UNICAMP

INF-0618
Tópicos em Aprendizado de Máquina II

Aula 1 – Redes Neurais

Profa. Fernanda Andaló

2018

Instituto de Computação - Unicamp

Roteiro

Motivação

Aproximando funções lineares

Adicionando não-linearidade

Treinando uma rede neural

Prática

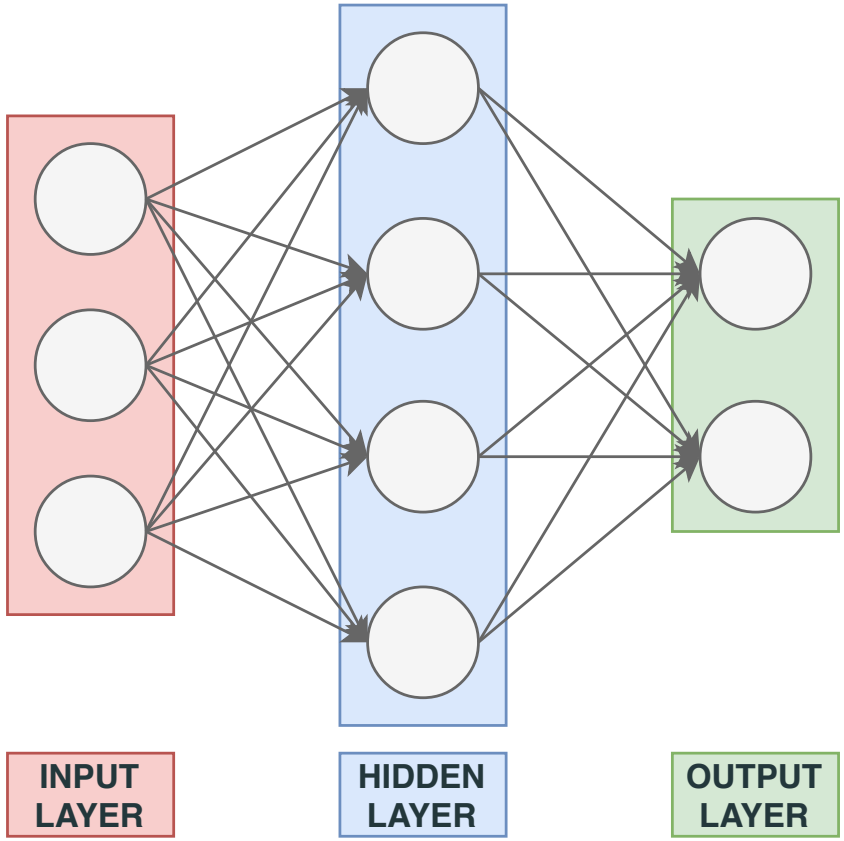
Motivação

Motivação

Rede Neural Artificial (ANN)

Conjunto de algoritmos capazes de aproximar funcões contínuas arbitrárias.

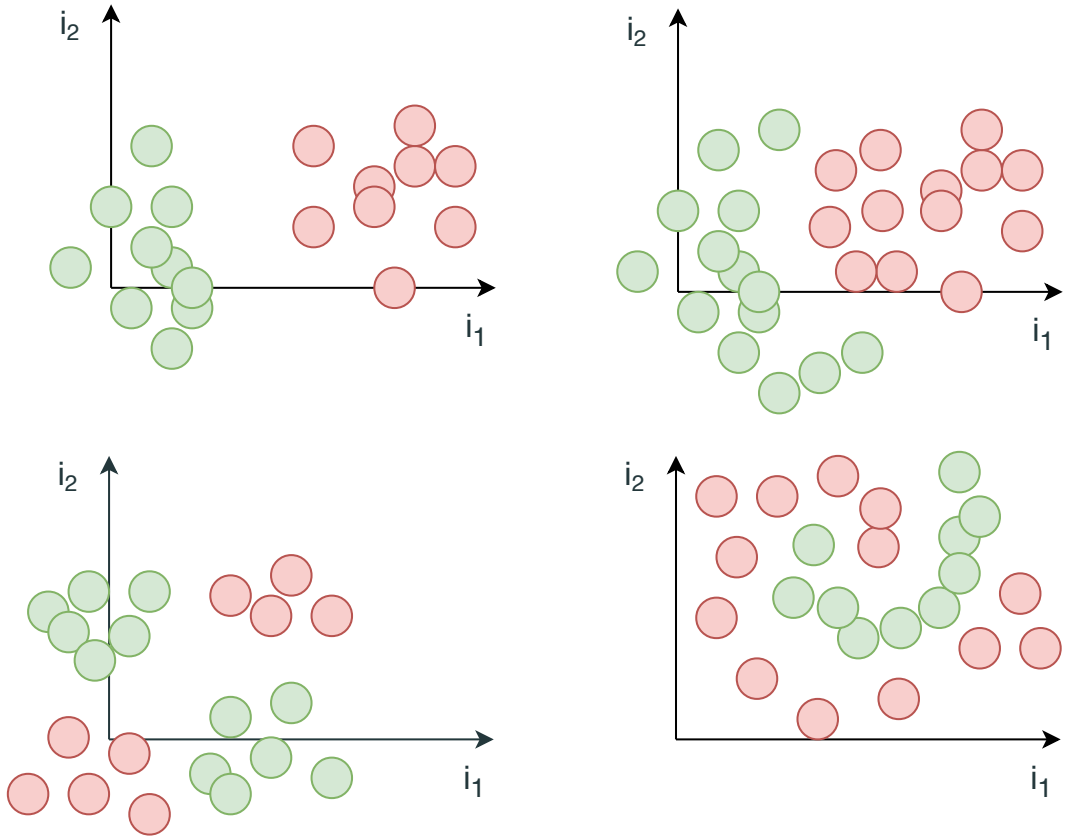
ANNs são formadas por camadas (*layers*) de entrada (*input*), saída (*output*) e intermediárias (*hidden*), construídas por nós (ou neurônios) interconectados.



Motivação

Uma ANN pode aproximar uma função $f(x)$ que leva um sample x a um valor utilizado para classificação.

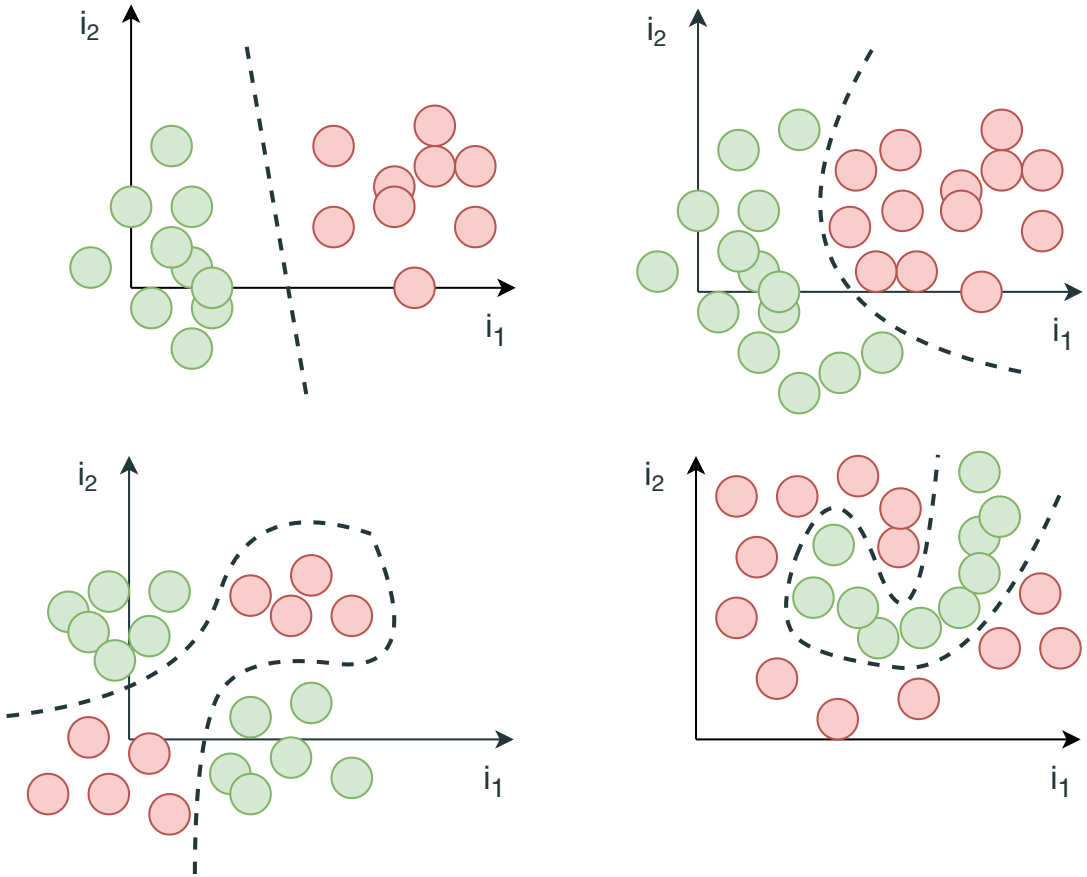
Exemplos:



Motivação

Uma ANN pode aproximar uma função $f(x)$ que leva um sample x a um valor utilizado para classificação.

