Sensores espectroscópicos e modelos de regressão aplicados na análise de solos

Aula 4 – Redes Neurais Clássicas

Me. José Vinícius Ribeiro

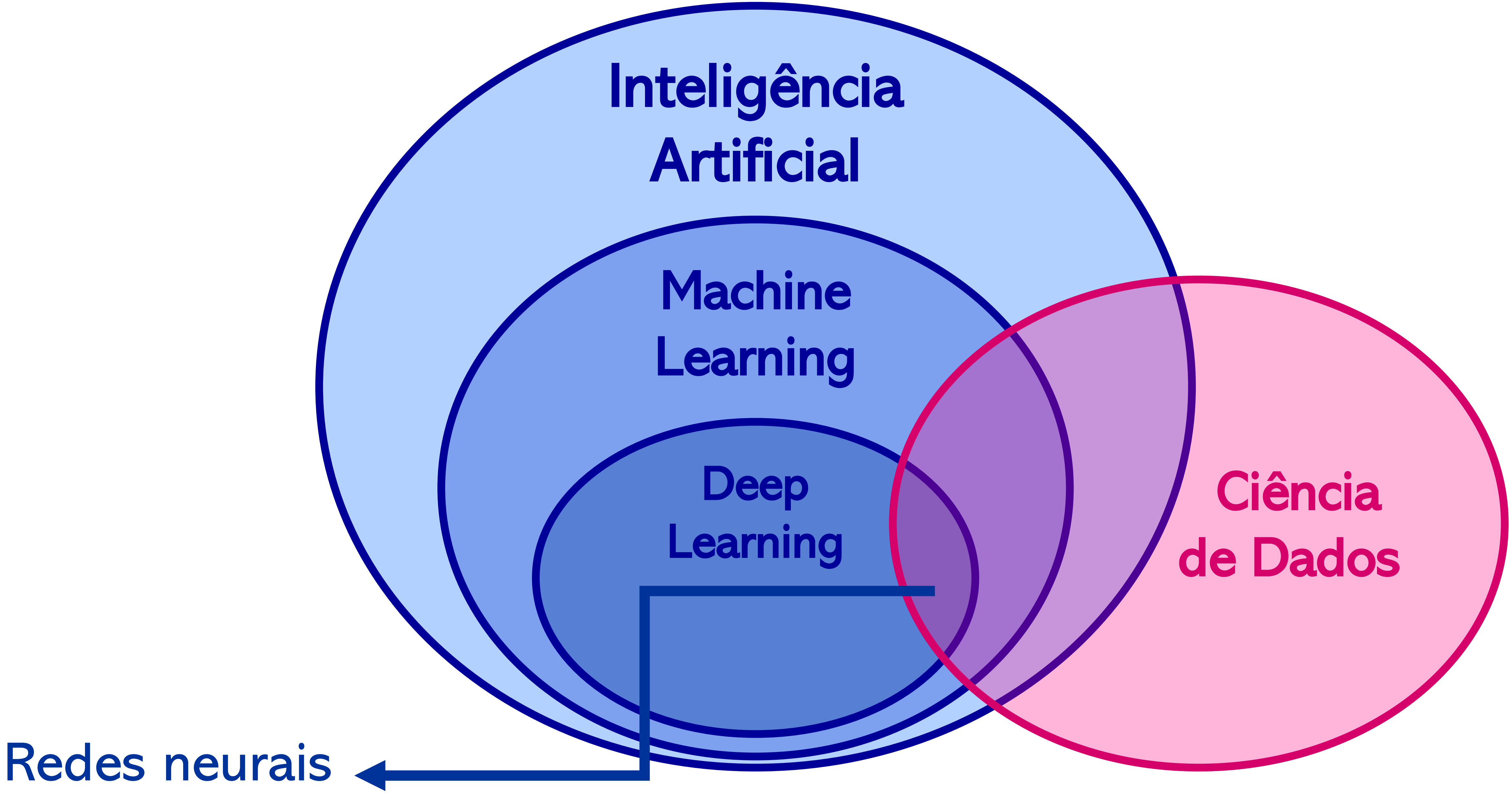
Universidade Estadual de Londrina

PÓS GRADUAÇÃO FÍSICA UEL

SUMÁRIO

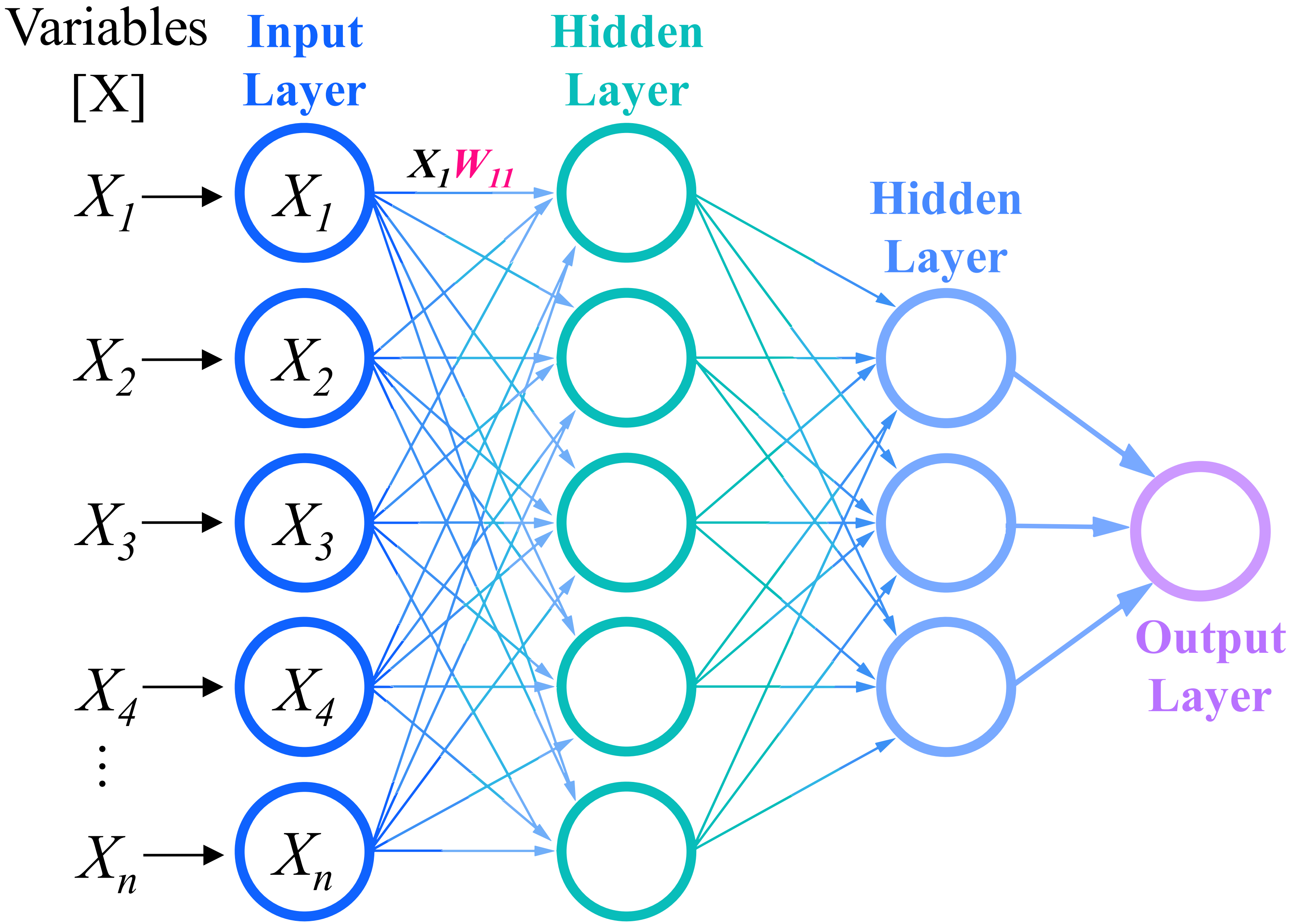
- Neurônio Artificial x Neurônio biológico
- Rede clássica (*Perceptron multi-camadas*)
- *Backpropagation*
- Gradiente descendente (cálculo teórico)
- Redes neurais informadas pela física (PINNs)
- Prática no python (google colab)

Vamos nos situar...



REDES NEURAIS CLÁSSICAS

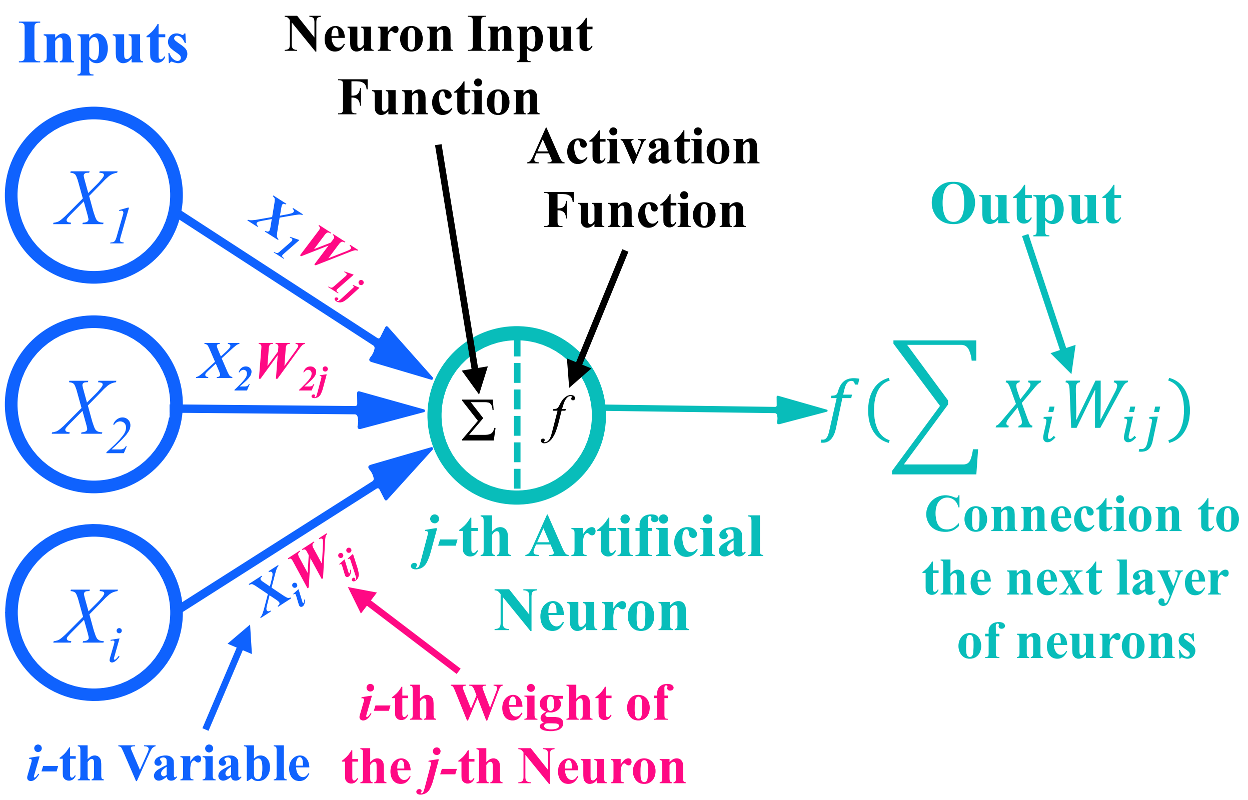
REDE NEURAL – *MULTI LAYER PERCEPTRON*



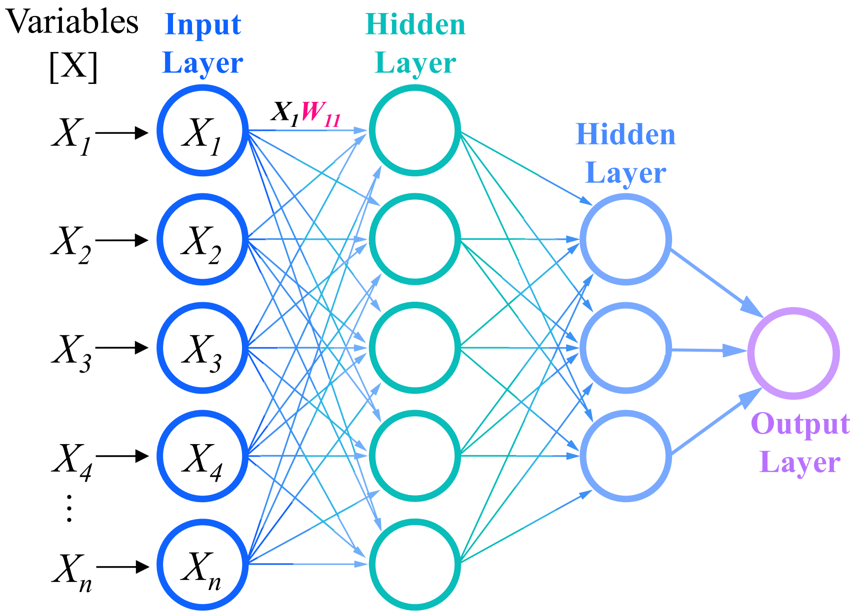
REDE NEURAL – *Perceptron*

Cada neurônio associa um *i*-ésimo peso a *i*-ésima variável

Em seguida, aplica uma função de ativação



Neurônio Artificial (Modelo *Perceptron*)

