



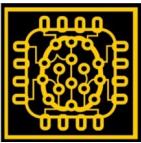
Dr. Prof. Eng. Arnaldo de Carvalho Junior
adecarvalhojr@ifsp.edu.br



- Pós-doutorado em Ciências (Sistemas Eletrônicos) pela POLI/USP (2023)
- Doutor em Ciências (Sistemas Eletrônicos) pela POLI/USP (2021)
- Mestre em Engenharia Mecânica pela UNISANTA (2017).
- Tutoria EAD pela FGV (2004) e Docência Nível Superior pela FGV (2002).
- MBA Em Gestão Empresarial pela Fundação Getúlio Vargas RJ (2001), com extensão de MBA na Universidade da Califórnia – Campus Irvine (2001).
- Professor Licenciado para ensino de nível segundo grau pelo CEFET – Paraná (1995).
- Engenheiro Eletrônico pela UNISANTA (1991).
- Professor Titular EBTT (2022) IFSP Cubatão desde 1992. Professor da UNISANTOS (2003 - 2015) e FORTEC (1990 - 1992).
- Pesquisador do EAILab e dos grupos de pesquisa Labmax e AutomSystem do IFSP.
- É colaborador e possui tutoriais publicados no Site Teleco (www.teleco.com.br), desde 2011.
- Obteve Certificações Cisco Business Transformation (2015), PMI (2012), Wireless CWNA, Cisco CCNA & CCNP de Router & Switches (2011).
- Inglês e Espanhol fluentes. Noções de Francês.
- Atuou profissionalmente em todo o Brasil, EUA, Inglaterra, França, Romênia, China e toda LATAM.
- Possui cursos de Fibras Óticas, Microcontroladores, Redes Wireless, Cisco (CCNA, QoS, VoIP), Gerenciamento de Projetos, entre outros.
- Atuou em empresas como Medidata, Cisco, Alcatel-Lucent (Nokia), MSI (hoje Mentum), Evadin , TV Tribuna (Afiliada Rede Globo), ocupando cargos Técnicos, de Consultoria e Gerencia.

Dr. Prof. Eng. Arnaldo de Carvalho Junior

aedcarvalhojr@ifsp.edu.br²



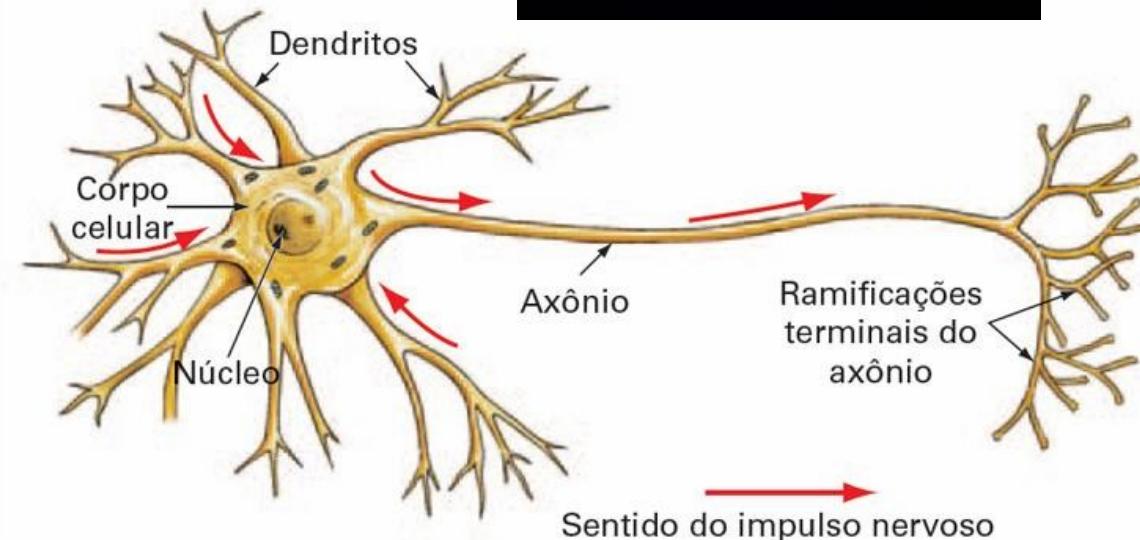
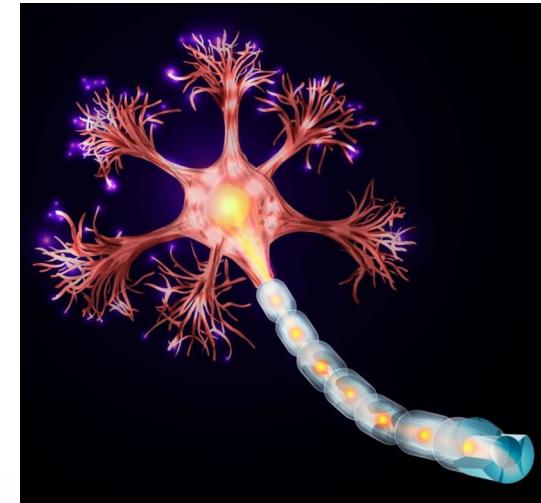
Definição:

- As redes neurais artificiais – RNAs (*artificial neural networks* – ANN) são técnicas computacionais que apresentam um modelo matemático inspirado na estrutura neural de organismos inteligentes e que adquirem conhecimento através da experiência.
- As RNAs são capazes de “aprender” uma determinada função ou reconhecimento de padrões, despertando interesse em identificação e controle de sistemas.
- As Redes Neurais são compostas de uma coleção massivamente paralela de unidades de processamento (Neurônios) pequenas e simples, onde as interligações são responsáveis pela maior parte da “inteligência”.
- Apesar de menos complexa, as RNAs apresentam duas similaridades básicas com as redes neurais biológicas:
 - descrição de seus blocos de construção por dispositivos computacionais simples;
 - conexões entre os neurônios determinam a função da rede.