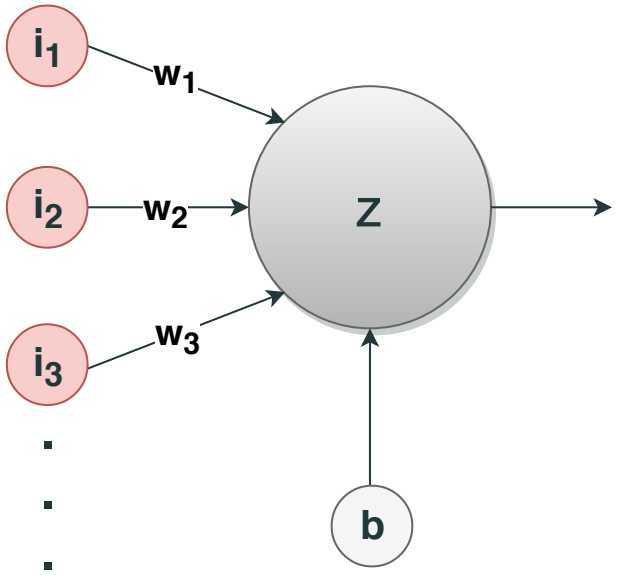
Exemplos:



Aproximando funções lineares

Aproximando funções lineares

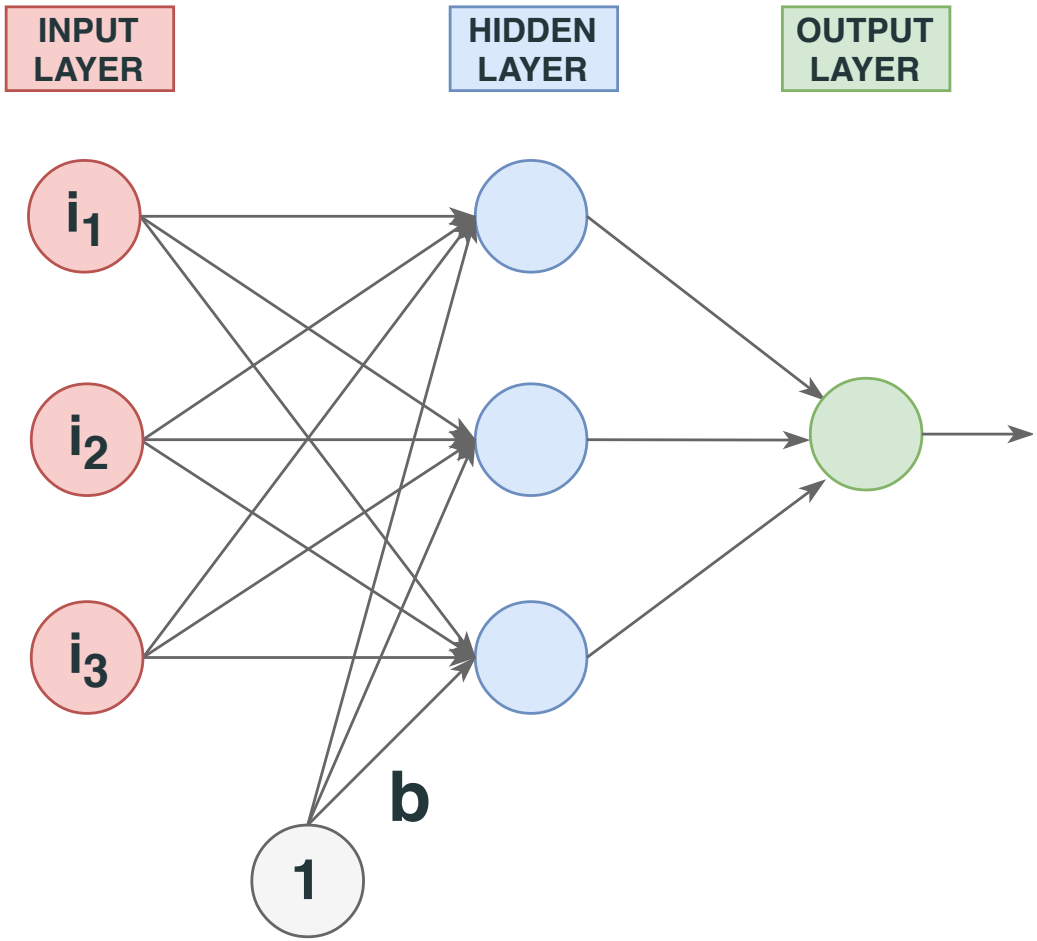
Neurônio



$$z = f(x_1, x_2, x_3, \ldots, x_n) = \sum_{j=1}^{n} w_j i_j + b = w \cdot x + b$$

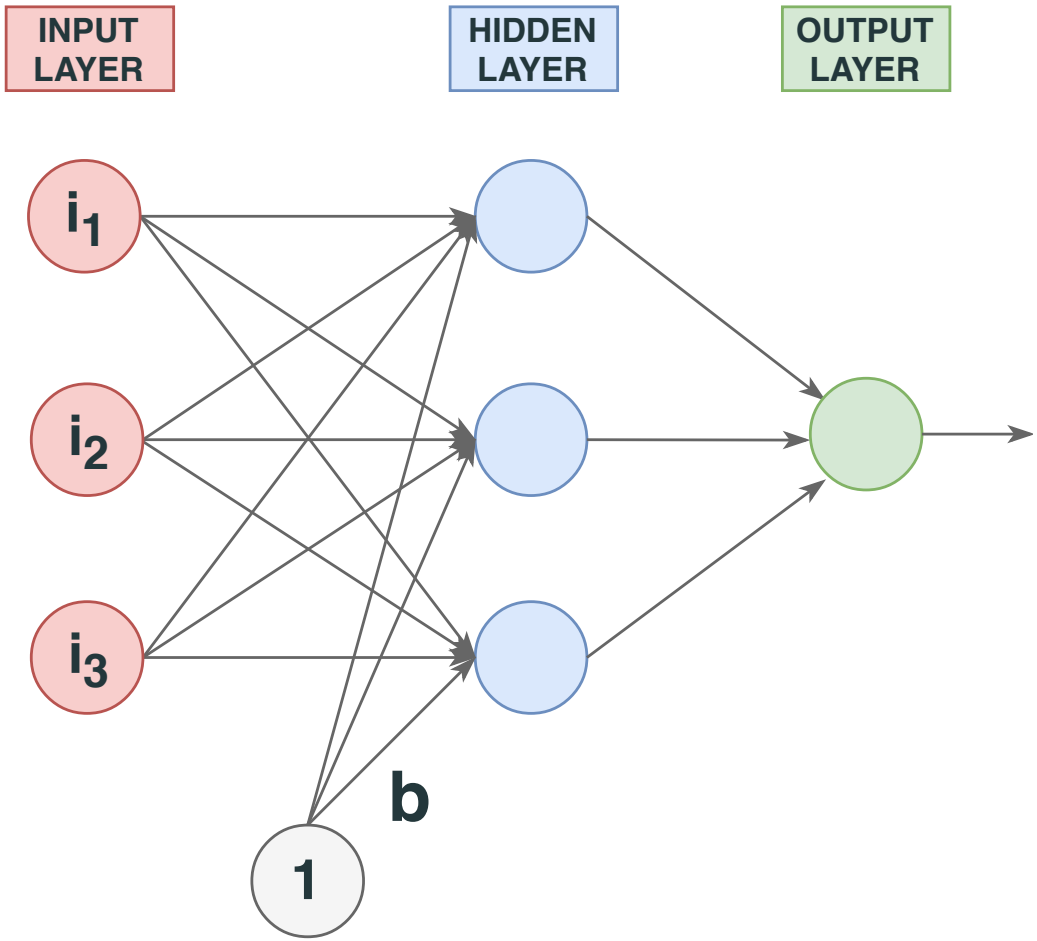
Aproximando funções lineares

Adicionando uma camada e mais neurônios...