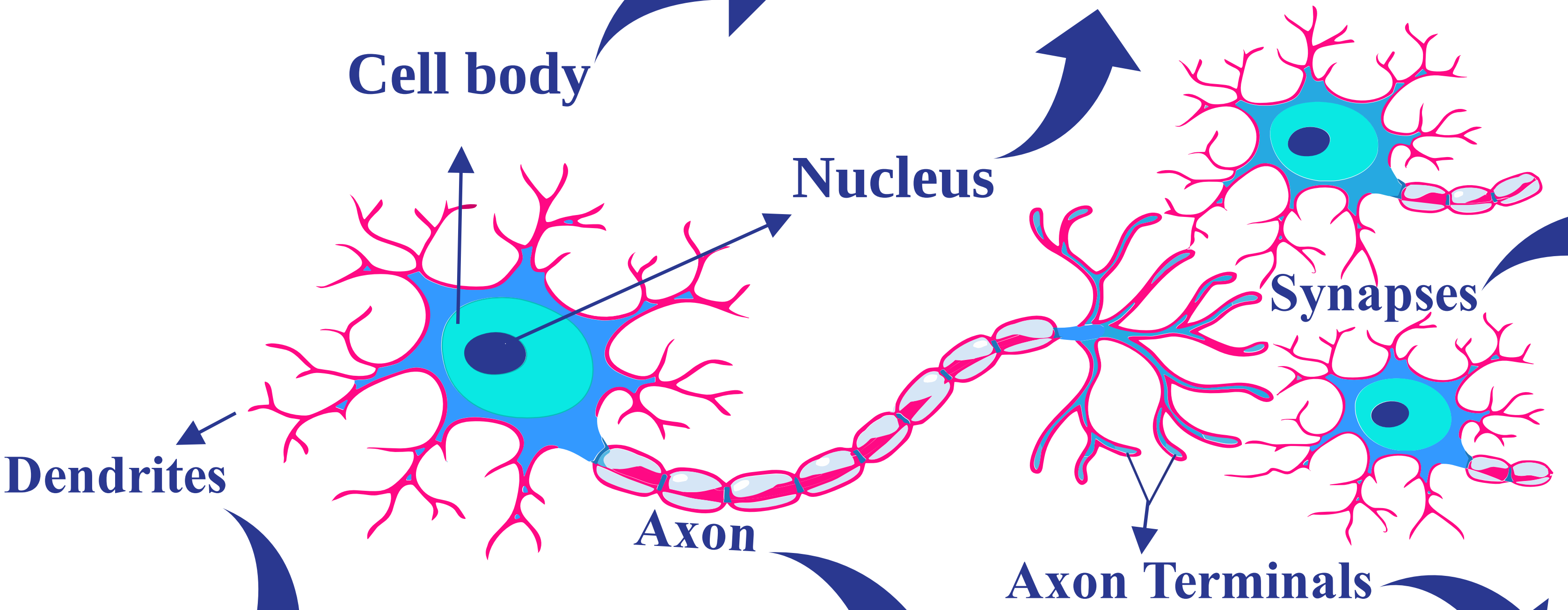


REDE NEURAL – inspiração no cérebro

Unidade básica de comunicação do sistema nervoso dos seres humanos

Neurônio Biológico

Processam os sinais químicos captados



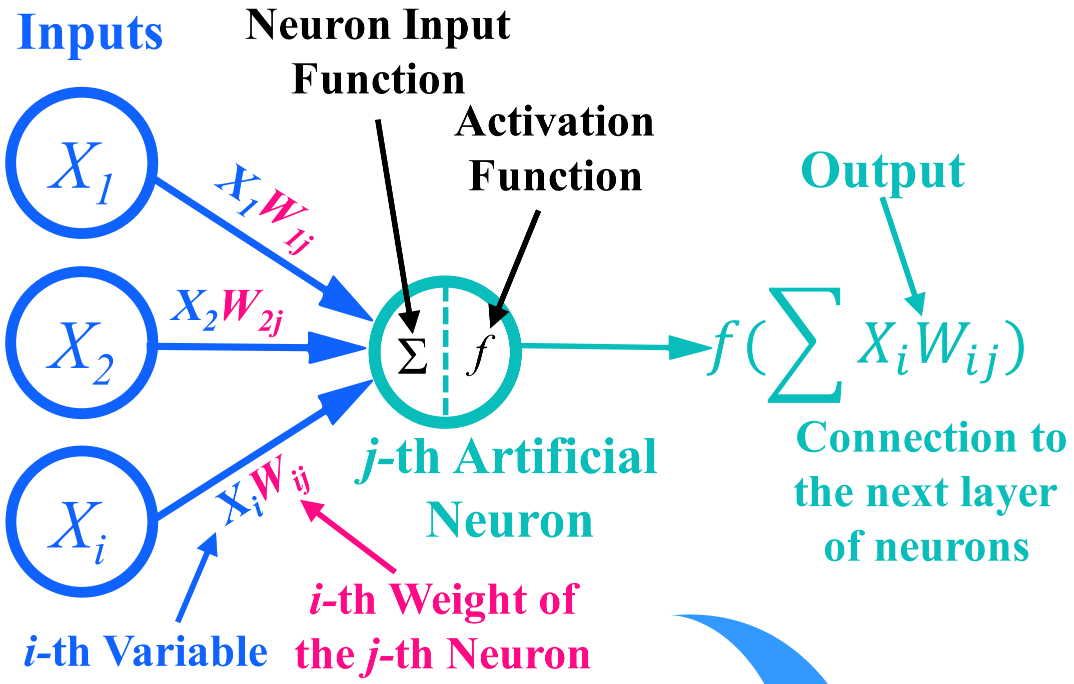
Quando o neurônio acumula sinais suficientes dos receptores do seu dendrito ele dispara, *i.e.*, uma corrente elétrica se propaga pelo axônio até chegar aos terminais, conectando-o com outros neurônios

Recebem os sinais, podendo se conectar com milhares de outros neurônios. São a área de contato disponível para receber informações

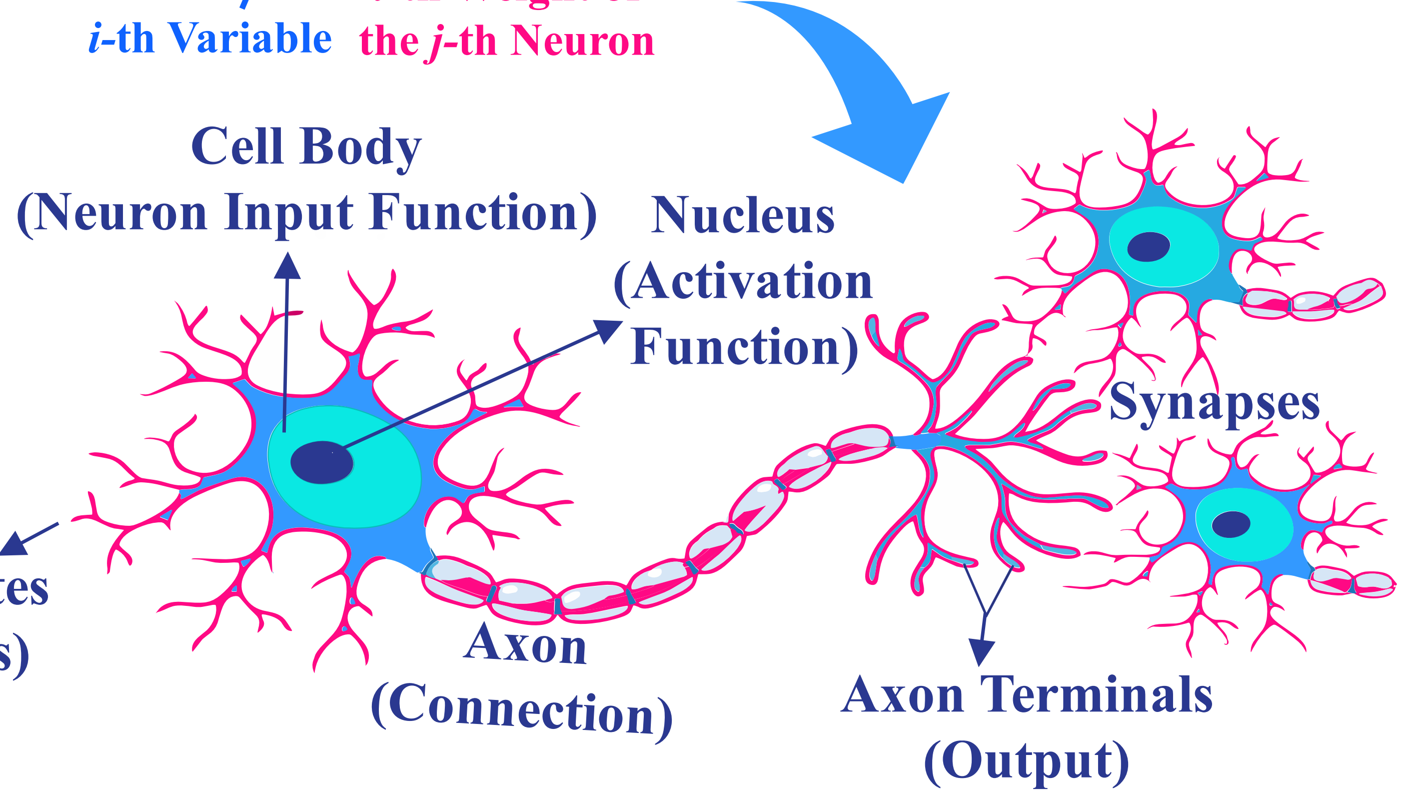
Propaga o sinal processado

Conecta os sinais químicos do neurônio com os outros (podendo gerar mais que conexões que as próprias captadas pelo dendrito)

Neurônio Artificial (Modelo *Perceptron*)