Neurônio Biológico

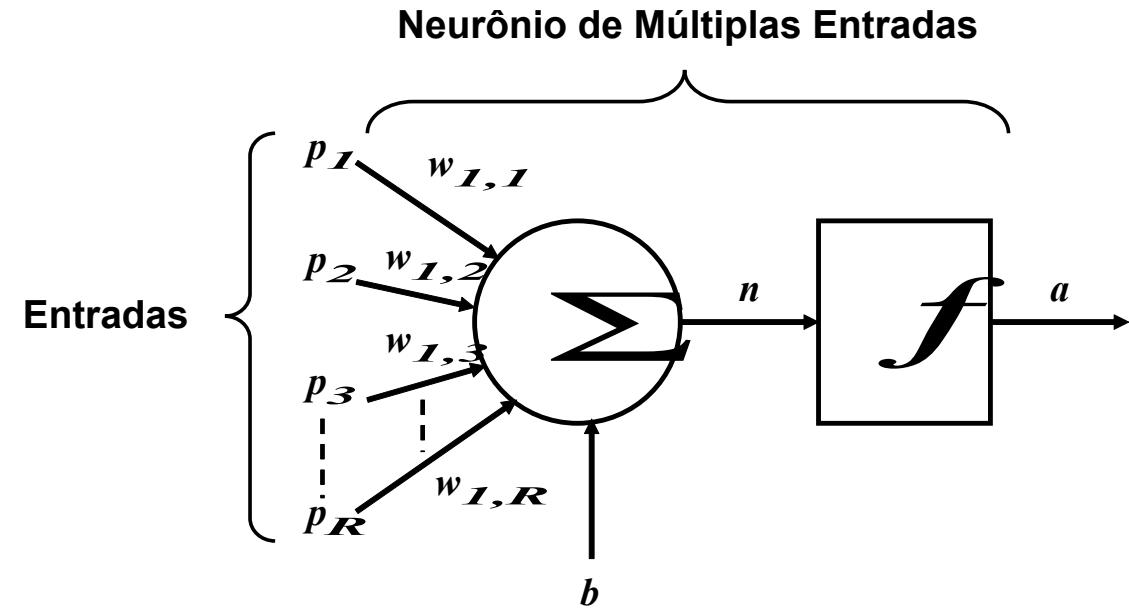
- Os **dendritos** são filamentos responsáveis por receber os sinais de informações para as células.
- O **corpo celular**, ou soma, reúne as informações recebidas pelos dendritos.
- O **axônio** transporta os impulsos elétricos que partem do corpo celular até as diversas ramificações com dilatações bulbosas conhecidas como terminais axônicos (ou terminais nervosos), que estabelecem as conexões sinápticas com outras células.





Neurônio Artificial

- Neurônios artificiais são algoritmos cujas funções matemáticas são inspiradas em neurônios biológicos e constituem a base das RNAs.
- Os pesos (w) representam a força das sinapses e são multiplicados aos valores das respectivas entradas e somados com um valor de ajuste ou bias (b).
- O resultado desta soma (n) é então aplicado a uma função de ativação (f) e apresentado na saída (a) do neurônio artificial.

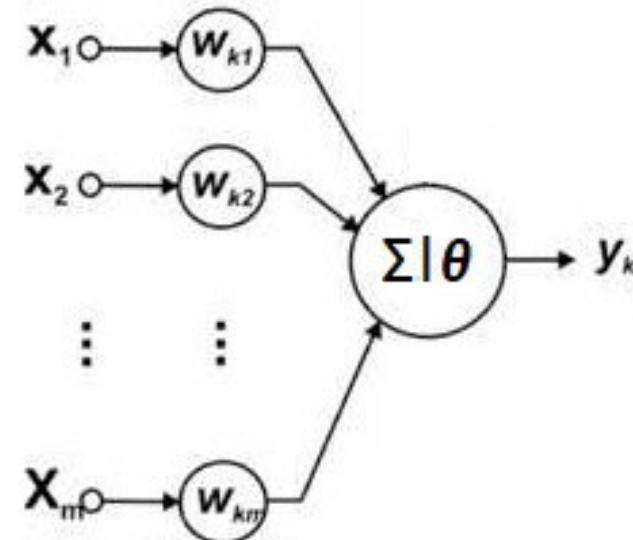


$$n = w_{1,1}p_1 + w_{1,2}p_2 + w_{1,3}p_3 + \cdots + w_{1,R}p_R$$
$$a = f(Wp + b)$$

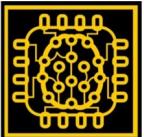


Neurônio Artificial:

- Os primeiros trabalhos na área datam de 1943: McCulloch e Pitts (MCP) desenvolveram o primeiro modelo matemático do neurônio.
- Neste modelo os neurônios são unidades de processamento simples com uma função de ativação degrau (Limiar θ).