Aproximando funções lineares

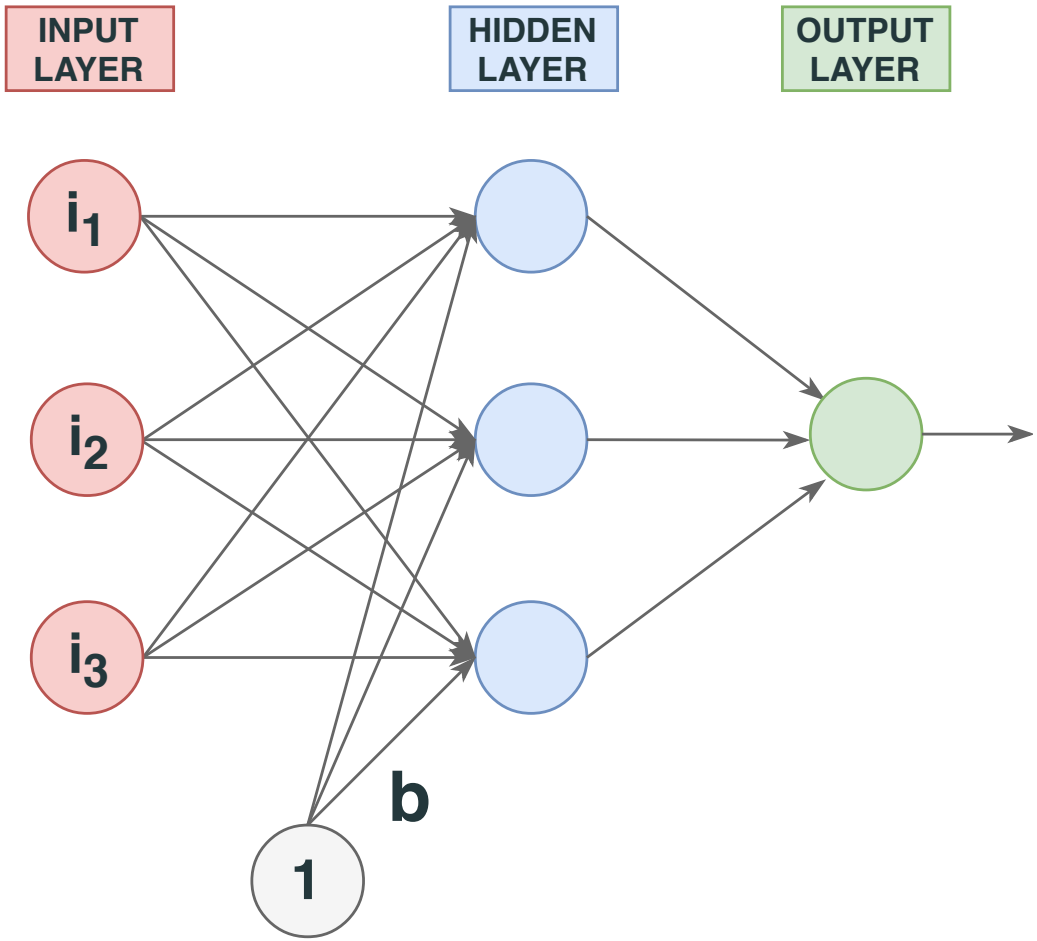
Adicionando uma camada e mais neurônios...
Esta ANN pode aproximar uma função não linear?



Aproximando funções lineares

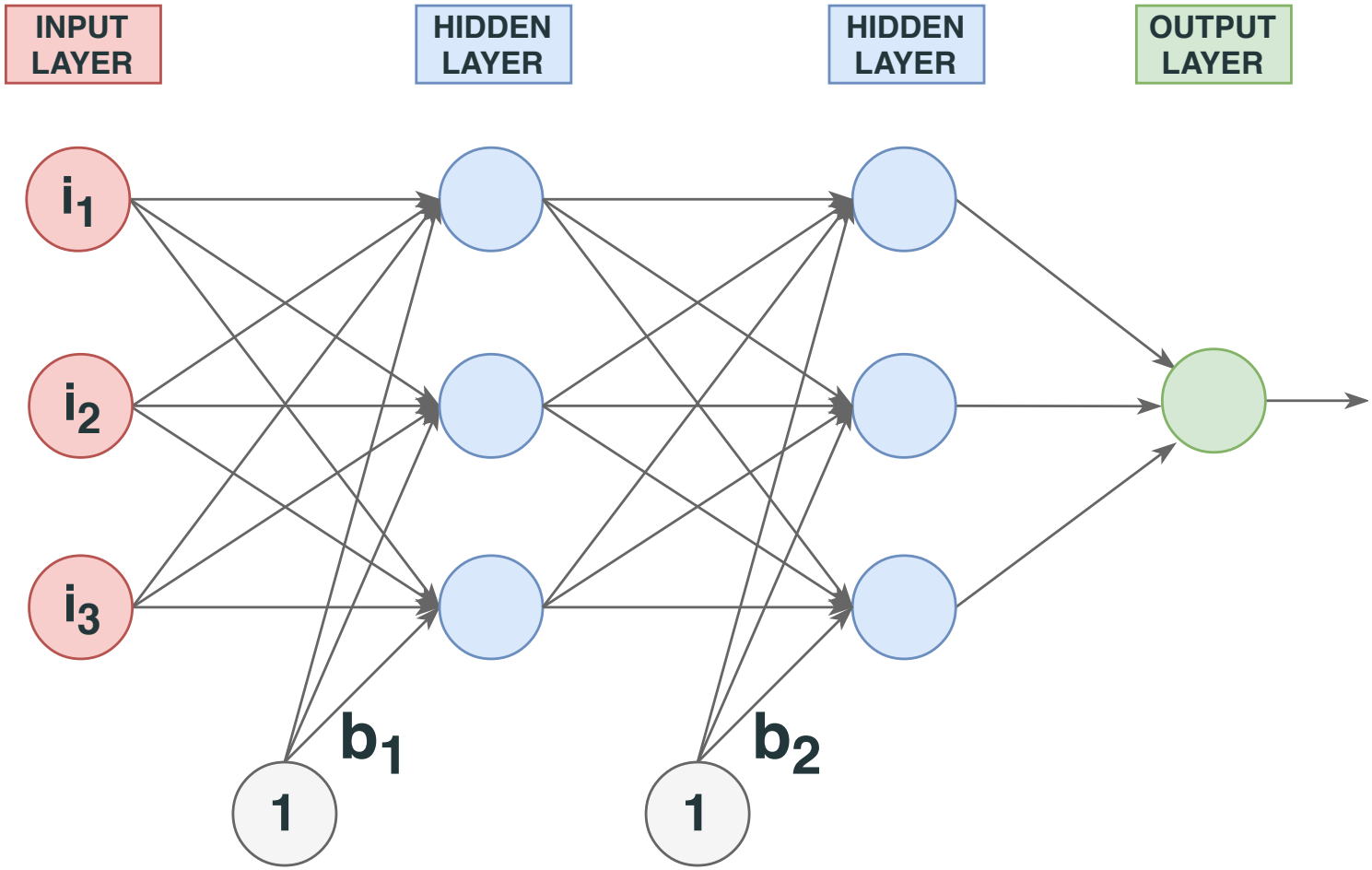
Adicionando uma camada e mais neurônios...

Esta ANN pode aproximar uma função não linear? Não!