Neurônio Biológico



REDE NEURAL – *MULTI LAYER PERCEPTRON*

Funções de ativação $f(\sum_{i=1}^{n} X_i W_{ij})$

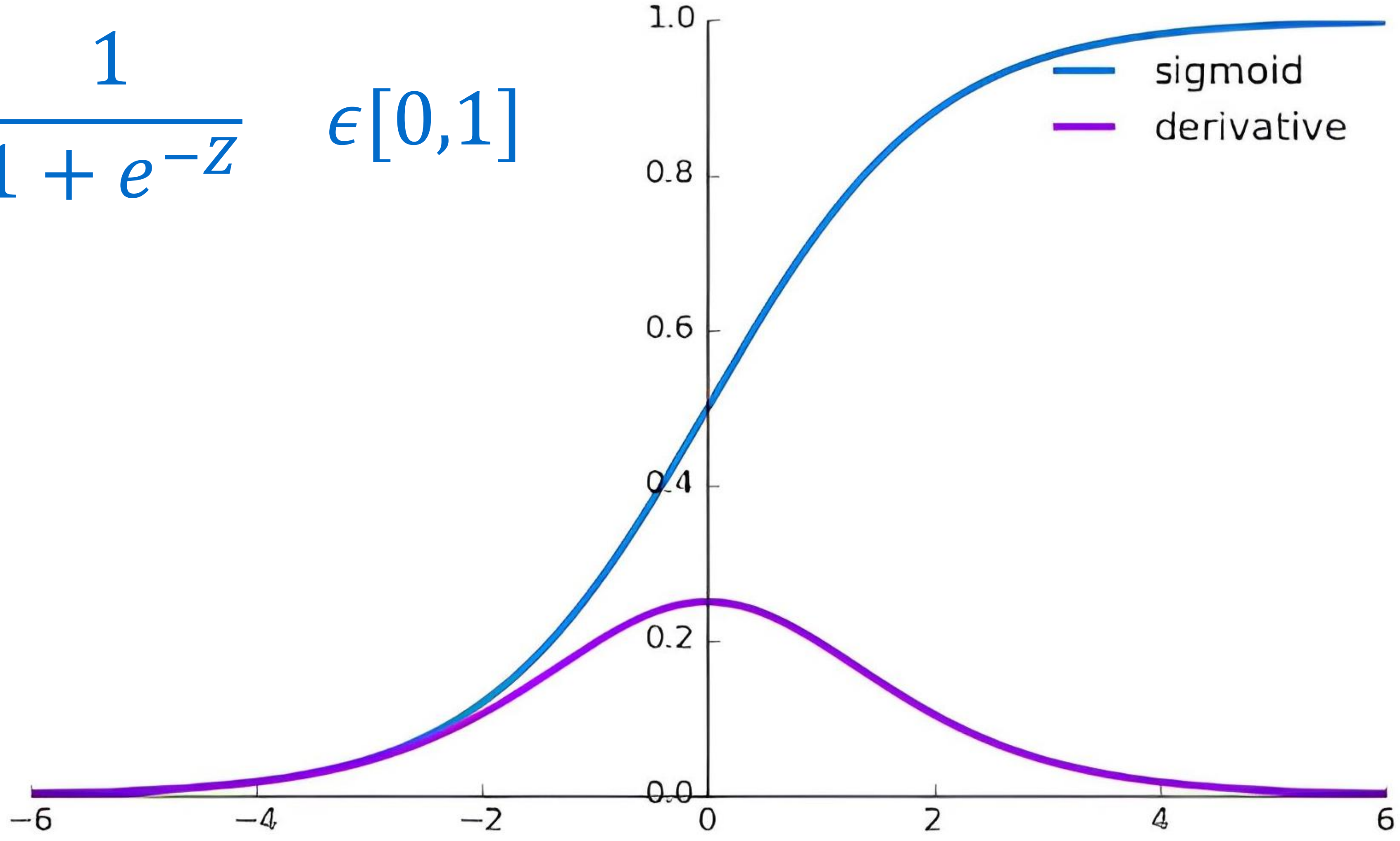
Principais objetivos

- Inserir não-linearidade na relação entre as variáveis e o target
- Facilitar a convergência (objetivo secundário)