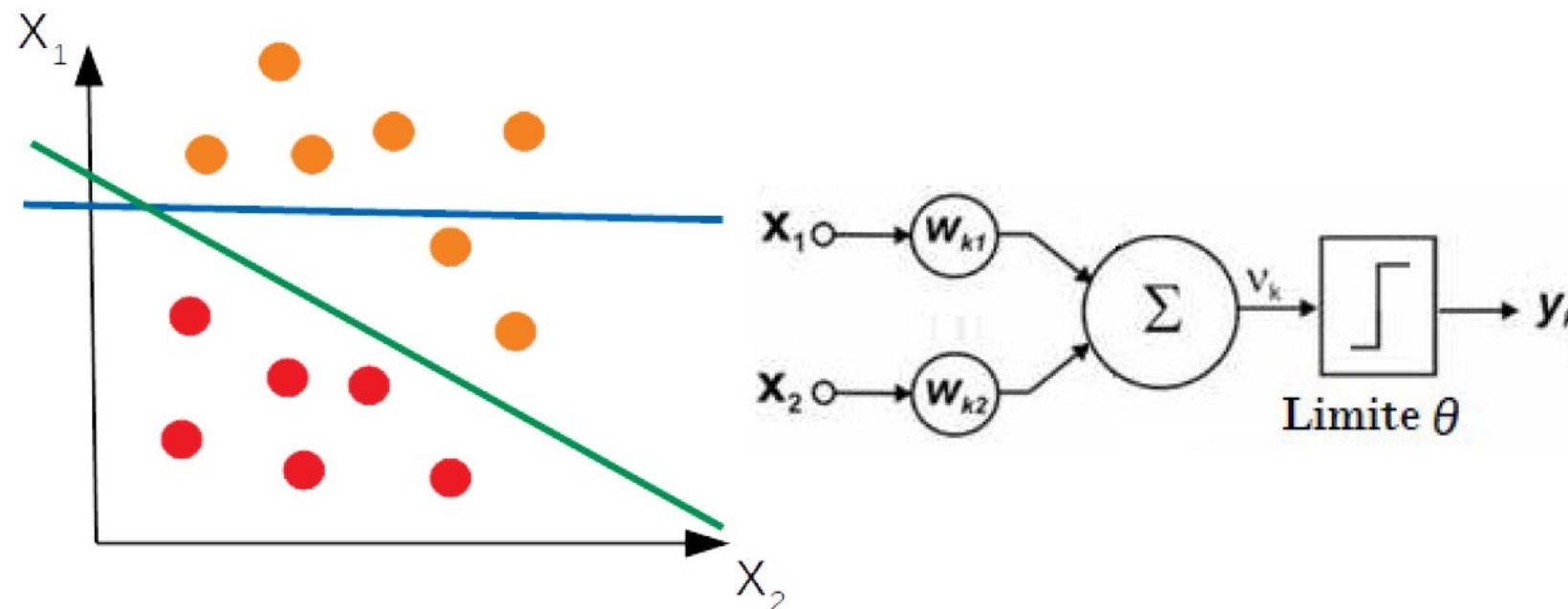


$$y_k = \begin{cases} 1, & \sum_{i=1}^n w_{ki} x_i \geq \theta \\ 0, & \sum_{i=1}^n w_{ki} x_i < \theta \end{cases}$$



Neurônio Artificial:

- Neurônios com diferentes pesos em suas entradas e diferentes valores de limiar produzem diferentes partições no espaço de entradas.





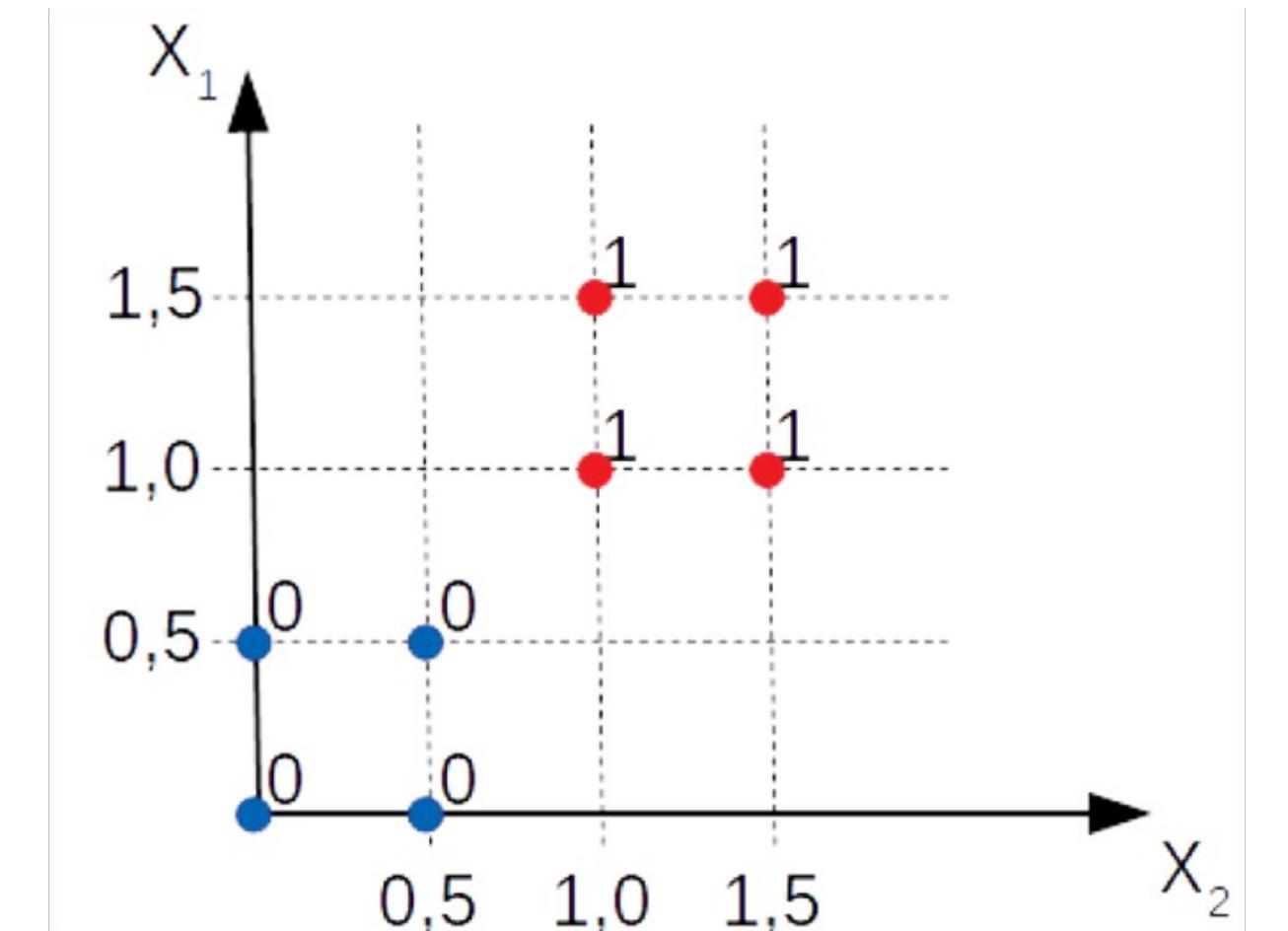
Neurônio Artificial:

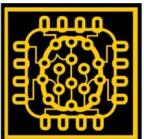
- Exemplo de 2 entradas e 1 saída

$$y_k = 1, \text{ se } w_{k1}x_1 + w_{k2}x_2 \geq \theta$$
$$y_k = 0, \text{ se } w_{k1}x_1 + w_{k2}x_2 < \theta$$