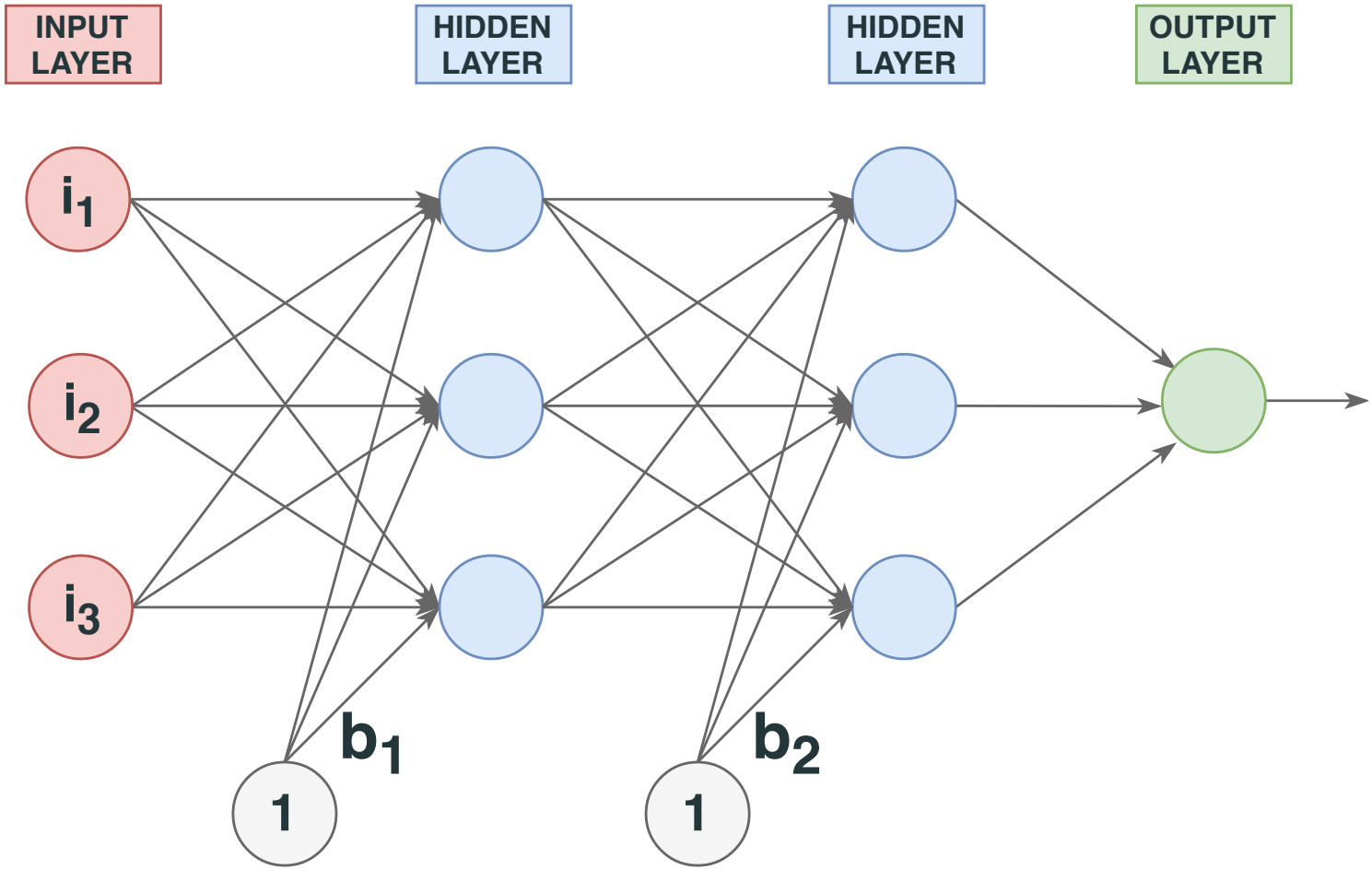
Aproximando funções lineares

Adicionando uma camada e mais neurônios...



Aproximando funções lineares

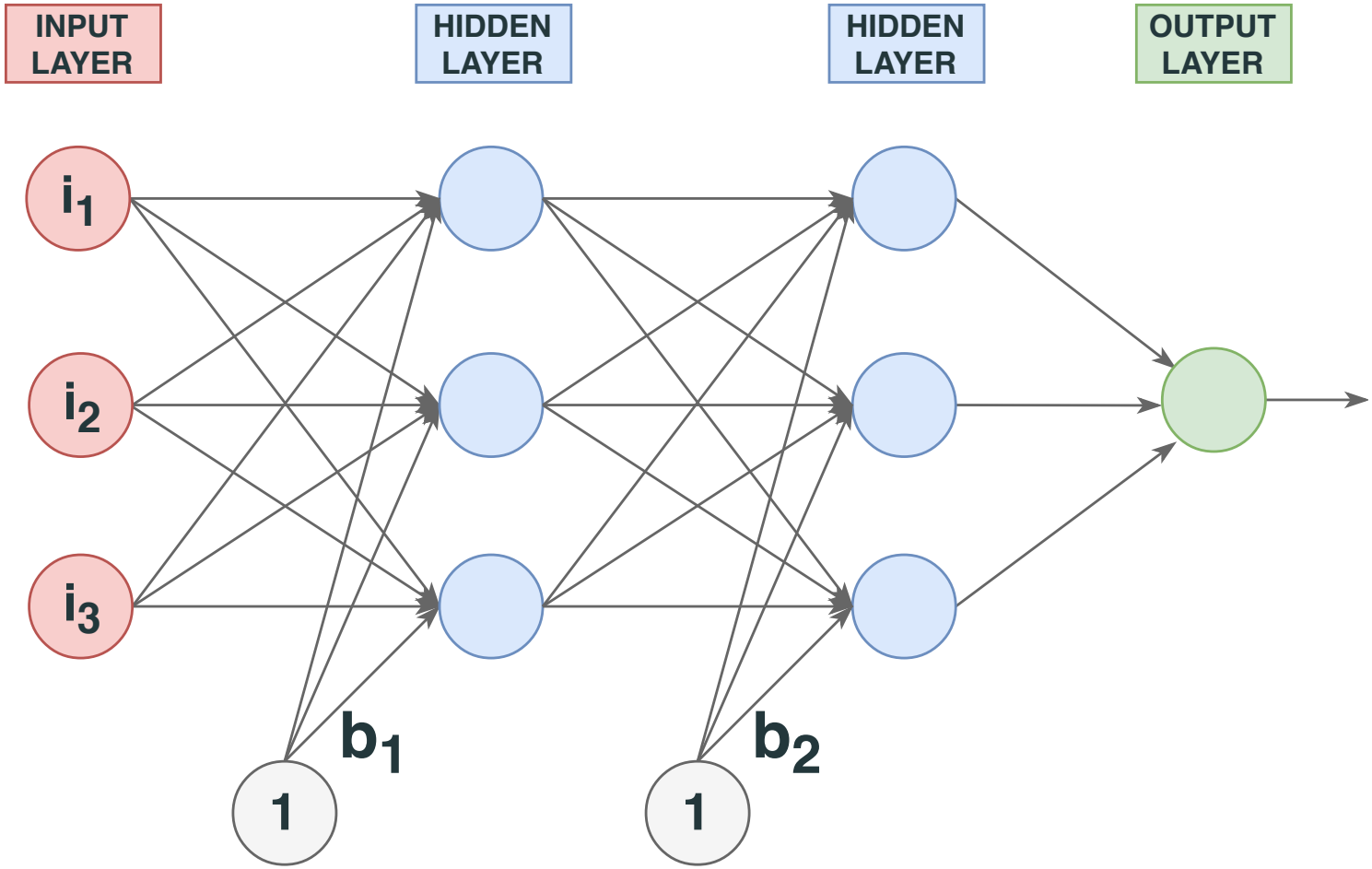
Adicionando uma camada e mais neurônios...
E esta ANN?



Aproximando funções lineares

Adicionando uma camada e mais neurônios...

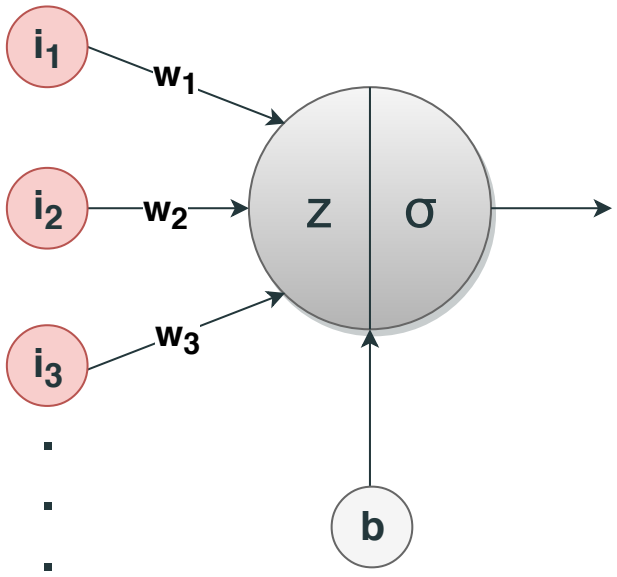
E esta ANN? Também não!