Principais tipos: **sigmoid**, **tanh**, **ReLU**, **Softmax**

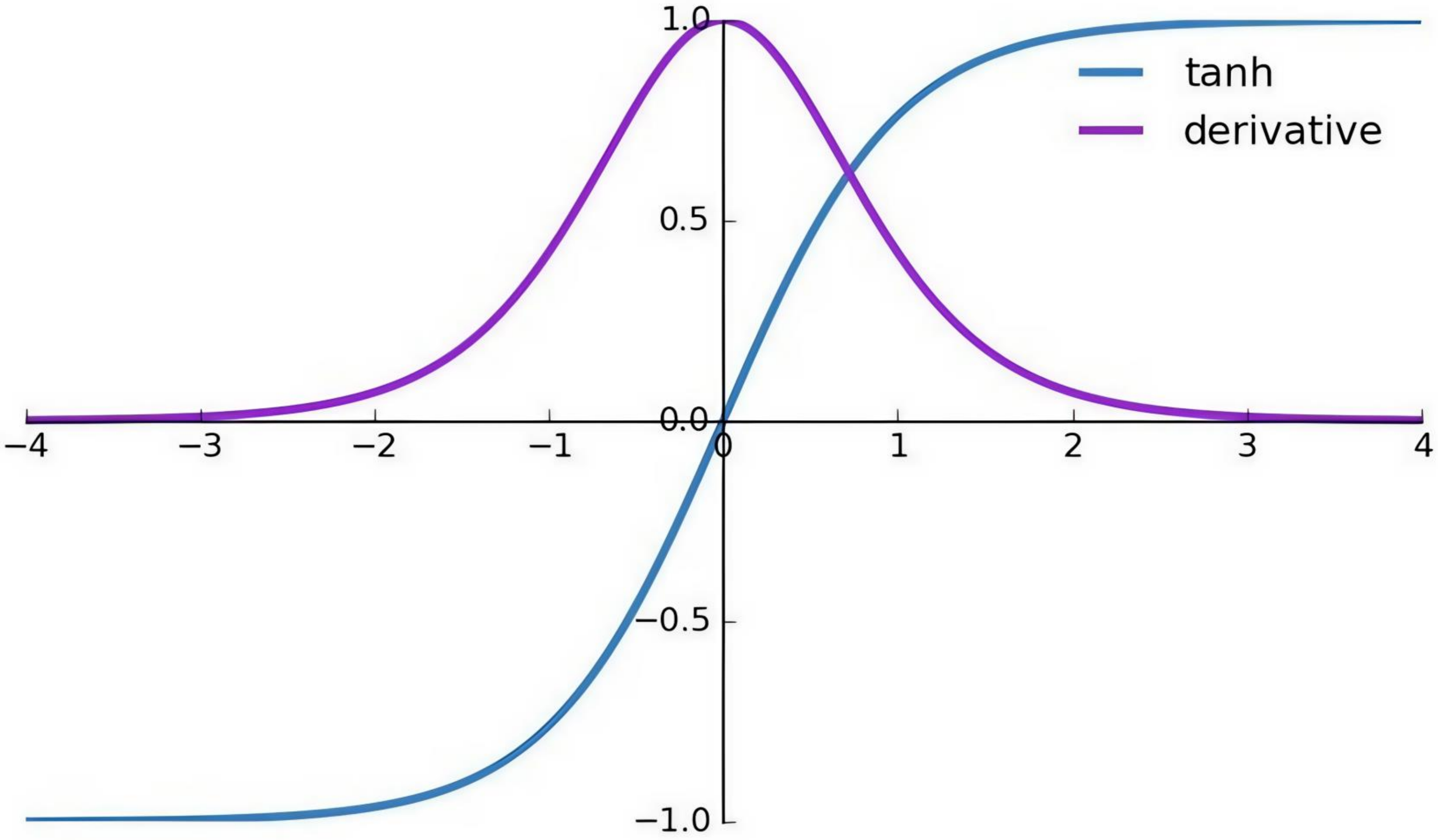
Funções de ativação: Sigmoid

$$f(z) = \frac{1}{1+e^{-z}} \quad \epsilon[0,1]$$

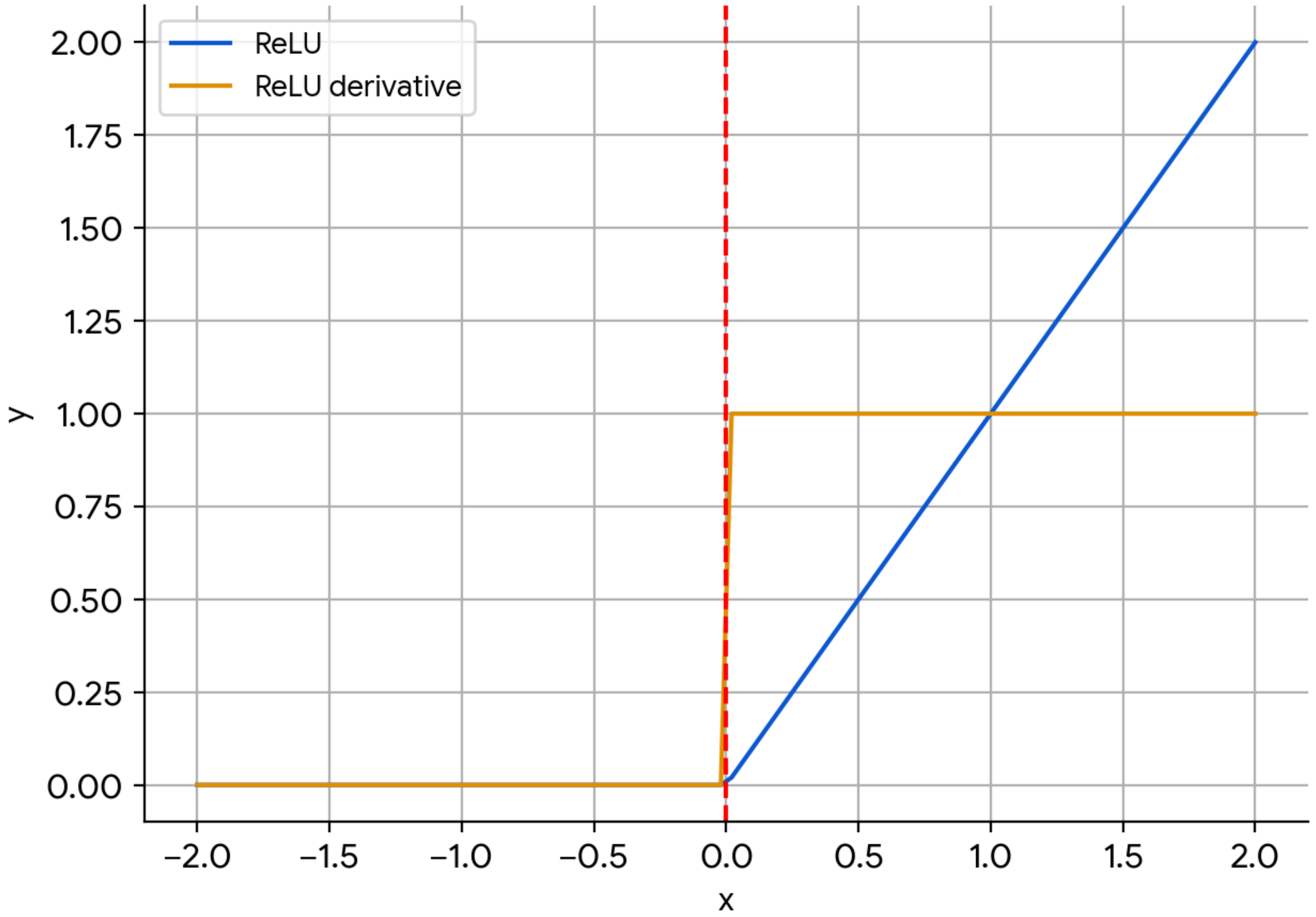


Funções de ativação: tanh

$$f(z) = \frac{e^Z - e^{-Z}}{e^Z + e^{-Z}} \quad \epsilon[-1,1]$$

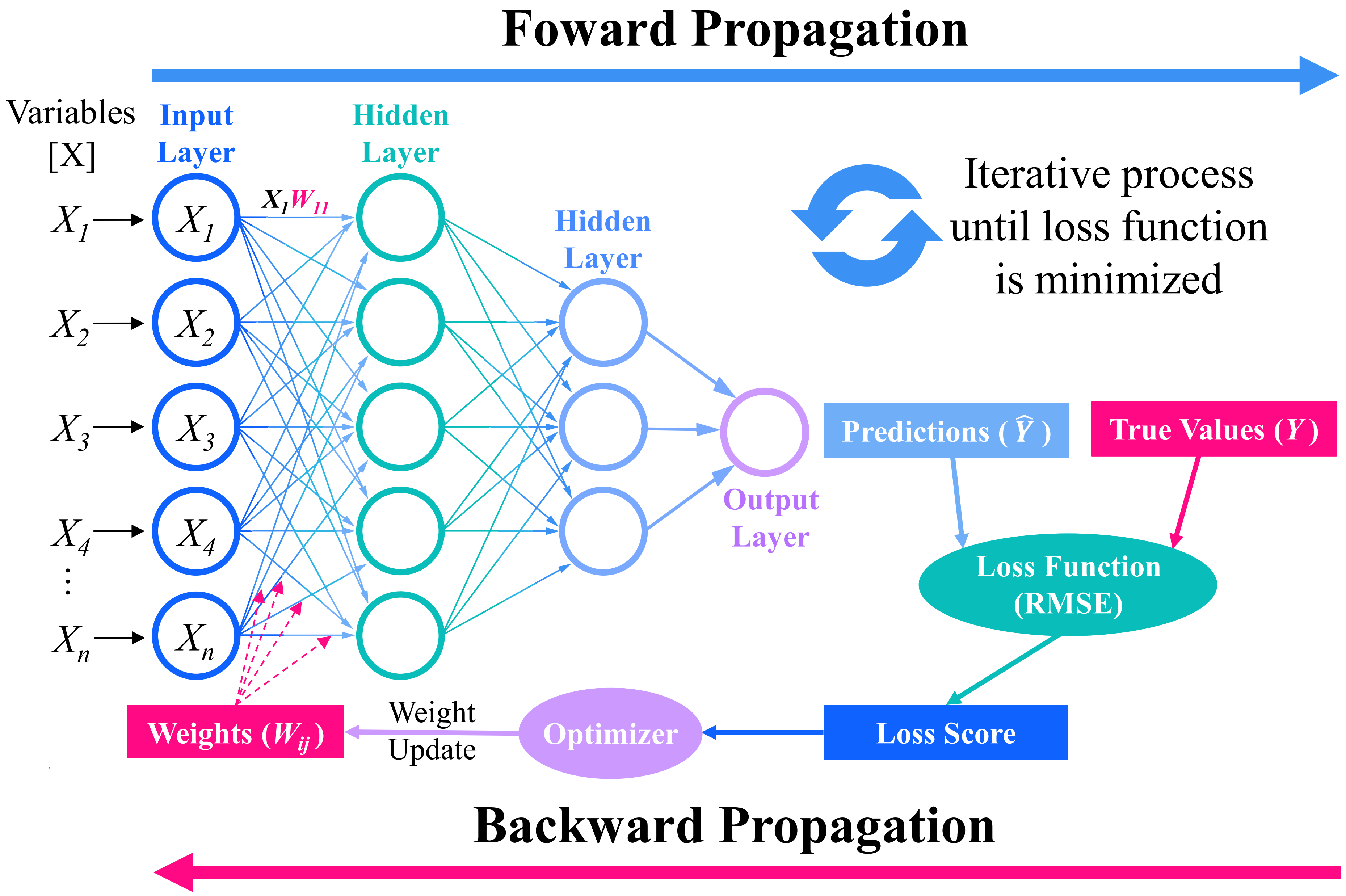


Funções de ativação: Rectified Linear Unit (ReLU)



$ReLU(z) = 0 \; se \; Z < 0 \; Z \; se \; Z > 0 \qquad \epsilon \; [0, \infty]$

REDE NEURAL – *MULTI LAYER PERCEPTRON*



REDE NEURAL – *MULTI LAYER PERCEPTRON*

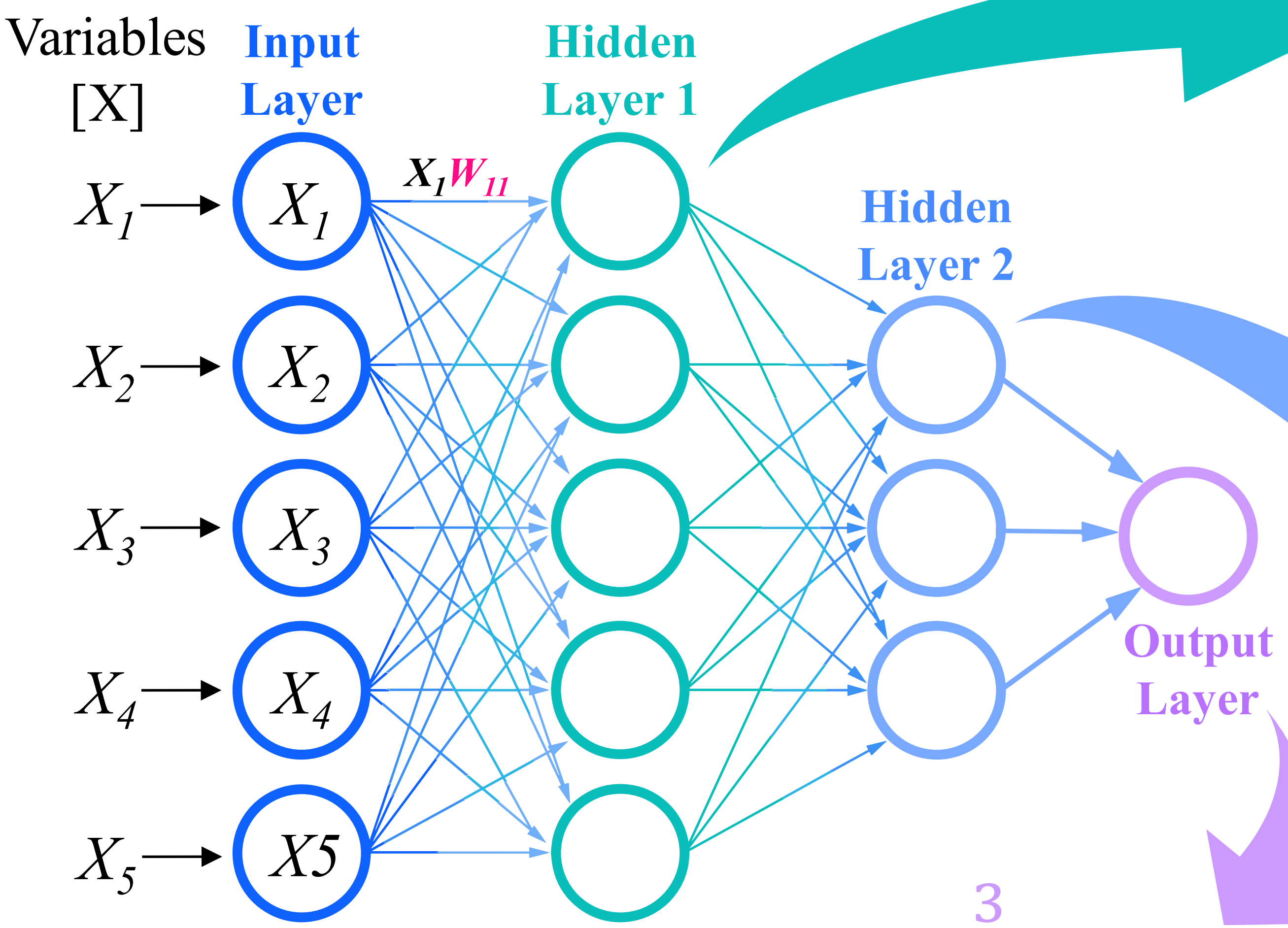
O objetivo da rede é gerar uma função com dependências em diversos parâmetros que, dado uma matriz X de variáveis de entrada, produz um vetor y com valores de saída

FASE FOWARD

A função é então aproximada (através da otimização dos hiper-parâmetros) até que atinja um nível de precisão desejado. Após esse processo ela deve apresentar capacidade de generalização

FASE BACKWARD

FASE FOWARD



$$s_1^{[1]} = w_{11}^{[1]} x_1 + w_{12}^{[1]} x_2 + w_{13}^{[1]} x_3 + w_{14}^{[1]} x_4 + w_{15}^{[1]} x_5$$

$$f_j^{[1]} = f\left(\sum_i^5 w_{ji}^{[1]} x_i\right)$$

$$s_1^{[2]} = w_{11}^{[2]} f_1^{[1]} + w_{12}^{[2]} f_2^{[1]} + w_{13}^{[2]} f_3^{[1]} + w_{14}^{[2]} f_4^{[1]} + w_{15}^{[2]} f_5^{[1]}$$

$$f_k^{[2]} = f\left(\sum_j^5 w_{kj}^{[2]} f_j^{[1]}\right)$$

$$\hat{y} = \sum_{k=1}^{3} w_k^{[3]} s_k^{[2]}$$

Nº pesos

5x5 + 3x5 + 3 = 43

REDE NEURAL – FASE *BACKWARD*

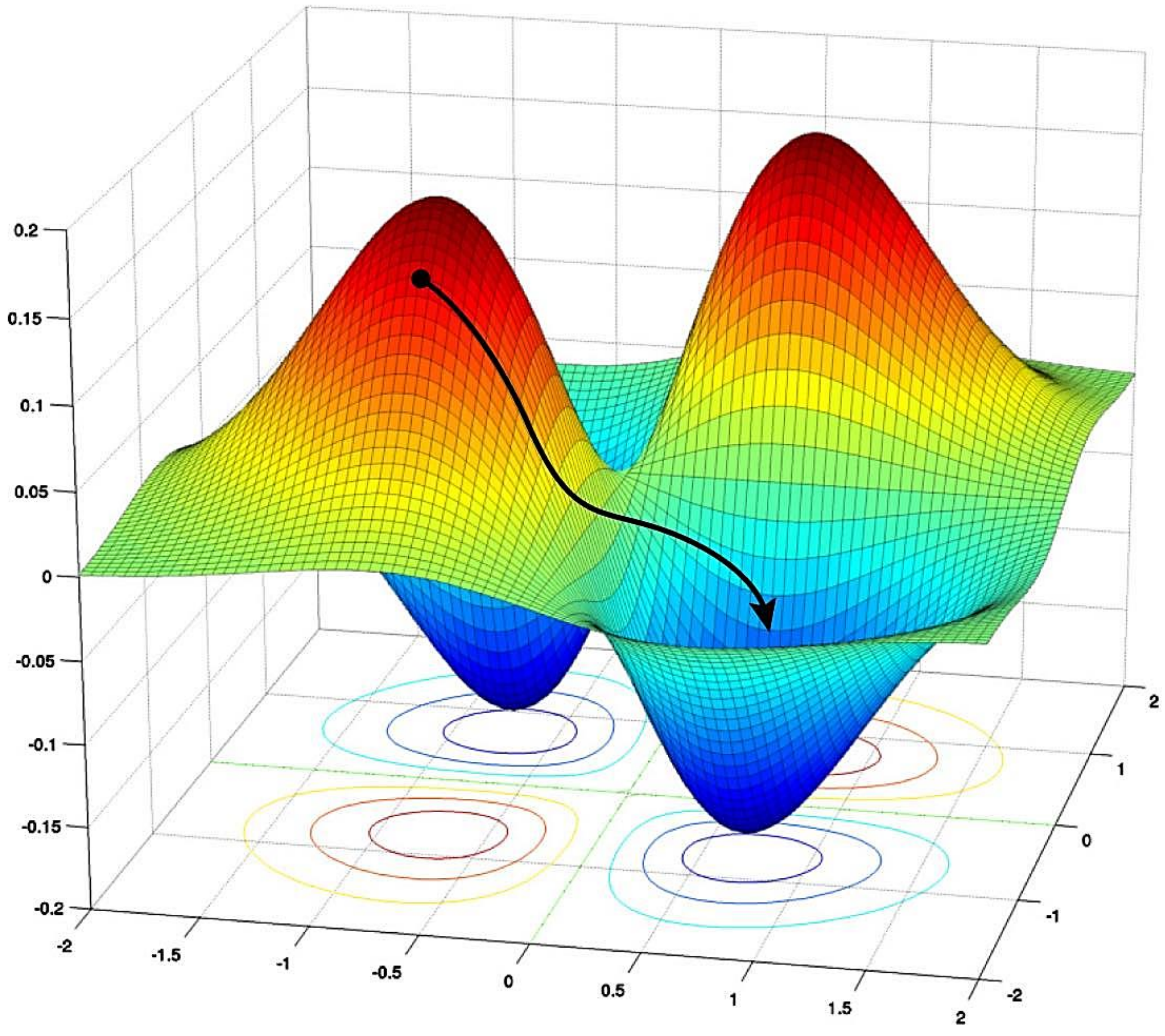
$Loss = y - \hat{y} = L(w)$

Queremos o conjunto de parâmetros w^* que minimiza $L(w)$

Dado um conjunto de valores w^t (dado pelas condições inicias), a uma nova configuração (w^t) na qual $L(w)$ mais irá crescer é dada pelo gradiente aplicado no ponto, *i.e.*, $\nabla L(w^t)$

Logo, em uma <u>**época**</u> futura (t+1): $w^{t+1} = w^t - \nabla L(w^t)$ nos fornecerá uma nova configuração de parâmetros (*i.e.*, novo ponto) que minimizará L