- Considerando-se $w_{k1} = w_{k2} = 1$ e $\theta = 2$

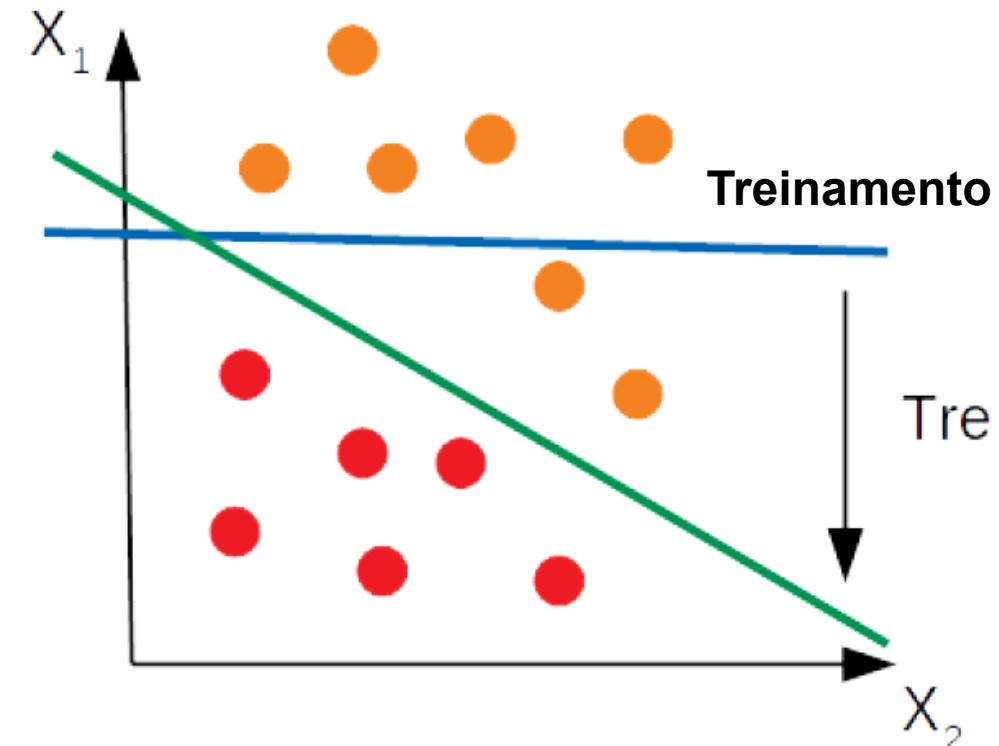
X_1	X_2	Y_k
0	0	0
0	0,5	0
0,5	0	0
0,5	0,5	0
1	1	1
1	1,5	1
1,5	1	1
1,5	1,5	1





Neurônio Artificial:

- A saída (y) é uma função da combinação linear das entradas (x).
- O treinamento de um neurônio e consequentemente de uma RNA acontece através da alteração dos parâmetros w_{k1} , w_{k2} e θ .
- Levando em consideração que a alteração dos parâmetros w e Θ (treinamento) modifica a posição da reta e portanto da partição no espaço de entrada é possível observar que a estrutura de uma RNA é adequada para resolver problemas de classificação.

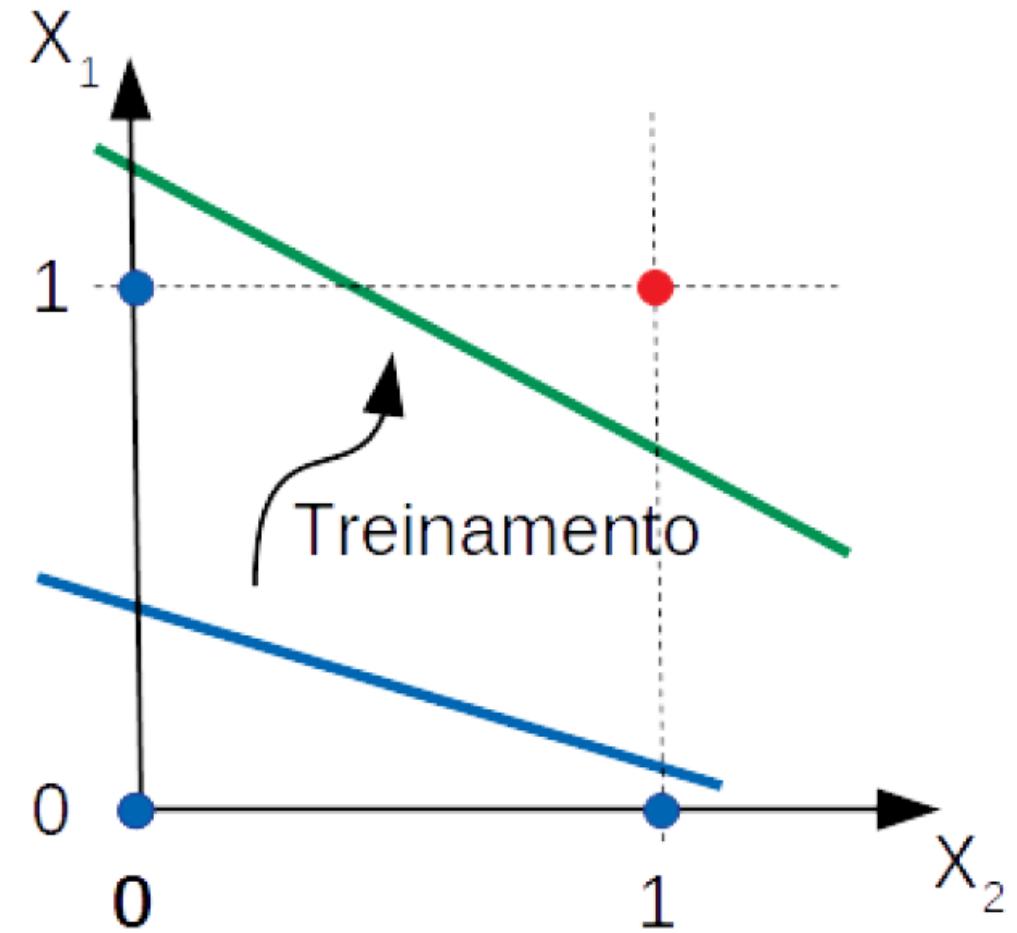


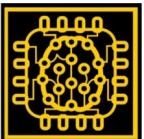


Resolução de Problemas:

- Um neurônio artificial é capaz de resolver apenas problemas simples ou linearmente separáveis.
- Exemplo: **AND** lógico