Adicionando não-linearidade

Adicionando não-linearidade

Neurônio



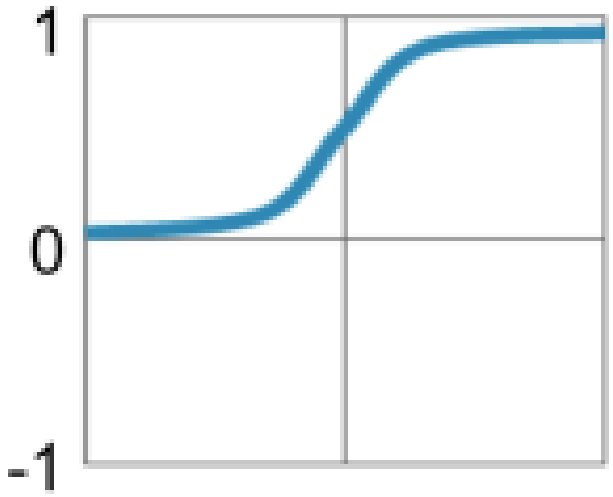
$$y = \sigma(f(w \cdot x + b)) = \sigma(z)$$

σ : função de ativação

Adicionando não-linearidade

Funções de ativação comuns

- Sigmóide

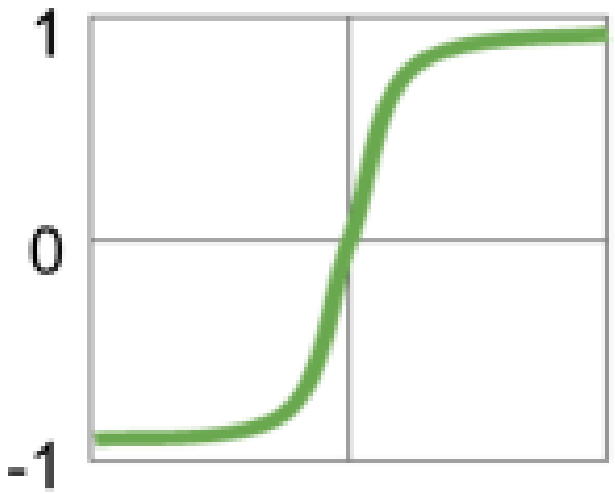


$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Adicionando não-linearidade

Funções de ativação comuns

- Tangente hiperbólica

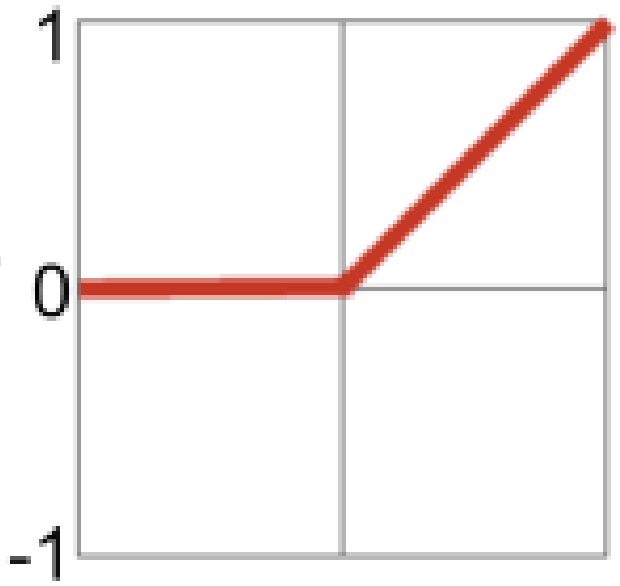


$$\sigma(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$$

Adicionando não-linearidade

Funções de ativação comuns

- Rectified Linear Unit (ReLU)

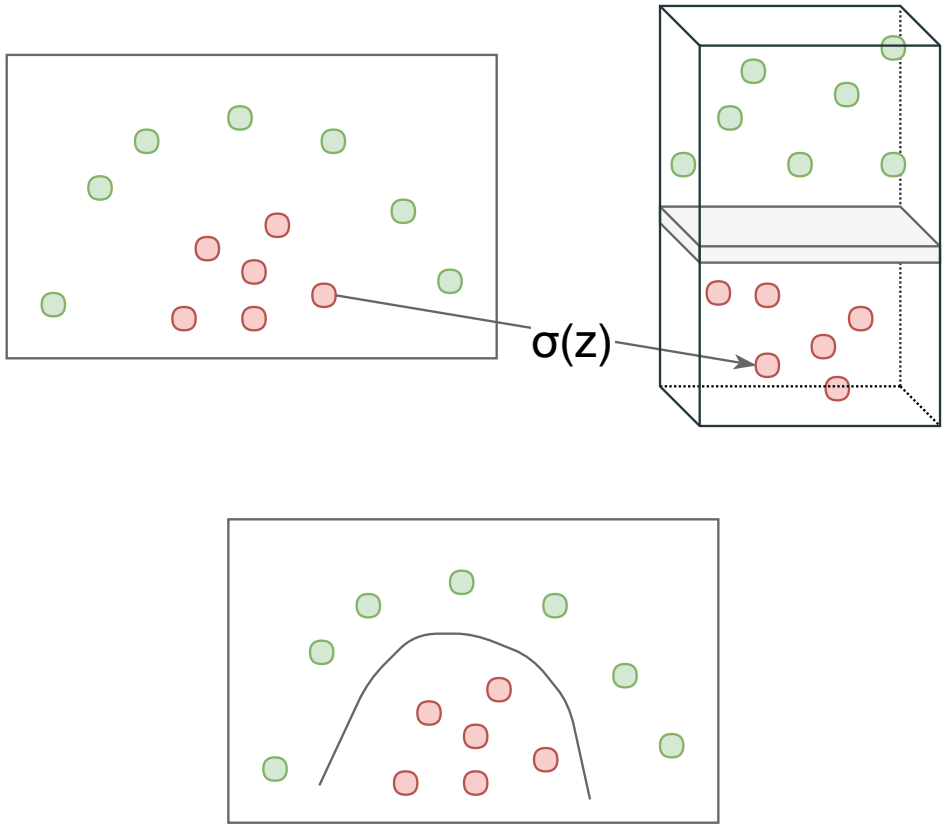


$$\sigma(z) = max(0, z)$$

Adicionando não-linearidade

Por que funções de ativação são importantes?

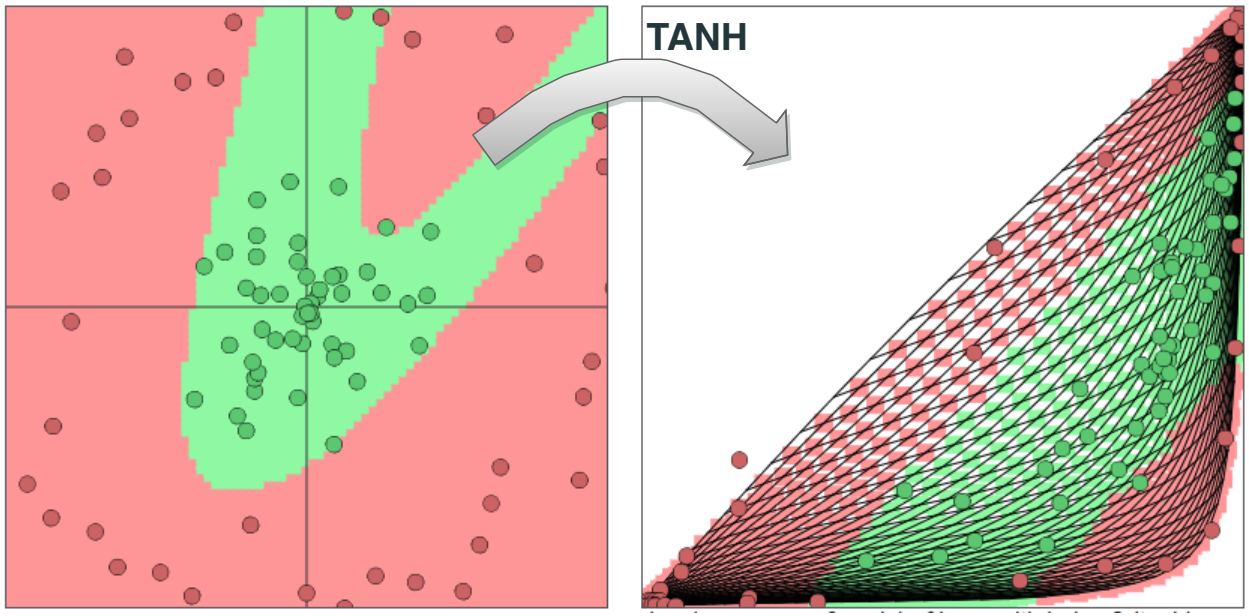
- Transformam os dados para facilitar a classificação.



Adicionando não-linearidade

Por que funções de ativação são importantes?