REDE NEURAL – GRADIENTE DESCENDENTE



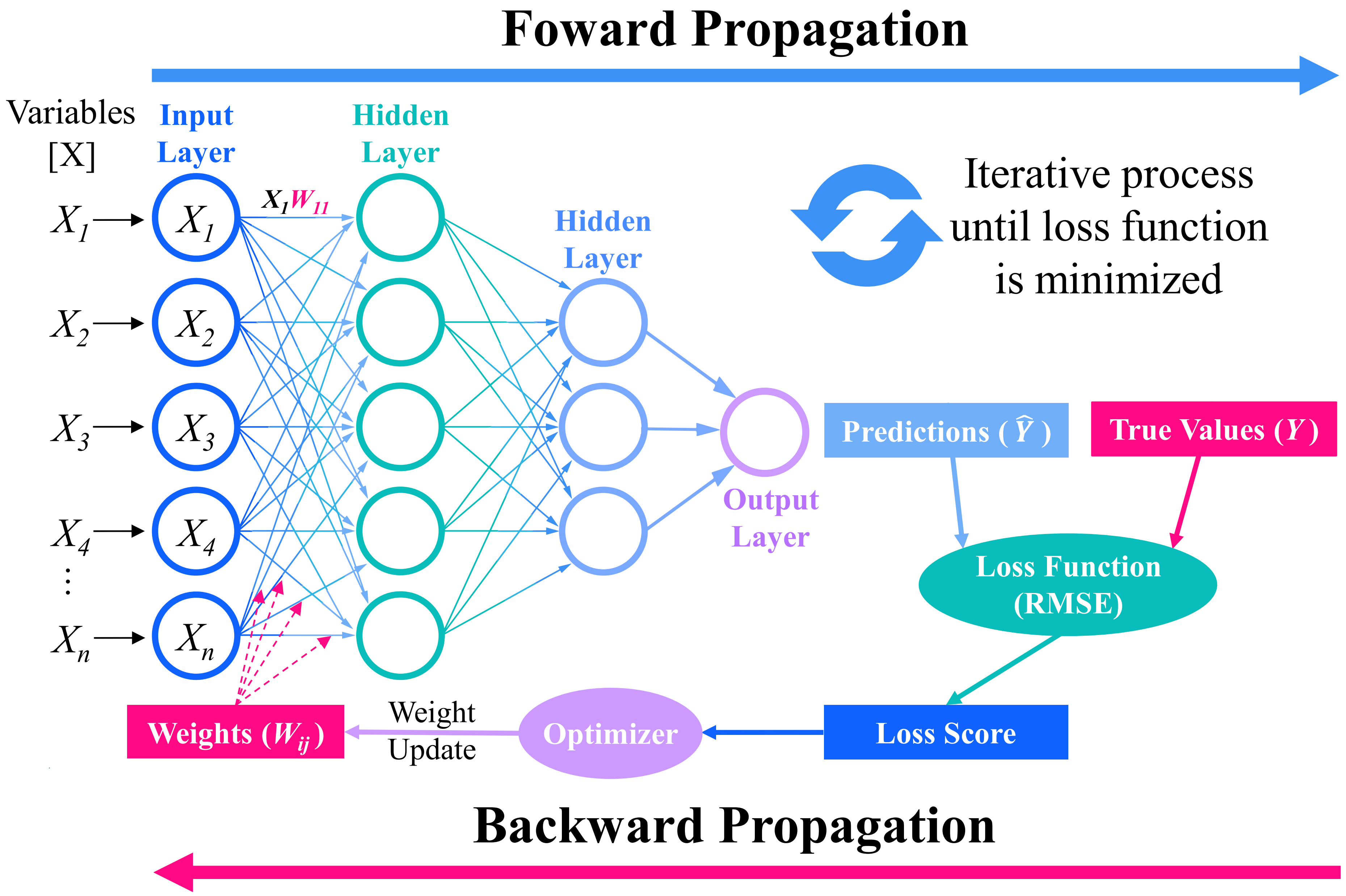
$$w^{t+1} = w^t - \eta_t \nabla L(w^t)$$

η_t : taxa de aprendizagem

GD (*Gradient Descent* - 1950)

Vamos a um exemplo de como é o cálculo

REDE NEURAL – BACKPROPAGATION



GRADIENTE DESCENDENTE

Para qualquer peso $w_{ij}^{[l]}$, queremos saber como o alteramos para que a função de perda $\boldsymbol{L}$ (escalar) fornece a máxima mudança*(-1). Matematicamente isso é o gradiente*(-1), que assume a seguinte forma para funções escalares:

$$(\boldsymbol{\nabla} f)_i = \frac{\partial f}{\partial x_i} \qquad\qquad (\boldsymbol{\nabla}_{\boldsymbol{w}^{[l]}} L)_{ij} = \frac{\partial L}{\partial w_{ij}^{[l]}}$$

Esse valor*(-1) nos diz a direção em que devemos ajustar $w_{ij}^{[l]}$ (positiva ou negativa) e a intensidade (quando calculado no ponto y)

$$\boldsymbol{X} \rightarrow \boldsymbol{w}^{[1]} \rightarrow f(\boldsymbol{w}^{[1]}) \rightarrow \boldsymbol{w}^{[2]} \rightarrow f(\boldsymbol{w}^{[2]}) \rightarrow \ldots \rightarrow \hat{\boldsymbol{y}} \rightarrow L$$

$$\frac{\partial L}{\partial \boldsymbol{w}^{[1]}} = \frac{\partial L}{\partial \hat{\boldsymbol{y}}} \times \frac{\partial \hat{\boldsymbol{y}}}{\partial f(\boldsymbol{w}^{[m]})} \times \cdots \frac{\partial f(\boldsymbol{w}^{[2]})}{\partial \boldsymbol{w}^{[1]}}$$

GRADIENTE DESCENDENTE

$$X \longrightarrow \mathbf{w}^{[1]} \longrightarrow f(\mathbf{w}^{[1]}) \longrightarrow \mathbf{w}^{[2]} \longrightarrow f(\mathbf{w}^{[2]}) \longrightarrow \ldots\ \mathbf{w}^{[l]} \longrightarrow f(\mathbf{w}^{[l]}) \longrightarrow \hat{\mathbf{y}} \longrightarrow L$$

$$\frac{\partial L}{\partial \mathbf{w}^{[1]}} = \frac{\partial L}{\partial \hat{\mathbf{y}}} \times \frac{\partial \hat{\mathbf{y}}}{\partial f(\mathbf{w}^{[m]})} \times \cdots \frac{\partial f(\mathbf{w}^{[2]})}{\partial \mathbf{w}^{[1]}}$$

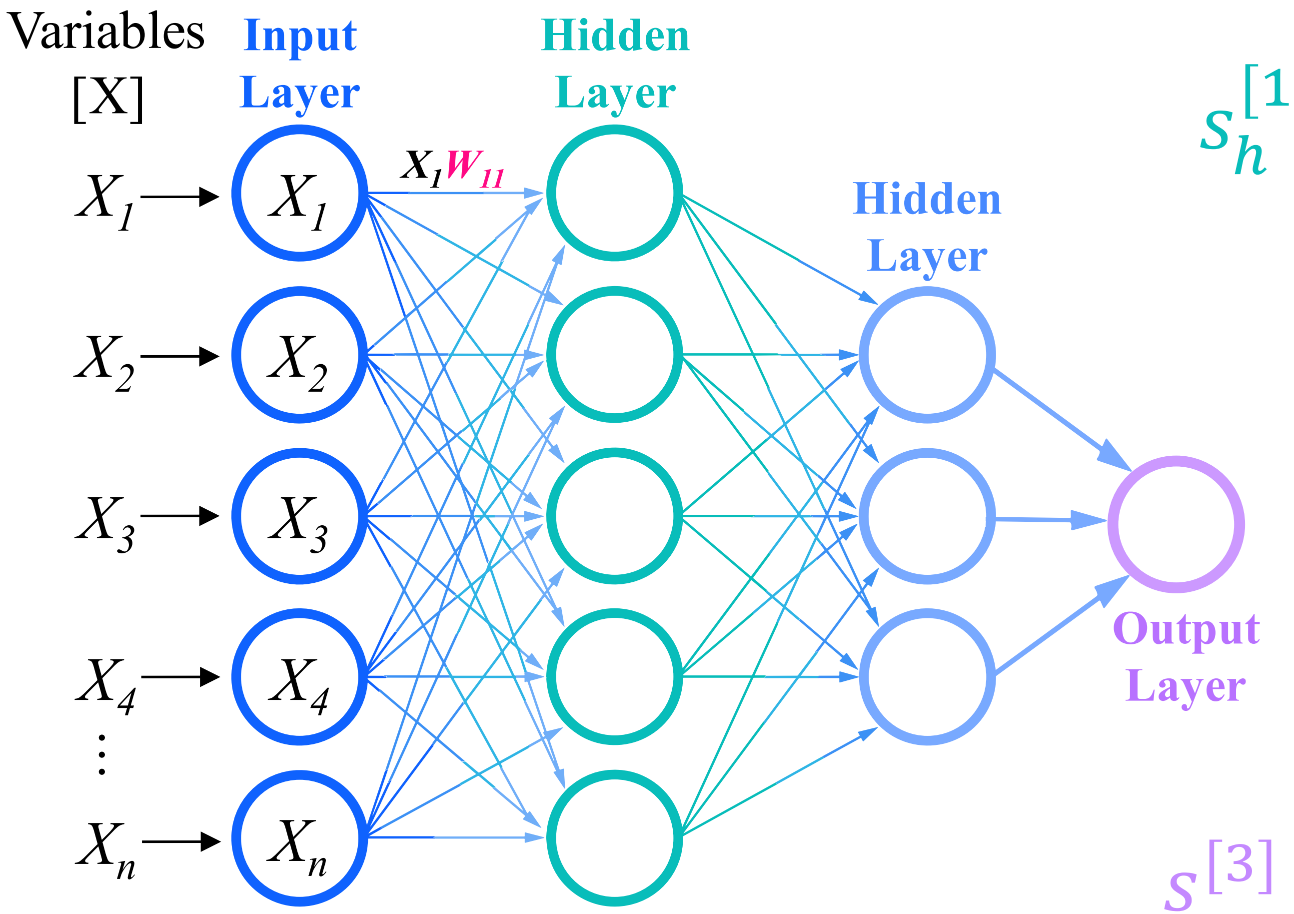
Após calcular as derivadas os pesos são atualizados da ultima até a primeira camada, segundo:

$$w_{hi}^{[1]} \rightarrow w_{hi}^{[1]} - \eta \frac{\partial L}{\partial w_{hi}^{[1]}} \qquad w_{kh}^{[2]} \rightarrow w_{kh}^{[2]} - \eta \frac{\partial L}{\partial w_{kh}^{[2]}} \qquad \cdots \qquad w_{1p}^{[l]} \rightarrow w_{1p}^{[l]} - \eta \frac{\partial L}{\partial w_{1p}^{[l]}}$$

GRADIENTE DESCENDENTE

Exemplo sobre como os pesos são atualizados via *backpropagation*

Para uma única amostra (N=1):



Camada 1 ($h = 1, 2, 3, 4, 5$)

$$s_h^{[1]} = \sum_i^5 w_{hi}^{[1]} x_i \qquad f_h^{[1]} = f(s_h^{[1]})$$

Camada 2 ($k = 1, 2, 3$)

$$s_k^{[2]} = \sum_{h=1}^5 w_{kh}^{[2]} f_h^{[1]} \qquad f_k^{[2]} = f(s_k^{[2]})$$

Output

$$s^{[3]} = \sum_{k=1}^3 w_{1k}^{[3]} f_k^{[2]} \qquad \hat{y} = f^{[3]} = f(s^{[3]})$$

GRADIENTE DESCENDENTE

$$X \longrightarrow w_{hi}^{[1]} \longrightarrow s_h^{[1]} \longrightarrow f_h^{[1]} \longrightarrow w_{kh}^{[2]} \longrightarrow s_k^{[2]} \longrightarrow f_k^{[2]} \longrightarrow w_{1k}^{[3]} \longrightarrow s^{[3]} \longrightarrow f^{[3]} \longrightarrow \hat{y} \longrightarrow L$$