X_1	X_2	Y_k
1	1	1
0	1	0
1	0	0
0	0	0

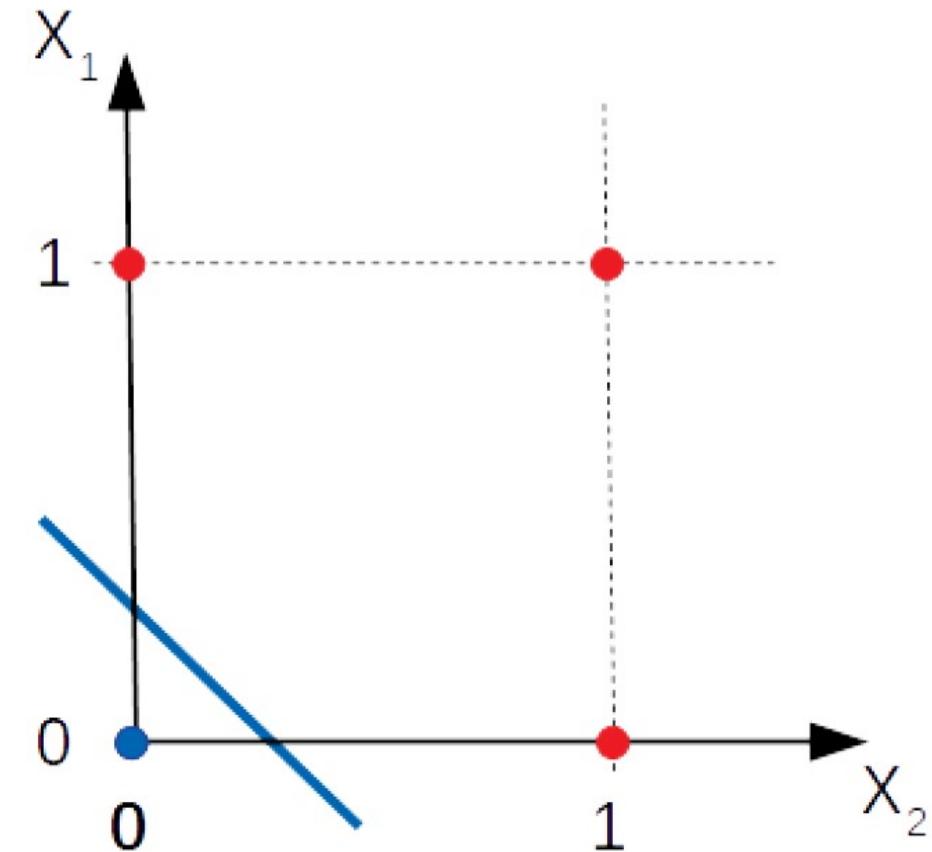


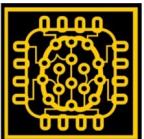


Resolução de Problemas:

- Um neurônio artificial é capaz de resolver apenas problemas simples ou linearmente separáveis.
- Exemplo: **OR** lógico

X_1	X_2	Y_k
1	1	1
0	1	1
1	0	1
0	0	0

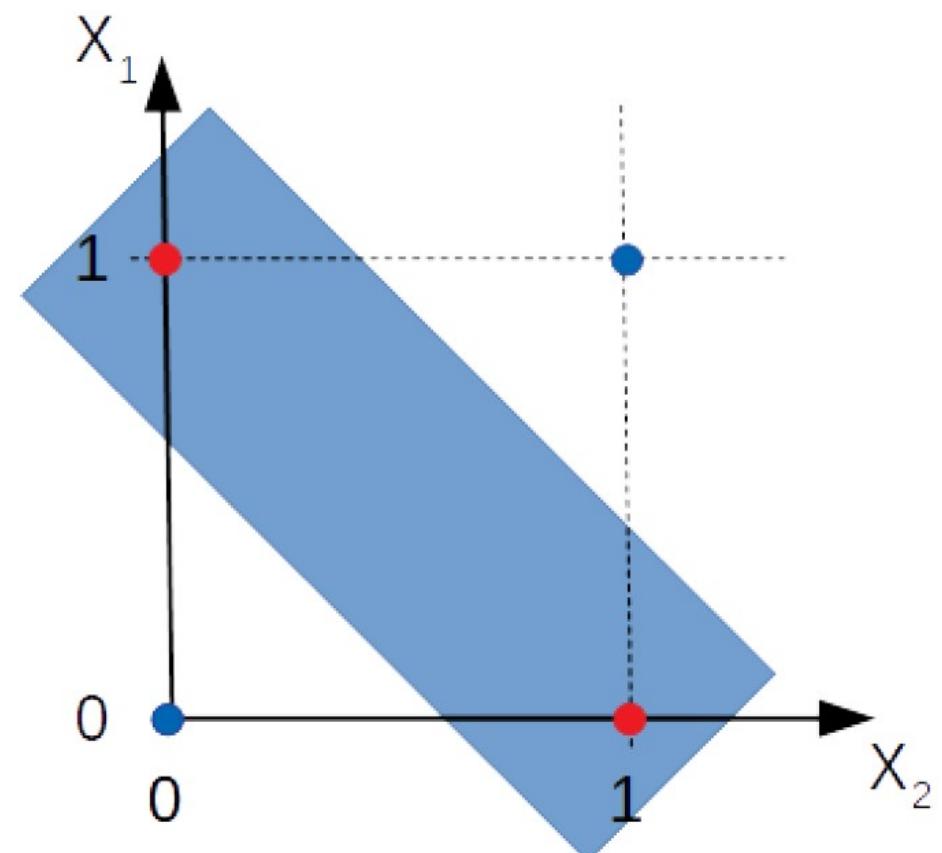




Resolução de Problemas:

- Um neurônio artificial sozinho **não** é capaz de resolver problemas não linearmente separáveis.
- Exemplo: **XOR** lógico