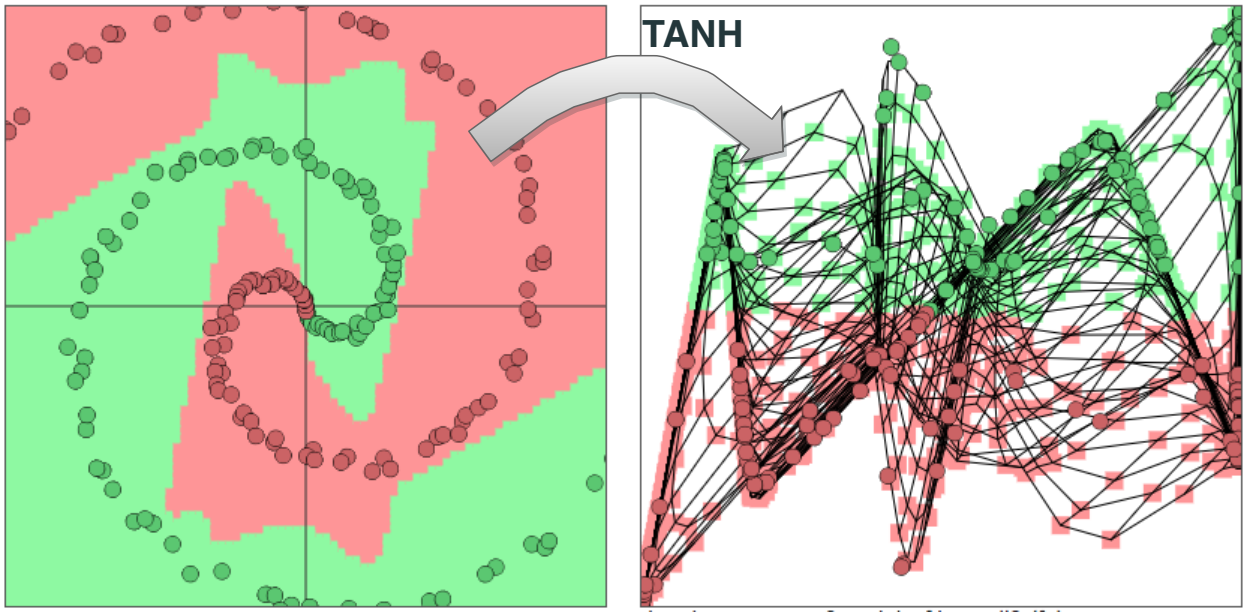
- Transformam os dados para facilitar a classificação.



Adicionando não-linearidade

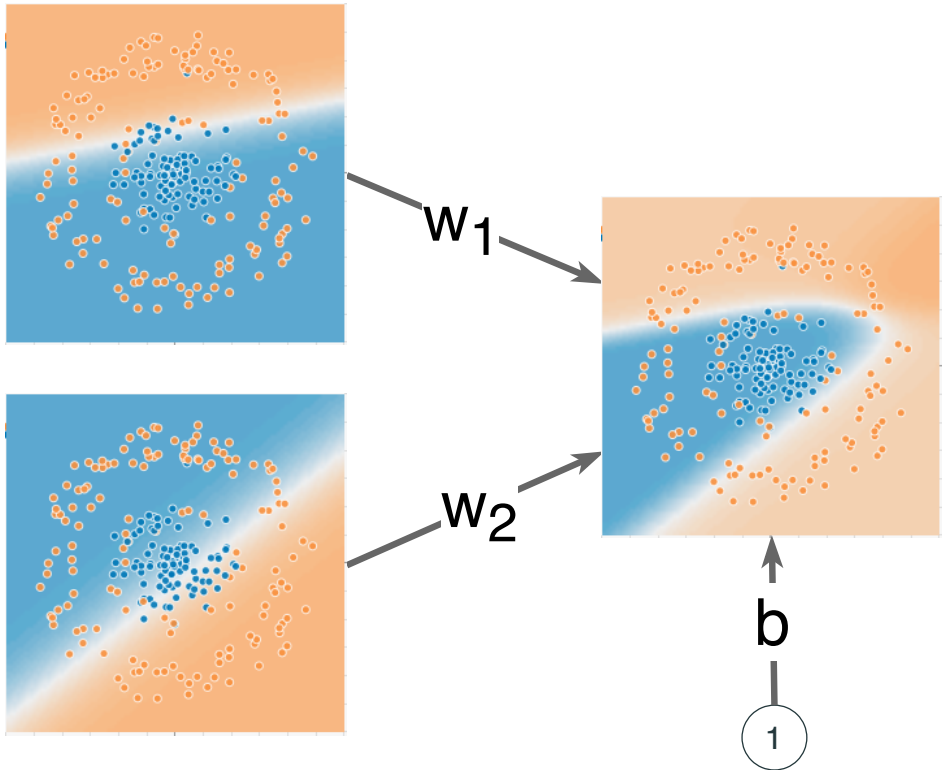
Por que funções de ativação são importantes?

- Transformam os dados para facilitar a classificação.



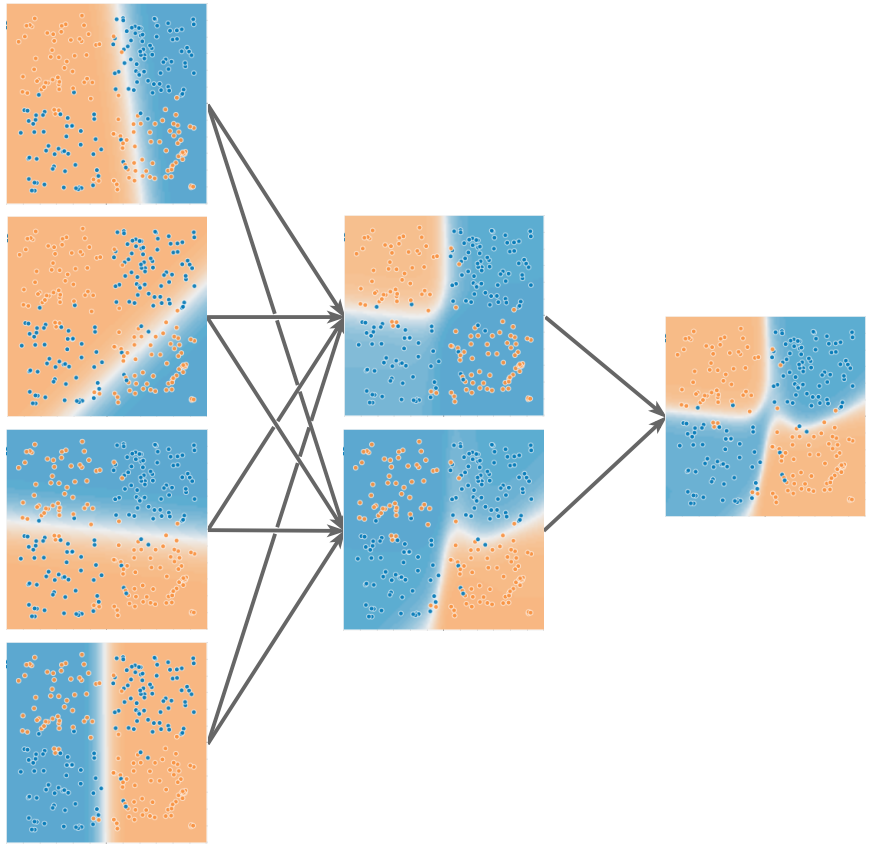
Por que funções de ativação são importantes?

- Permitem gerar funções mais complexas pela combinação de funções mais simples.



Por que funções de ativação são importantes?

- Permitem gerar funções mais complexas pela combinação de funções mais simples.



Treinando uma rede neural

Treinando uma rede neural

- Definir uma função de custo.
- Procurar o mínimo da função de custo, dados samples x_i e labels y_i: *backpropagation*.
- Backpropagation: *forward pass* e *backward pass*.

Treinando uma rede neural

Função de custo: medida para saber o erro da rede.