GRADIENTE DESCENDENTE

$$w_{1k}^{[3]} \rightarrow s^{[3]} \rightarrow f^{[3]} \rightarrow \hat{y} \rightarrow L$$

Output

$$s^{[3]} = \sum_{k=1}^{3} w_{1k}^{[3]} f_k^{[2]} \qquad \hat{y} = f^{[3]} = f(s^{[3]})$$

O quanto L muda se $s^{[3]}$ muda?

$$\frac{\partial L}{\partial s^{[3]}} = \frac{\partial L}{\partial \hat{y}} \frac{d\hat{y}}{ds^{[3]}} = \varepsilon^{[3]}$$

$$\varepsilon^{[3]} = 2(\hat{y} - y)\frac{d\hat{y}}{ds^{[3]}}$$

Função de perda MSE (N=1)

$$L = \frac{\sum_{n=1}^{N} (\hat{y}_n - y_n)^2}{N}$$

$$L = (\hat{y}_1 - y_1)^2$$

$$\hat{y}_1 = \hat{y} = \hat{y}(f(s^{[3]}))$$

GRADIENTE DESCENDENTE

$$w_{1k}^{[3]} \rightarrow s^{[3]} \rightarrow f^{[3]} \rightarrow \hat{y} \rightarrow L$$

Output $$s^{[3]} = \sum_{k=1}^{3} w_{1k}^{[3]} f_k^{[2]} \qquad \hat{y} = f^{[3]} = f(s^{[3]})$$

O quanto L muda se $s^{[3]}$ muda?

$$\frac{\partial L}{\partial s^{[3]}} = \varepsilon^{[3]} = 2(\hat{y} - y)\frac{d\hat{y}}{ds^{[3]}}$$

O quanto $s^{[3]}$ muda se $w_{1k}^{[3]}$?

$$\frac{\partial s^{[3]}}{\partial w_{1k}^{[3]}} = \sum_{k=1}^{3} \delta_{k,l} f_k^{[2]} = f_k^{[2]}$$

Logo $$\frac{\partial L}{\partial w_{1k}^{[3]}} = \frac{\partial L}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial s^{[3]}} \frac{\partial s^{[3]}}{\partial w_{1k}^{[3]}} = \varepsilon^{[3]} f_k^{[2]}$$

GRADIENTE DESCENDENTE

$$w_{kh}^{[2]} \rightarrow s_k^{[2]} \rightarrow f_k^{[2]} \rightarrow w_{1k}^{[3]} \rightarrow s^{[3]} \rightarrow f^{[3]} \rightarrow \hat{y} \rightarrow L$$

Na camada dois, queremos saber como L varia quando os pesos $w_{kh}^{[2]}$ variam

$$\frac{\partial L}{\partial s^{[3]}} = \varepsilon^{[3]} \qquad s^{[3]} = \sum_{k=1}^{3} w_{1k}^{[3]} f_k^{[2]} \qquad \frac{\partial s^{[3]}}{\partial f_k^{[2]}} = \sum_{k=1}^{3} w_{1k}^{[3]} \delta_{k,l} = w_{1k}^{[3]}$$

$$\frac{\partial L}{\partial s_k^{[2]}} = \varepsilon_k^{[2]} = \frac{\partial L}{\partial s^{[3]}} \frac{\partial s^{[3]}}{\partial f_k^{[2]}} \frac{\partial f_k^{[2]}}{\partial s_k^{[2]}} = \varepsilon^{[3]} w_{1k}^{[3]} \frac{\partial f_k^{[2]}}{\partial s_k^{[2]}}$$

GRADIENTE DESCENDENTE

$$w_{kh}^{[2]} \rightarrow s_k^{[2]} \rightarrow f_k^{[2]} \rightarrow w_{1k}^{[3]} \rightarrow s^{[3]} \rightarrow f^{[3]} \rightarrow \hat{y} \rightarrow L$$

Na camada dois, queremos saber como L varia quando os pesos $w_{kh}^{[2]}$ variam

$$\frac{\partial L}{\partial s_k^{[2]}} = \varepsilon_k^{[2]} \qquad \frac{\partial}{\partial w_{kh}^{[2]}} s_k^{[2]} = \sum_{h=1}^{5} \delta_{k,p}\delta_{h,q}\, f_h^{[1]} = f_h^{[1]}$$

$$\frac{\partial L}{\partial w_{kh}^{[2]}} = \frac{\partial L}{\partial s_k^{[2]}} \frac{\partial s_k^{[2]}}{\partial w_{kh}^{[2]}} = \varepsilon_k^{[2]} f_h^{[1]}$$

GRADIENTE DESCENDENTE

$$X \rightarrow w_{hi}^{[1]} \rightarrow s_h^{[1]} \rightarrow f_h^{[1]} \rightarrow w_{kh}^{[2]} \rightarrow s_k^{[2]} \rightarrow f_k^{[2]} \rightarrow w_{1k}^{[3]} \rightarrow s^{[3]} \rightarrow f^{[3]} \rightarrow \hat{y} \rightarrow L$$

Na camada um, queremos saber como L varia quando os pesos $w_{hi}^{[1]}$ variam

$$\frac{\partial L}{\partial s_k^{[2]}} = \varepsilon_k^{[2]} \qquad s_k^{[2]} = \sum_h^5 w_{kh}^{[2]} f_h^{[1]} \qquad \frac{\partial}{\partial f_h^{[1]}} s_k^{[2]} = \sum_{h=1}^{5} \delta_{h,q}\, w_{kh}^{[2]} = w_{kh}^{[2]}$$

$$\frac{\partial L}{\partial s_h^{[1]}} = \varepsilon_h^{[1]} = \sum_{k=1}^{3} \frac{\partial L}{\partial s_k^{[2]}} \frac{\partial s_k^{[2]}}{\partial f_h^{[1]}} \frac{\partial f_h^{[1]}}{\partial s_h^{[1]}} = \varepsilon_k^{[2]} \sum_{k=1}^{3} w_{kh}^{[2]} \frac{\partial f_h^{[1]}}{\partial s_h^{[1]}}$$

GRADIENTE DESCENDENTE

$$X \rightarrow w_{hi}^{[1]} \rightarrow s_h^{[1]} \rightarrow f_h^{[1]} \rightarrow w_{kh}^{[2]} \rightarrow s_k^{[2]} \rightarrow f_k^{[2]} \rightarrow w_{1k}^{[3]} \rightarrow s^{[3]} \rightarrow f^{[3]} \rightarrow \hat{y} \rightarrow L$$

Na camada um, queremos saber como L varia quando os pesos $w_{hi}^{[1]}$ variam

$$\frac{\partial L}{\partial s_h^{[1]}} = \varepsilon_h^{[1]} \qquad s_h^{[1]} = \sum_{i}^{5} w_{hi}^{[1]} x_i$$

$$\frac{\partial L}{\partial w_{hi}^{[1]}} = \frac{\partial L}{\partial s_h^{[1]}} \frac{\partial s_h^{[1]}}{\partial w_{hi}^{[1]}} = \varepsilon_h^{[1]} x_i$$

GRADIENTE DESCENDENTE

$$X \rightarrow w_{hi}^{[1]} \rightarrow s_h^{[1]} \rightarrow f_h^{[1]} \rightarrow w_{kh}^{[2]} \rightarrow s_k^{[2]} \rightarrow f_k^{[2]} \rightarrow w_{1k}^{[3]} \rightarrow s^{[3]} \rightarrow f^{[3]} \rightarrow \hat{y} \rightarrow L$$

Atualizando os pesos

$$w_{1k}^{[3]} \rightarrow w_{1k}^{[3]} - \eta \frac{\partial L}{\partial w_{1k}^{[3]}} = w_{1k}^{[3]} - \eta \varepsilon^{[3]} f_k^{[2]}$$

$$\varepsilon^{[3]} = \frac{\partial L}{\partial s^{[3]}} = \frac{\partial L}{\partial \hat{y}} \frac{d\hat{y}}{ds^{[3]}}$$

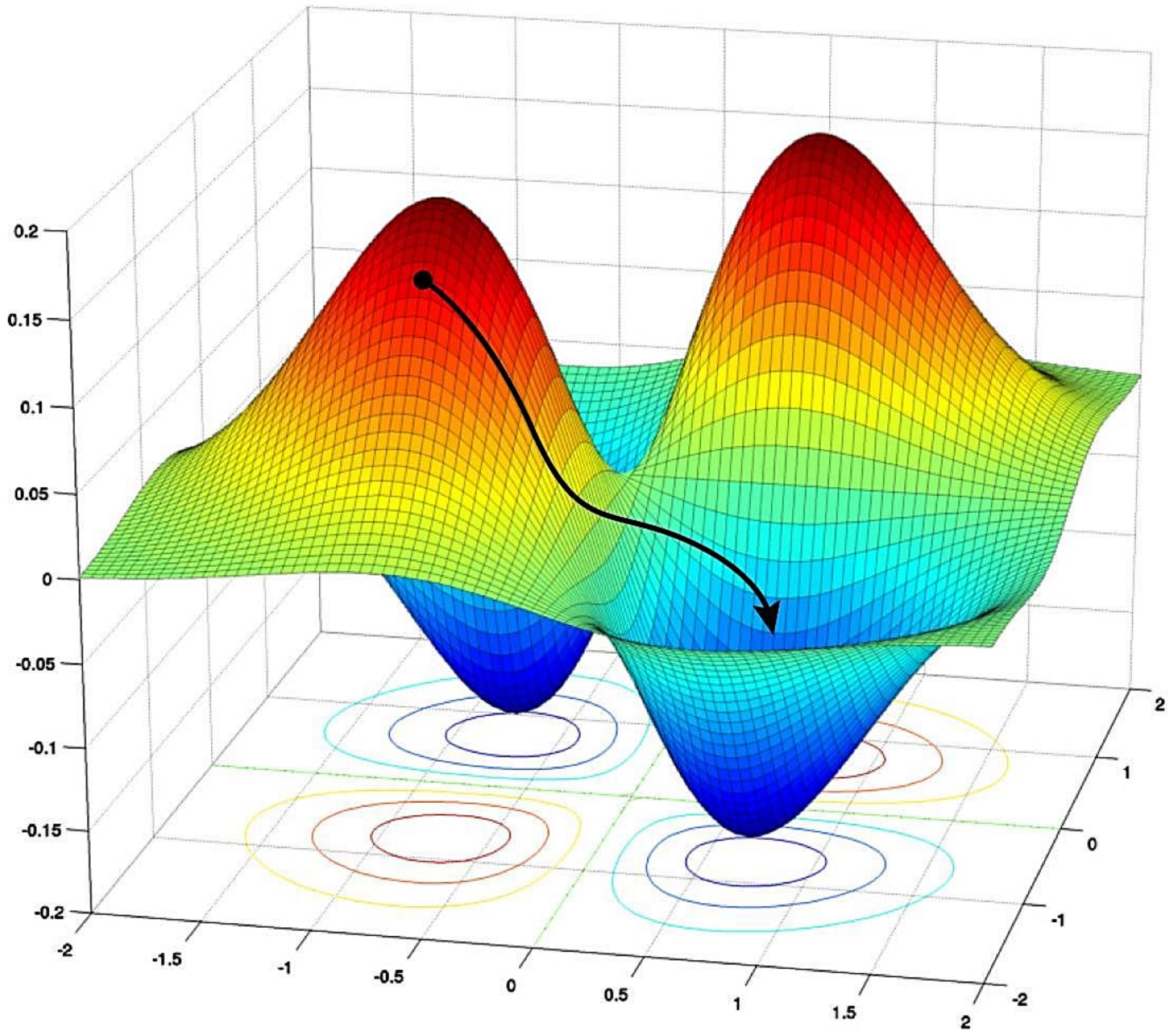
$$w_{kh}^{[2]} \rightarrow w_{kh}^{[2]} - \eta \frac{\partial L}{\partial w_{kh}^{[2]}} = w_{kh}^{[2]} - \eta \varepsilon_k^{[2]} f_h^{[1]}$$

$$\varepsilon_k^{[2]} = \varepsilon^{[3]} w_{1k}^{[3]} \frac{\partial f_k^{[2]}}{\partial s_k^{[2]}}$$

$$w_{hi}^{[1]} \rightarrow w_{hi}^{[1]} - \eta \frac{\partial L}{\partial w_{hi}^{[1]}} = w_{hi}^{[1]} - \eta \varepsilon_h^{[1]} x_i$$

$$\varepsilon_h^{[1]} = \sum_{k=1}^{3} \varepsilon_k^{[2]} w_{kh}^{[2]} \frac{\partial f_h^{[1]}}{\partial s_h^{[1]}}$$