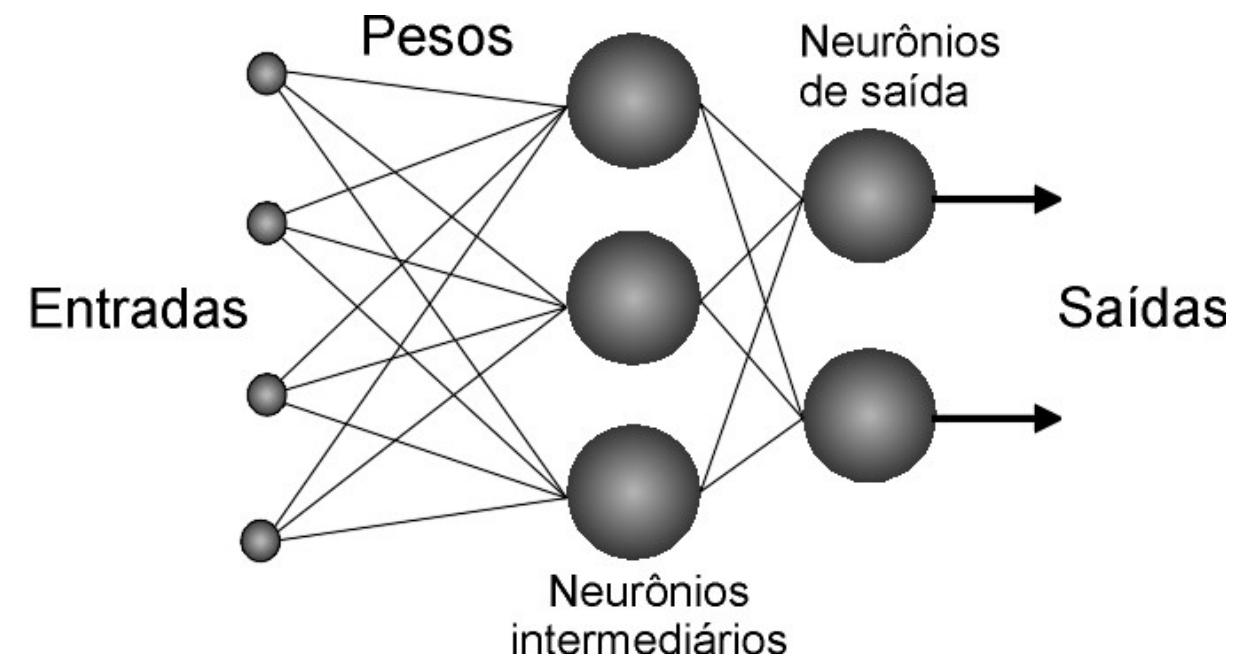
X_1	X_2	Y_k
1	1	0
0	1	1
1	0	1
0	0	0

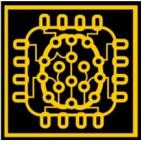




Resolução de Problemas - RNAs:

- A adição de neurônios em paralelo não resolve o problema.
- A solução é aumentar o número de camadas de neurônios, formando as RNAs.
- Pode ser demonstrado que se a função de ativação for **não linear**, uma RNA de duas camadas pode ser um **aproximador universal** de função.
- As redes neurais mais modernas são compostas por múltiplas camadas de neurônios e possibilitam a resolução de problemas não linearmente separáveis.





Aplicações das RNAs:

- Diagnóstico: Médico, Falhas de Sistemas etc.;
- Previsão de Séries Temporais: Cotações da Bolsa de Valores, Dados Econômicos, Meteorologia, etc.;
- Processamento de Linguagem Natural;
- Sistemas de Controle e Automação;
- Reconhecimento e Síntese de Voz;
- Processamento de Sinais e Imagens: Radar, Sensores, Imagens de satélite etc.;
- Reconhecimento ótico de caracteres (OCR);
- Controle de processos industriais;
- Piloto automático;
- Reprodução da fala.



Características das RNAs

- **Armazenar informações em toda a rede**
 - As informações são armazenadas na rede e não em um banco de dados.
 - Se algumas informações desaparecem de um lugar, isso não impede o funcionamento de toda a rede.
- **Capacidade de trabalhar com conhecimento insuficiente**
 - Após o treinamento da RNA, a saída produzida pelos dados pode ser incompleta ou insuficiente.
 - A importância dessa informação em falta determina a falta de desempenho.
- **Boa tolerância a falhas**
 - A geração de saída não é afetada pela corrupção de uma ou mais células da rede neural artificial.
 - Isso torna as redes melhores em tolerar falhas.



Características das RNAs (cont.)

- **Memória distribuída**
 - Para que uma rede neural artificial se torne capaz de aprender, é necessário delinear os exemplos e ensiná-la de acordo com a saída desejada, mostrando esses exemplos para a rede.
 - O progresso da rede é diretamente proporcional às instâncias selecionadas.
- **Corrupção Gradual**
 - De fato, uma rede pode sofrer uma degradação relativa e fica mais lenta com o tempo, mas não corrói imediatamente a rede.
- **Capacidade de treinar máquina**
 - As RNAs aprendem com os eventos e tomam decisões comentando eventos semelhantes.
- **A capacidade de processamento paralelo**
 - Possuem força numérica que as tornam capazes de realizar mais de uma função ao mesmo tempo.



Aplicações de RNAs

As RNAs podem ser utilizadas em diferentes campos. As tarefas realizadas por RNAs tendem a se enquadrar nas seguintes categorias:

- **Aproximação de Função:**
 - Também chamada de análise de regressão, incluindo identificação, modelagem e predição de séries temporais.
- **Classificação**
 - Incluindo reconhecimento e sequência de padrões, detecção de novidades e tomada de decisão sequencial.
- **Processamento de Dados**
 - Incluindo filtragem, agrupamento, separação e compressão de sinal cego.



Vantagens das RNAs:

- Aplicações em Sistemas Adaptativos;
- Aplicadas em tarefas onde tem-se bases de exemplos disponíveis, realizando a aquisição automática de conhecimentos;
- Associação de padrões de entradas e saída;
- Classificação de padrões de forma supervisionada ou não;
- Aproximação de funções desconhecidas através de amostras destas funções;
- Trabalhar com dados aproximados, incompletos e inexatos;
- Paralelismo, generalização, robustez;
- “Tarefas complexas realizadas por seres humanos”.

Limitações das RNAs:

- Composição e construção de conhecimentos estruturados;
- Dificuldade de explicitação dos conhecimentos adquiridos;
- Dificuldade para definir a estrutura da rede, seus parâmetros e a base de dados;
- Falta de garantia de uma convergência do algoritmo para uma solução ótima.
- Quanto maior e mais complexa a estrutura da RNA, maior a dificuldade de aprendizagem e consumo de recursos computacionais para o seu treinamento.

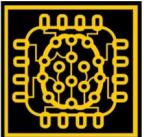


Parâmetros que definem a arquitetura da RNA:

- Arquitetura da RNA e número de camadas;
- Função de Ativação utilizada;
- Número de neurônios em cada camada;
- Tipo de conexão entre os neurônios;
 - Feedforward
 - Feedback
- Conectividade da Rede;
 - Parcialmente conectada
 - Completamente conectada
- Taxa de aprendizagem (*learning rate*) utilizada pelo algoritmo de treinamento da RNA;
- Número de épocas (*epochs*) ou ciclos de treinamento.