- MSE

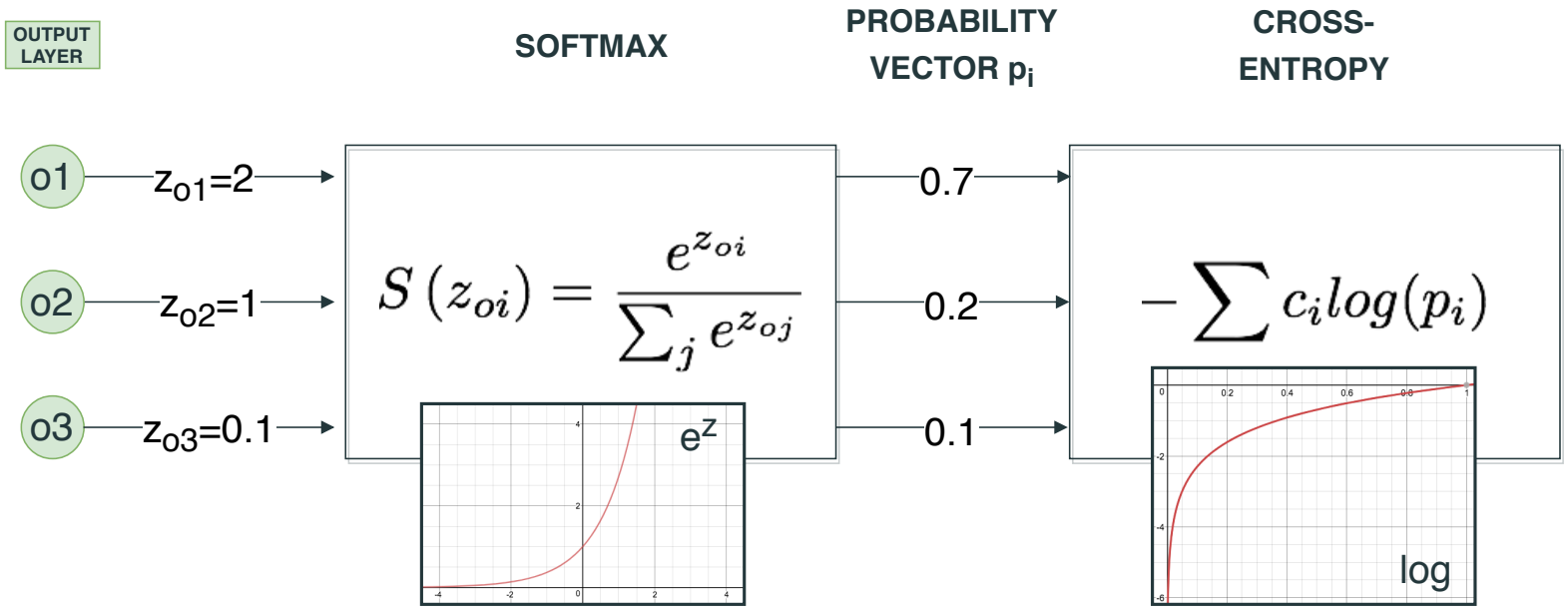
$$\frac{1}{n}\sum_{i=1}^{n}\frac{1}{2}\left(y_i - output_i\right)^2,$$

onde n é a quantidade de *samples* de treinamento, y_i é o valor esperado (*label*) e $output_i$ é o valor predito pela rede para o *sample* x_i.

Treinando uma rede neural

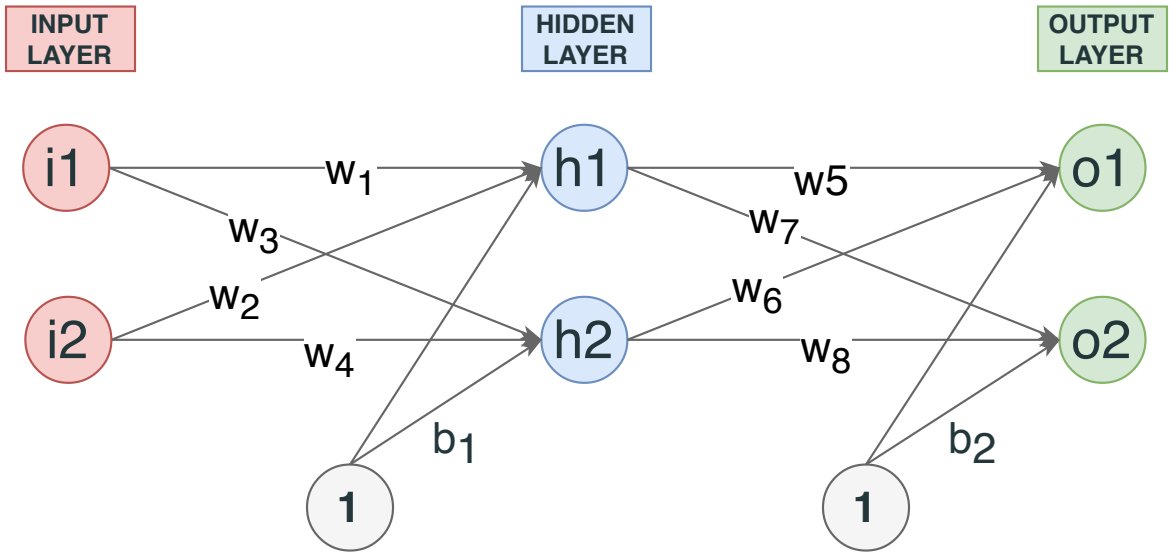
Função de custo: medida para saber o erro da rede.

- Cross-entropy



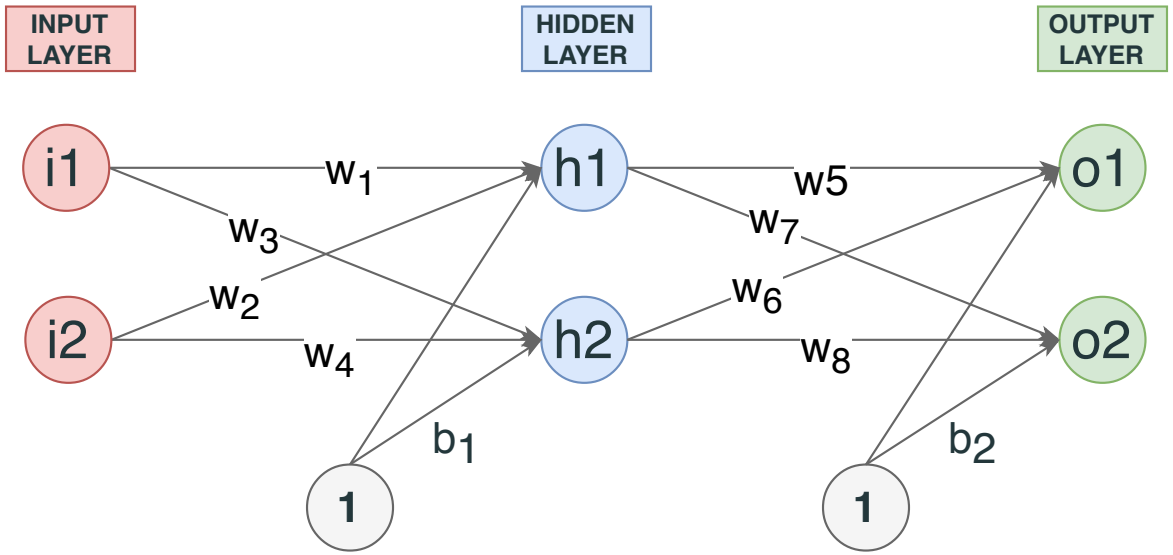
onde c_i é o *hot-encoded label* do sample x_i e p_i é o vetor de probabilidades *softmax* retornado para o sample x_i.

Exemplo de Backpropagation



- Função de ativação σ: $\dfrac{1}{1+e^{-z}}$
- Função de custo:
  $C = C_{o1} + C_{o2} = \frac{1}{2}(y_{o1} - \sigma_{o_1})^2 + \frac{1}{2}(y_{o2} - \sigma_{o2})^2$

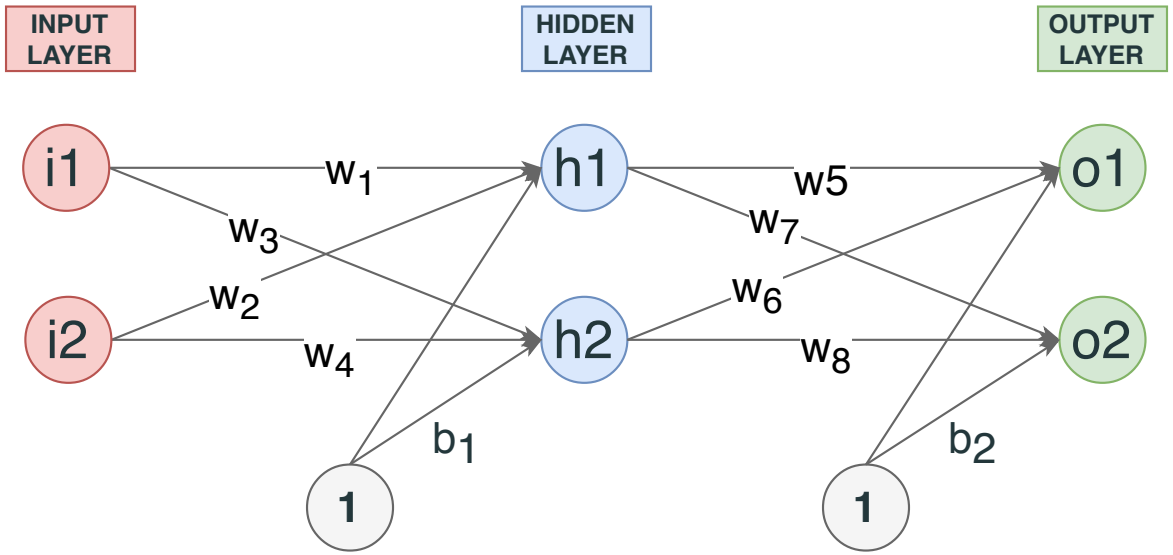
Exemplo de Backpropagation



Forward pass:

