo - Treinamento

- Repetir até atingir critério de parada:
  - Para cada instância (x,y) no dataset
  - Submeter x na rede e propagar as ativações (forward pass)
  - Computar o erro da saída (diferença entre predição e y)
  - Retropropagar os erros (backpropagation)
  - Ajustar os parâmetros (gradiente descendente)



Referências

- Curso Redes Neurais Artificiais do canal Canal Machine Learning para humanos do Youtube:
  - https://www.youtube.com/watch?v=ebToEXQFCo4&list=PLQH6T1jnIb5J7vugBAauJsFU8Qgvuf-4X

- Material do Prof. Joel (disponível no Moodle)

- A step-by-step backpropagation example:
  - https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Próximas aulas

- Aula teórica:
  - Redes neurais para textos [2]

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
Instituto de Informática
Departamento de Informática Aplicada

Obrigado pela atenção!
Dúvidas?

Prof. Dennis Giovani Balreira
(Material adaptado do Prof. Joel Carbonera e Canal Machine Learning para humanos)

INF01221 - Tópicos Especiais em Computação XXXVI:
Processamento de Linguagem Natural