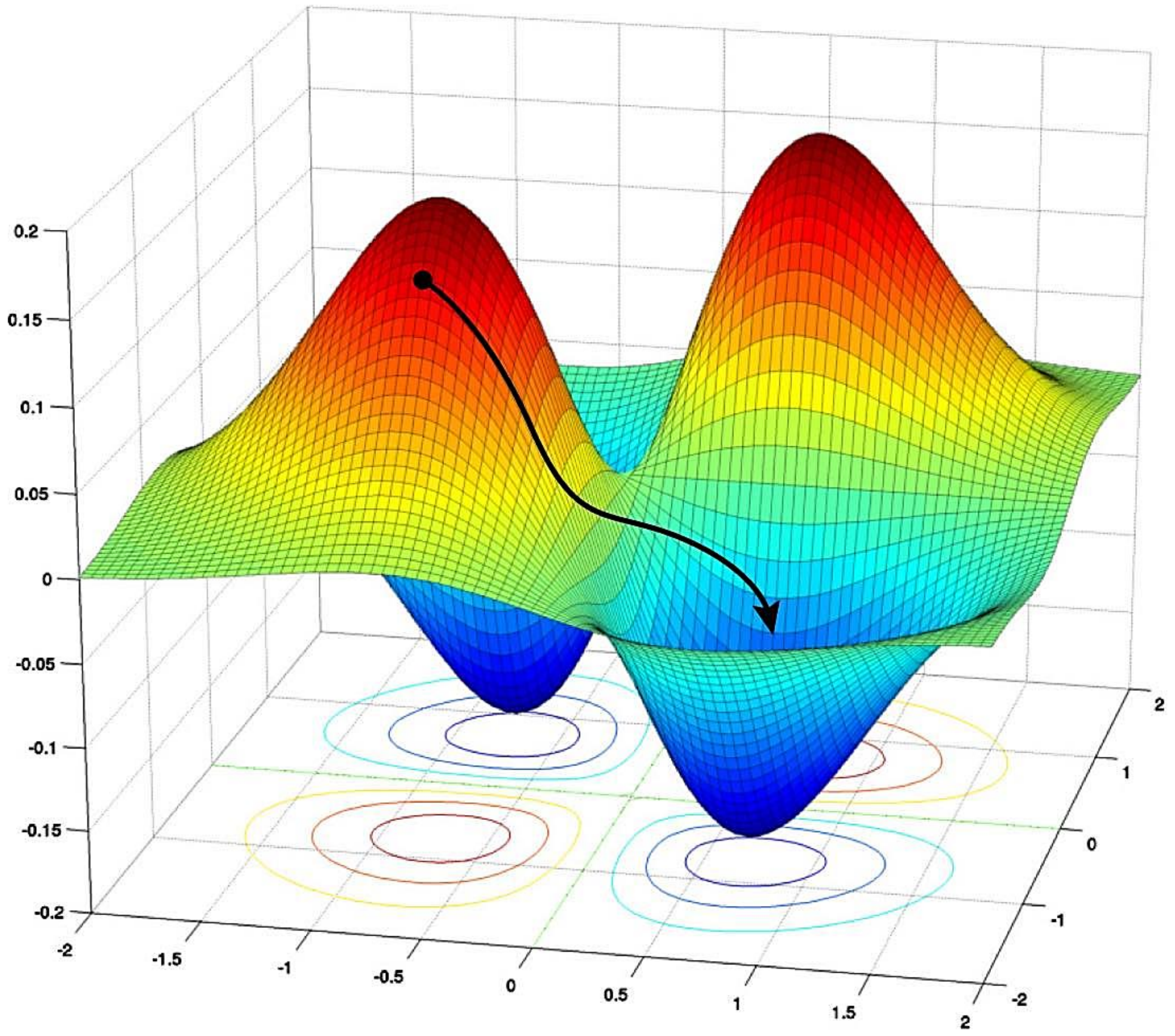
Backpropagation



$$w \rightarrow w - \eta \frac{\partial L}{\partial w}$$

Na prática o gradiente descendente tem sido cada vez menos utilizado devido a novas propostas. Novas abordagens fazem o cálculo computacionalmente, por meio de algoritmos otimizados para aproximar as derivadas

Backpropagation



$$w \rightarrow w - \eta \frac{\partial L}{\partial w}$$

L-BFGS (*Limited-Memory* Broyden–Fletcher –Goldfarb–Shannon - 1989)

ADAM (*Adaptative Moment Estimation* - 2014)

REDE NEURAL – *MULTI LAYER PERCEPTRON*

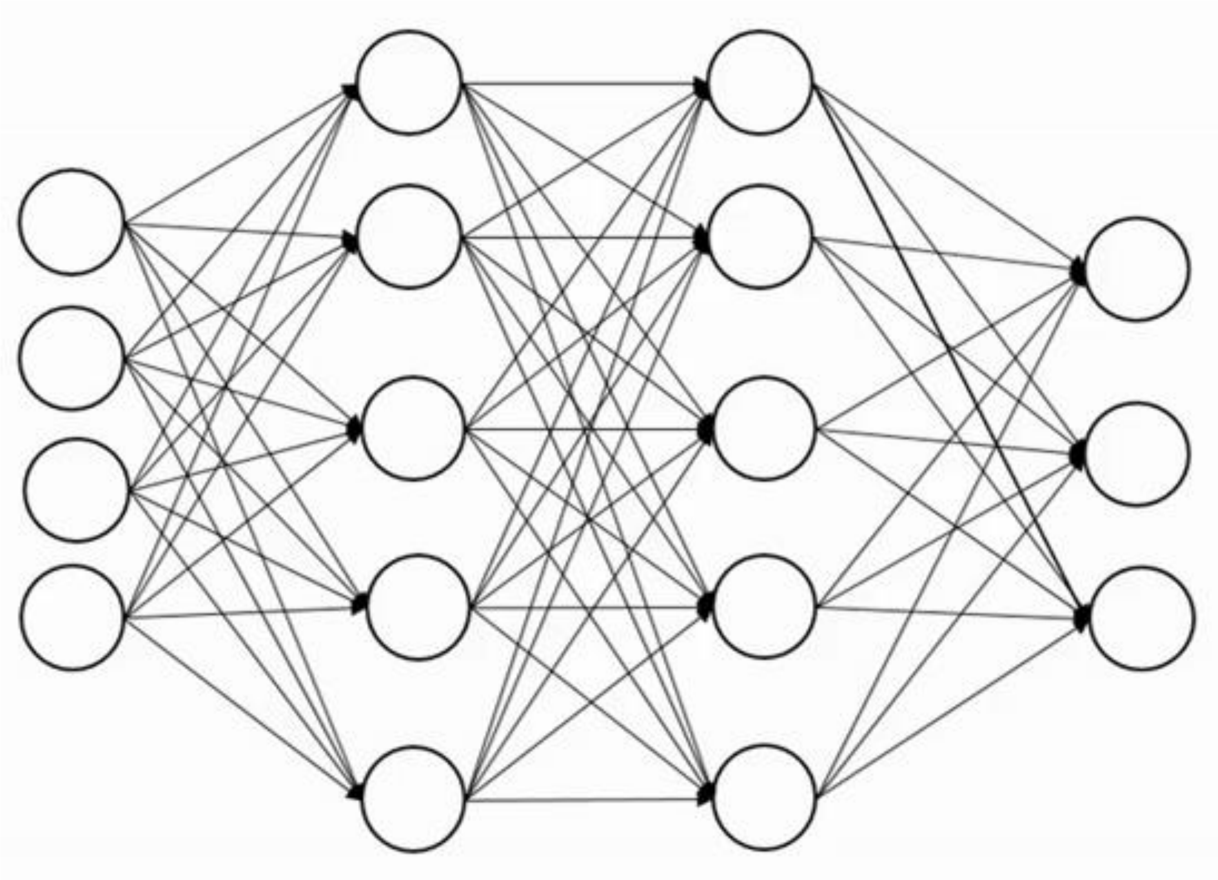
É importante discernir que há dois processos de otimização a serem seguidos.

- O primeiro em relação a arquitetura da rede: número de camadas, neurônios, função de perda, etc. Esse é feito a priori de acordo com a experiência, via grades de pesquisa ou via algum outro algoritmo heurístico (ex: GA).

- O segundo é o ajuste dos hiperparâmetros da rede em sí. É feito via backpropagation (Adam, Gradiente Descendente, etc.)

Dropout regularization

A regularização por ***dropout*** é uma técnica que desativa aleatoriamente uma proporção de neurônios (pré-fixada) em camadas específicas durante o treinamento da rede



Os neurônios são menos dependentes uns dos outros, resultando maior robustez e menos fragilidade para dados ligeiramente distintos (mais generalização).

Ela também pode ser implementada estrategicamente para reduzir a complexidade de ramos específicos, e portanto para diminuir também *overfitting*

REDE NEURAL – *MULTI LAYER PERCEPTRON*

Vantagens

- Capacidade de modelar relações complexas
- Alta capacidade preditiva
- Lida bem com *datasets* grandes
- Adaptabilidade/flexibilidade para os mais diversos problemas

Desvantagens

- Complexidade computacional
- Necessidade de muitos dados
- Propensão a *overfitting*
- Baixa ou nenhuma interpretabilidade
- Complexidade no ajuste dos hiperparametros

Redes Neurais Informadas Pela Física (PINNs)

Redes Neurais Informadas pela Física (PINNs)

As redes neurais também podem ser adaptadas para resolver problemas físicos e aproximar todos os tipos de funções, são as conhecidas redes neurais informadas pela física (PINNs)

Elas são uma abordagem alternativa interessante aos métodos tradicionais para resolver equações diferenciais, por exemplo

A estratégia das PINNs é integrar a equação diferencial e as condições de contorno (cc) diretamente na função de perda durante o treinamento da rede neural

$$L_{total} = L_{cc} + L_{física}$$

Redes Neurais Informadas pela Física (PINNs)

$$L_{total} = L_{cc} + L_{física}$$

L_{cc} representará o erro entre as previsões da rede e os “dados observados” (condições iniciais e de contorno)

$L_{física}$ quantifica o erro residual comparando a rede com o resultado teórico das equações diferenciais (que modelam física do problema)

No final, teremos uma função (aproximação) que resolverá o problema físico elaborado (através da EDO). Quanto mais precisamente abordado o problema, mais próxima da realidade a função gerada pela rede será

PINNs – Exemplo: oscilador harmônico

EDO oscilador harmônico amortecido

$$\frac{d^2x(t)}{dt} + \gamma\frac{dx(t)}{dt} + \omega^2 x(t) = 0 \qquad \gamma = \frac{b}{m},\ \omega^2 = \frac{k}{m}$$

Condições de contorno: $x(t=0) = x_0$ $\quad \dfrac{dx(t=0)}{dt} = v_0$

$$L_{cc} = [\hat{x}(t=0) - x_0]^2 + [\frac{d\hat{x}(t=0)}{dt} - v_0]^2 \qquad L_{física} = \frac{1}{N}\sum_{i=1}^{N}[r_i(t)]^2$$

$$r_i(t) = resíduo_i(t) = rede_i(t) - teórico_i(t)$$

$$r_i(t) = \frac{d^2\hat{x}_i(t)}{dt} + \gamma\frac{d\hat{x}_i(t)}{dt} + \omega^2\hat{x}_i(t) - \cancelto{0}{\frac{d^2x_i(t)}{dt} + \gamma\frac{dx_i(t)}{dt} + \omega^2 x_i(t)}$$

PINNs – oscilador harmônico

EDO oscilador harmônico amortecido

$$\frac{d^2x(t)}{dt} + \gamma\frac{dx(t)}{dt} + \omega^2 x(t) = 0 \qquad \gamma = \frac{b}{m},\ \omega^2 = \frac{k}{m}$$