- $z_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$
- $\sigma_{h1} = \dfrac{1}{1 + e^{-z_{h1}}}$
- Mesmo cálculo para $h2$

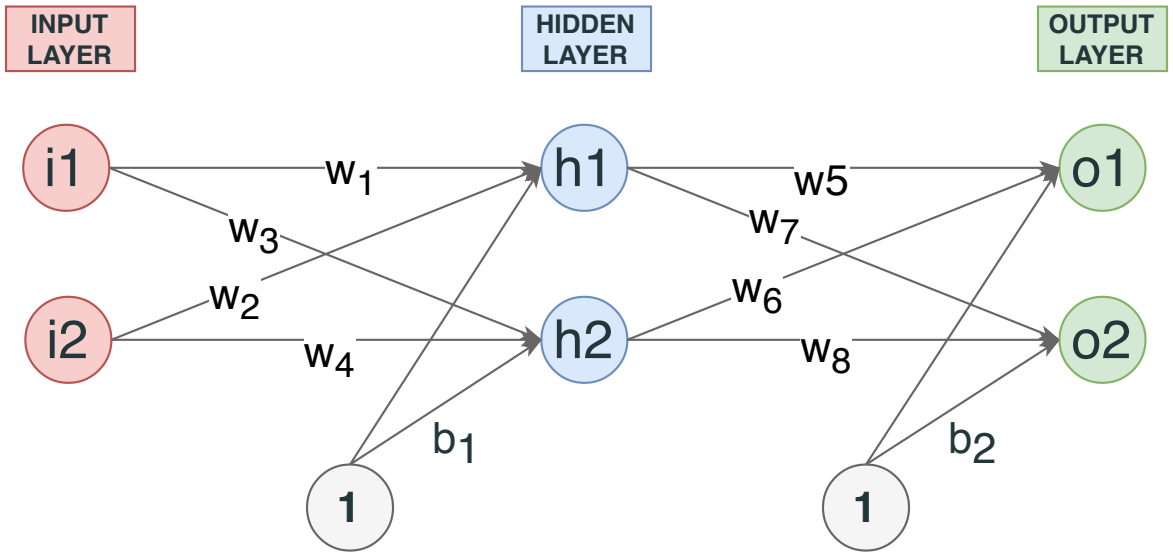
Exemplo de Backpropagation



Forward pass:

- $z_{o1} = w_5 * \sigma_{h1} + w_6 * \sigma_{h_2} + b_2 * 1$
- $\sigma_{o1} = \dfrac{1}{1 + e^{-z_{o1}}}$
- Mesmo cálculo para $o2$

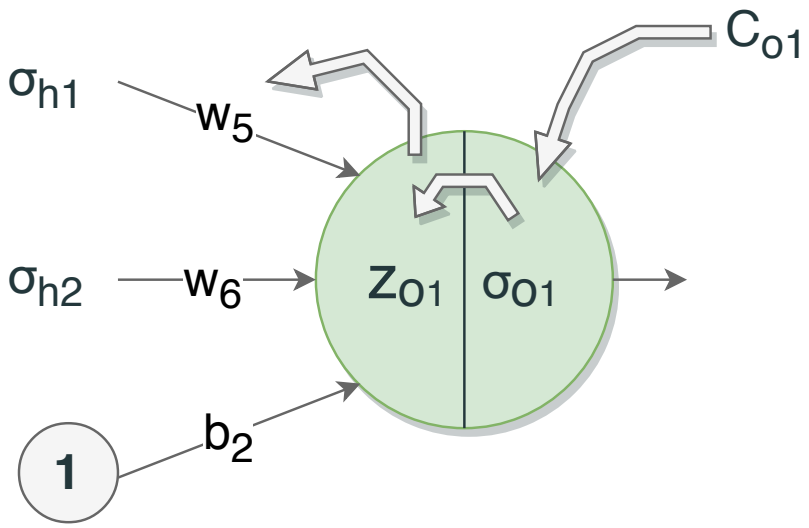
Exemplo de Backpropagation



Backward pass:

- $C = C_{o1} + C_{o2} = \frac{1}{2}(y_{o1} - \sigma_{o_1})^2 + \frac{1}{2}(y_{o2} - \sigma_{o2})^2$
- Quanto cada peso afeta o custo C?

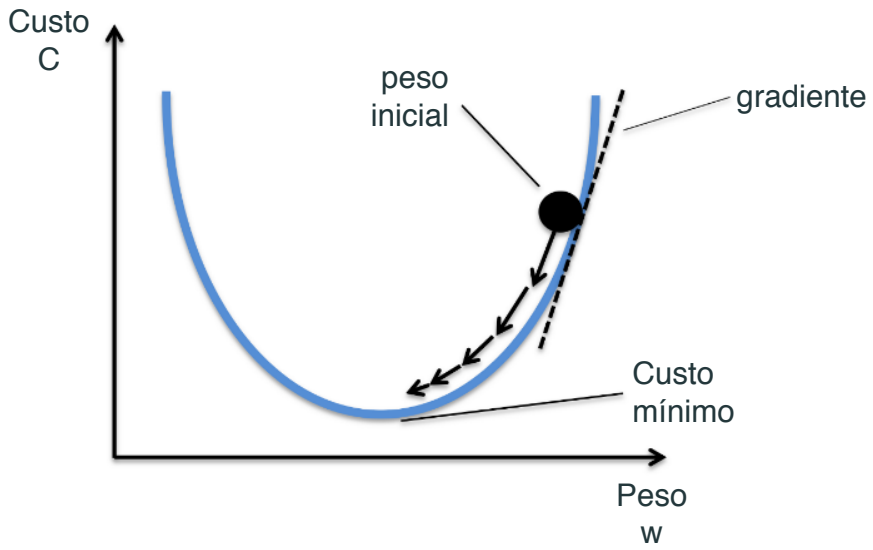
Exemplo de Backpropagation



Backward pass:

- Quanto w_5 afeta o custo C?
- Aplicar regra da cadeia: $\frac{\partial C}{\partial w_5} = \frac{\partial C}{\partial \sigma_{o1}} * \frac{\partial \sigma_{o1}}{\partial z_{o1}} * \frac{\partial z_{o1}}{\partial w_5}$

Exemplo de Backpropagation



Gradient descent:

- Atualizar pesos para minimizar C.
- $w_5 = w_5 - \eta * \frac{\partial C}{\partial w_5}$, onde η é o *learning rate*.
- Mesmo cálculo para os demais pesos.

Prática

Prática

Keras

- API alto-nível para redes neurais, escrita em *Python*.
- Pode rodar em cima do *TensorFlow* ou do *Theano*.
- Possibilidade rápida prototipação, de modo amigável ao usuário.
- Modelos são especificados em alto-nível, comparando-se com o *backend*.
- Também possui uma interface para *R*.

Prática

Exemplos – datasets sintéticos

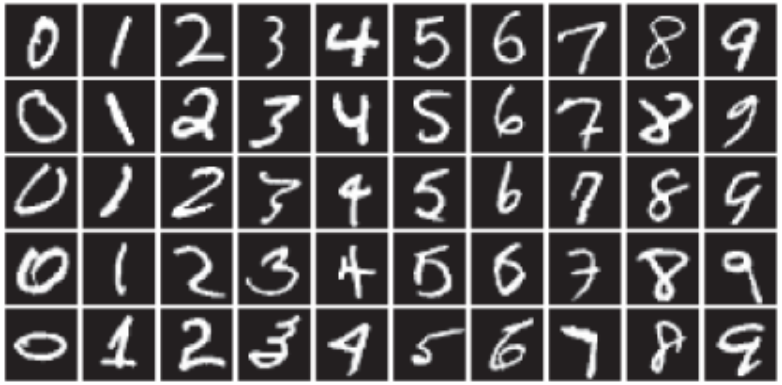
Jupyter notebook:

INF0618_Aula01_Exemplos_ANN_synth.ipynb

Prática

Exemplos – dataset MNIST

Jupyter notebook:
INF0618_Aula01_Exemplos_ANN_MNIST.ipynb



- 70.000 dígitos escritos a mão
- *Labels* de 0 a 9
- Cada imagem tem 28×28 pixels (matriz de 28 por 28 com valores de 0 a 255).
- Problema: treinar uma ANN para classificar uma imagem em um dígito de 0 a 9.