Condições de contorno: $x(t=0) = x_0$ $\quad \frac{dx(t=0)}{dt} = v_0$

$$L_{total} = [\hat{x}_w(0) - x_0]^2 + [\frac{d\hat{x}_w(0)}{dt} - v_0]^2$$

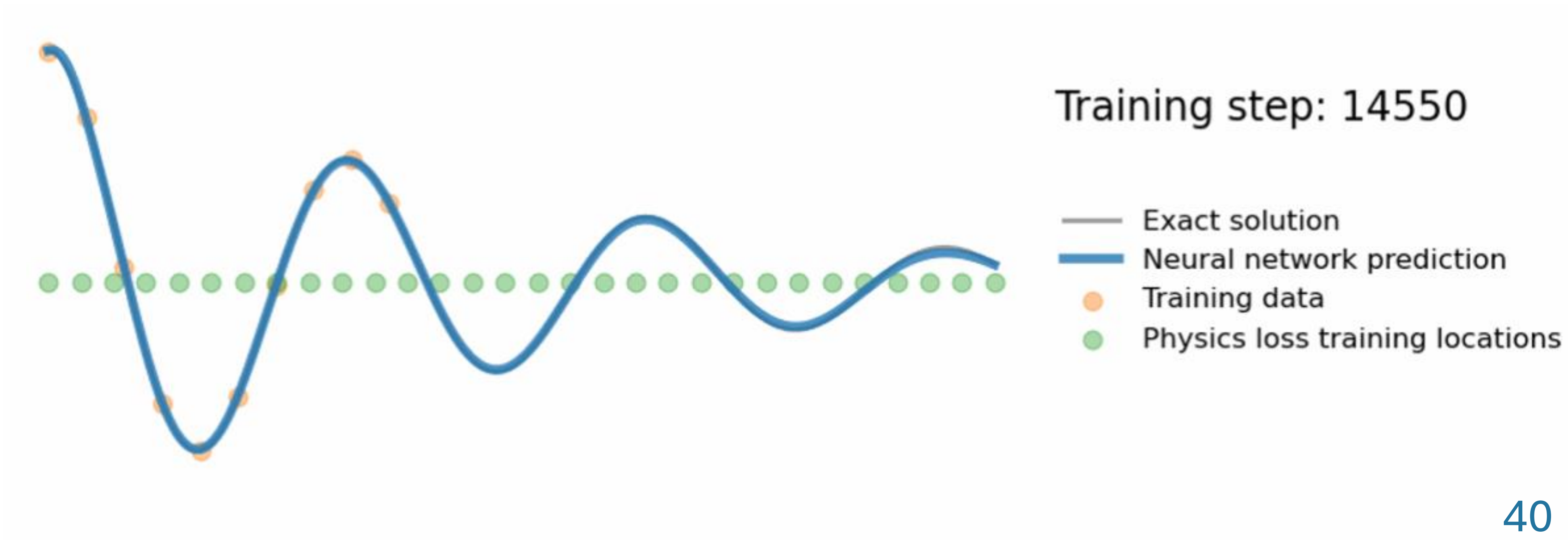
$$+ \frac{1}{N}\sum_{i=1}^{N}[\frac{d^2\hat{x}_i(t)}{dt} + \gamma\frac{d\hat{x}_i(t)}{dt} + \omega^2\hat{x}_i(t)]^2$$

Agora é só treinar a rede como usual...

PINNs – oscilador harmônico

Oscilador Harmônico Amortecido:

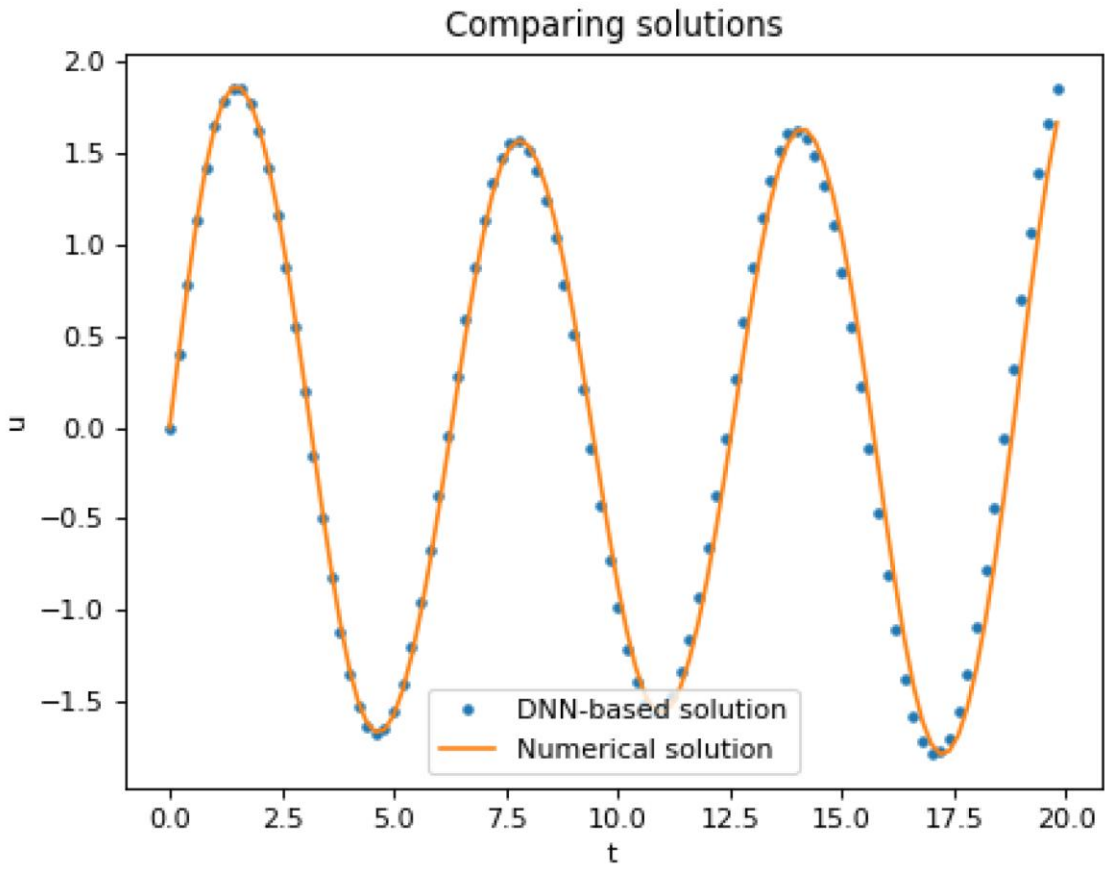
.github

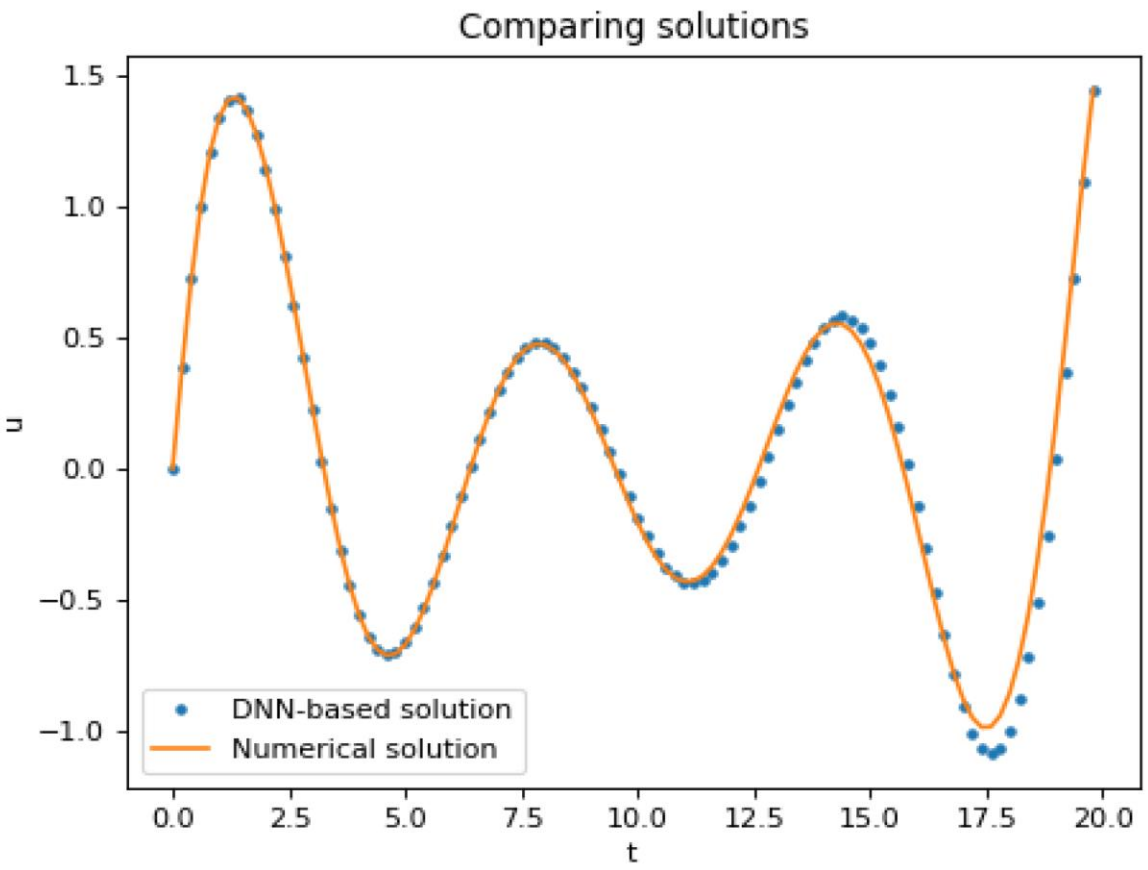


PINNs – oscilador harmônico

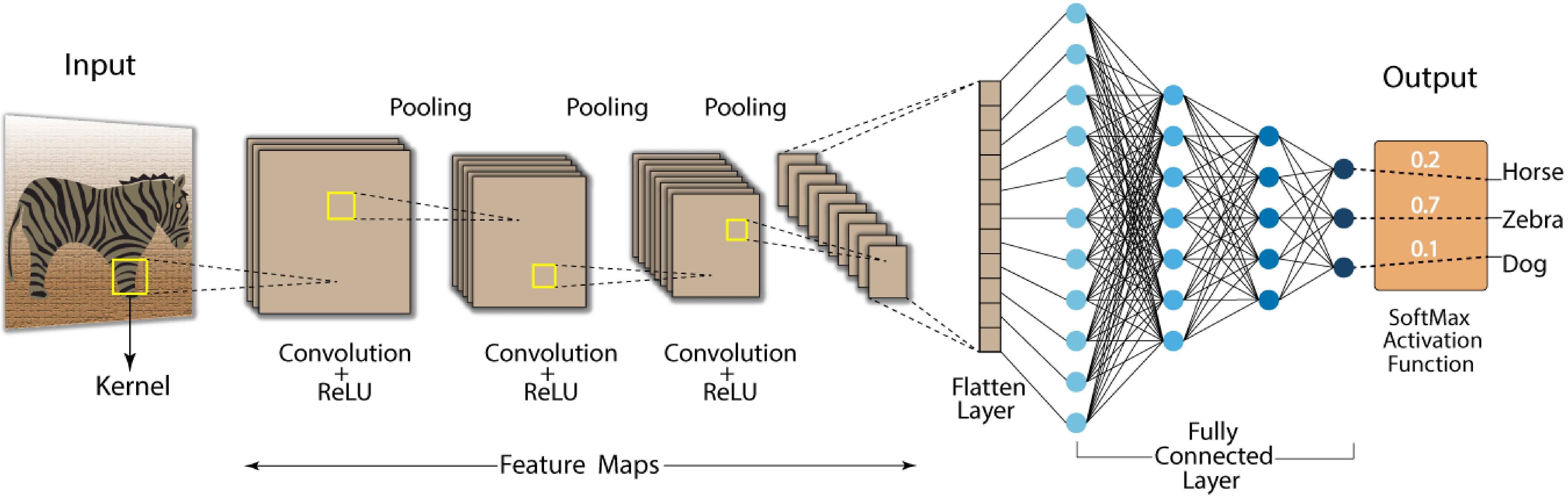
Oscillator Simulation with Deep Neural Networks:

10.3390/math12070959





Redes Neurais Convolucionais



PRÁTICA – GOOGLE COLAB