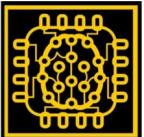
Dr. Prof. Eng. Arnaldo de Carvalho Junior

aedecarvalhojr@ifsp.edu.br 19



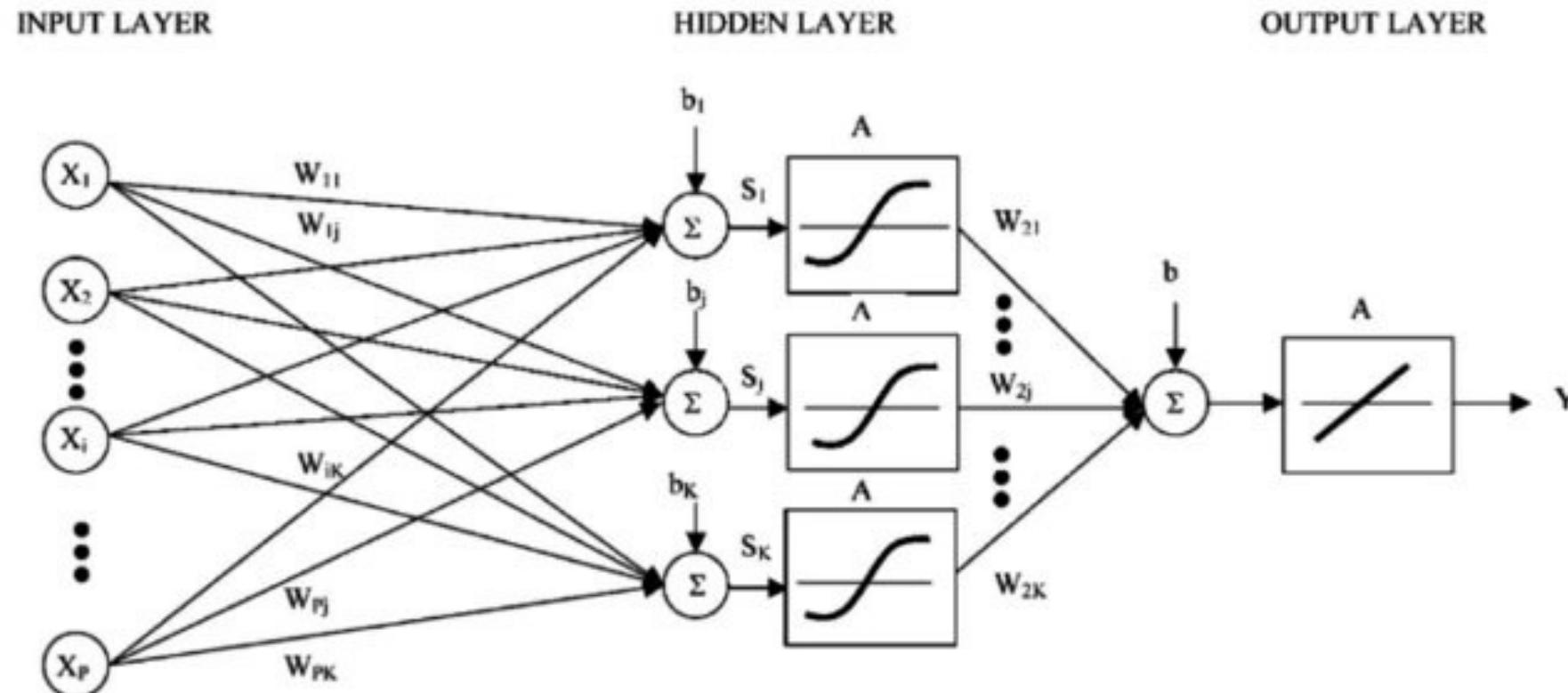
Arquitetura das RNAs

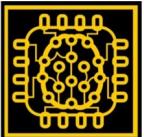
- Existem diferentes arquiteturas de RNAs.
 - Multilayer perceptron (MLP): Classificação e aproximação de funções
 - Rede neural recorrente (*recurrent neural network* – RNN): séries temporais
 - Rede neural convolucional (*convolutional neural network* – CNN): tratamento e classificação de imagens
 - Redes de aprendizado profundo (*deep learning*): RNAs de múltiplas camadas intermediárias e processamento intenso, para análises de problemas complexos.
- Estudar post EAILab: “Redes Neurais Artificiais: Algoritmos poderosos para aplicações de IA e ML”, disponível em: <https://eailab.labmax.org/2024/04/03/redes-neurais-artificiais-algoritmos-poderosos-para-aplicacoes-de-ia-e-ml/>



Arquitetura das RNAs

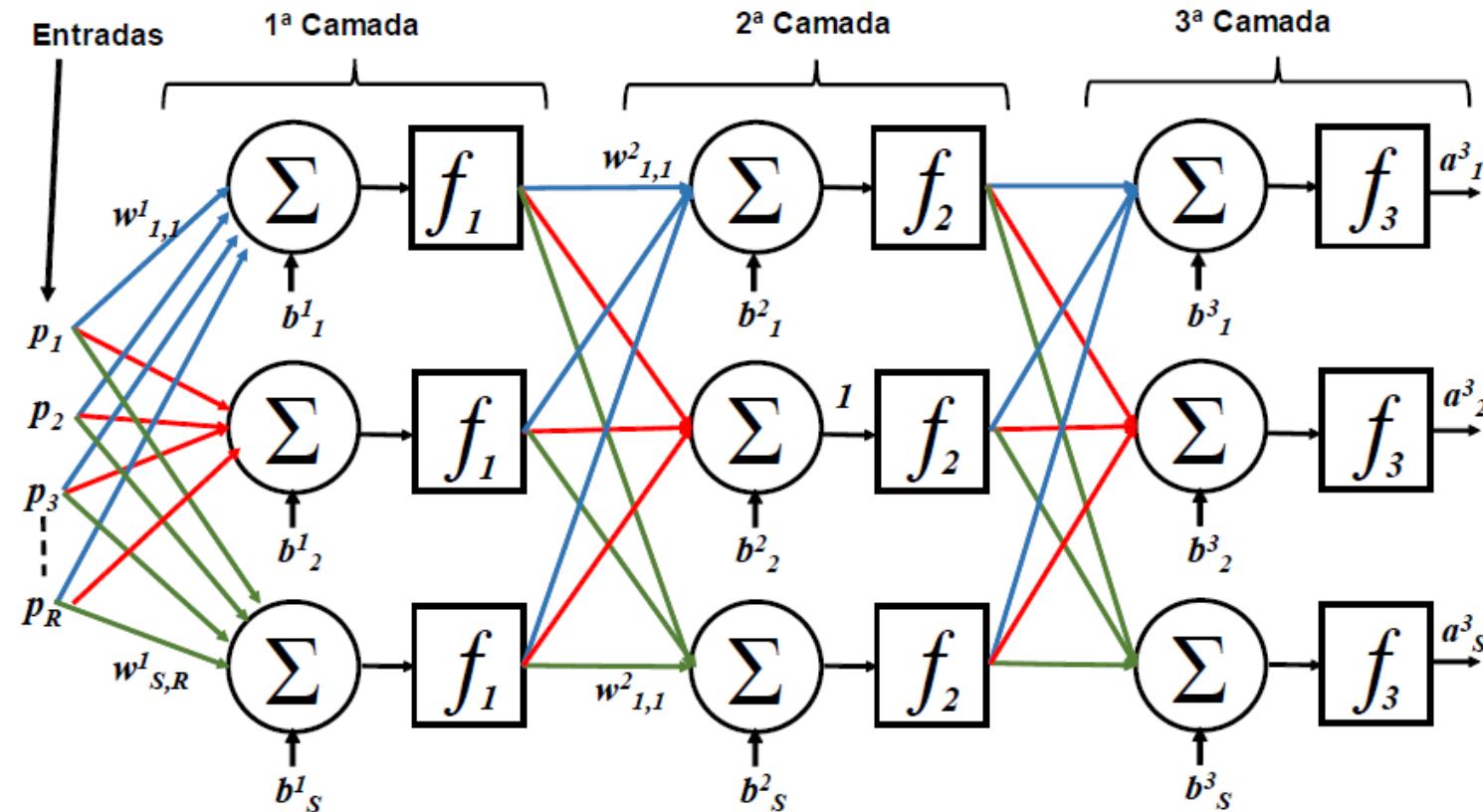
- Exemplo de RNA do tipo *feed-forward* com função *tanh* na camada oculta e linear na camada de saída:





Arquitetura das RNAs

- Exemplo de RNA multicamadas ou profunda (*deep learning*)





Neurônio Artificial

- Várias funções (f) podem ser utilizadas, de acordo com a finalidade,
 - Exemplo: limite binário (*hard limit*), linear, log-sigmoide (σ), sigmoide tangente hiperbólica (\tanh), unidade linear retificada (*rectified linear unit* – ReLU) e suas variações como Leaky-ReLU (L-ReLU), PAL2v, entre outras .
- A combinação de **pesos**, **bias** e f permite que o neurônio aproxime qualquer função.
- É importante que a função f adotada seja *derivável*, de modo a permitir a elaboração de algoritmos de regressão para a calibração (“aprendizado”) dos pesos e *bias* do neurônio.
- Pode ser demonstrado que se a função de ativação for *não linear*, uma RNA de duas camadas pode ser um *aproximador universal* de função.
- Pode-se utilizar funções de ativação diferentes em cada camada. Por exemplo, tangente hiperbólica na camada oculta (*hidden layer*) e linear na camada de saída (*output layer*).
- Estudar post EAILab: “Função de Ativação, o Núcleo da Composição de Neurônios Artificiais”, disponível em: <https://eailab.labmax.org/2024/02/28/funcao-de-ativacao-o-nucleo-da-composicao-de-neuronios-articiais/>

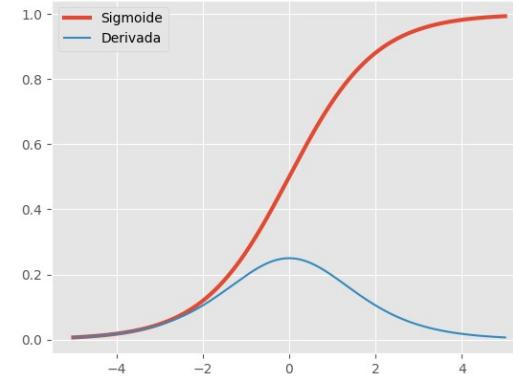


Funções de Ativação

- Log-sigmoide:

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

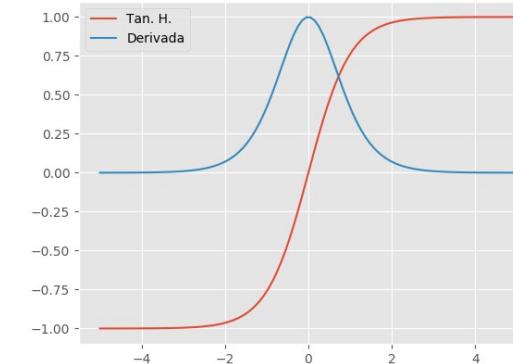
$$\sigma'(n) = \sigma(n)(1 - \sigma'(n))$$



- Tangente Hiperbólica (\tanh):

$$a = \tanh(n) = \frac{e^n - e^{-n}}{e^n + e^{-n}}$$

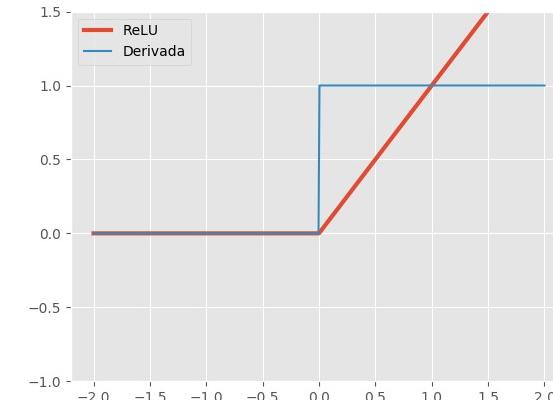
$$\tanh'(n) = 1 - \tanh^2(n)$$



- Unidade Retificadora Linear (ReLU):

$$a = \text{ReLU}(n) = \begin{cases} 0 & n \leq 0 \\ n & n > 0 \end{cases}$$

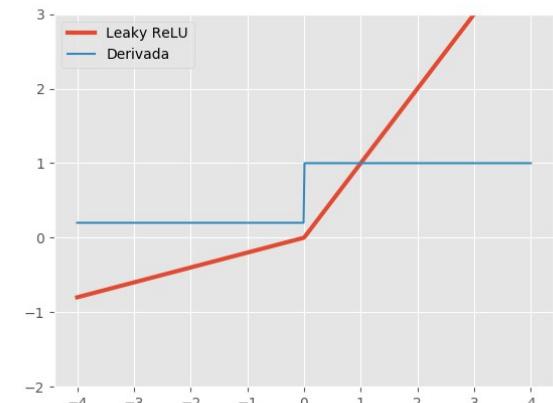
$$\text{ReLU}'(n) = \begin{cases} 0 & n \leq 0 \\ 1 & n > 0 \end{cases}$$



- Leaky-ReLU (LReLU):

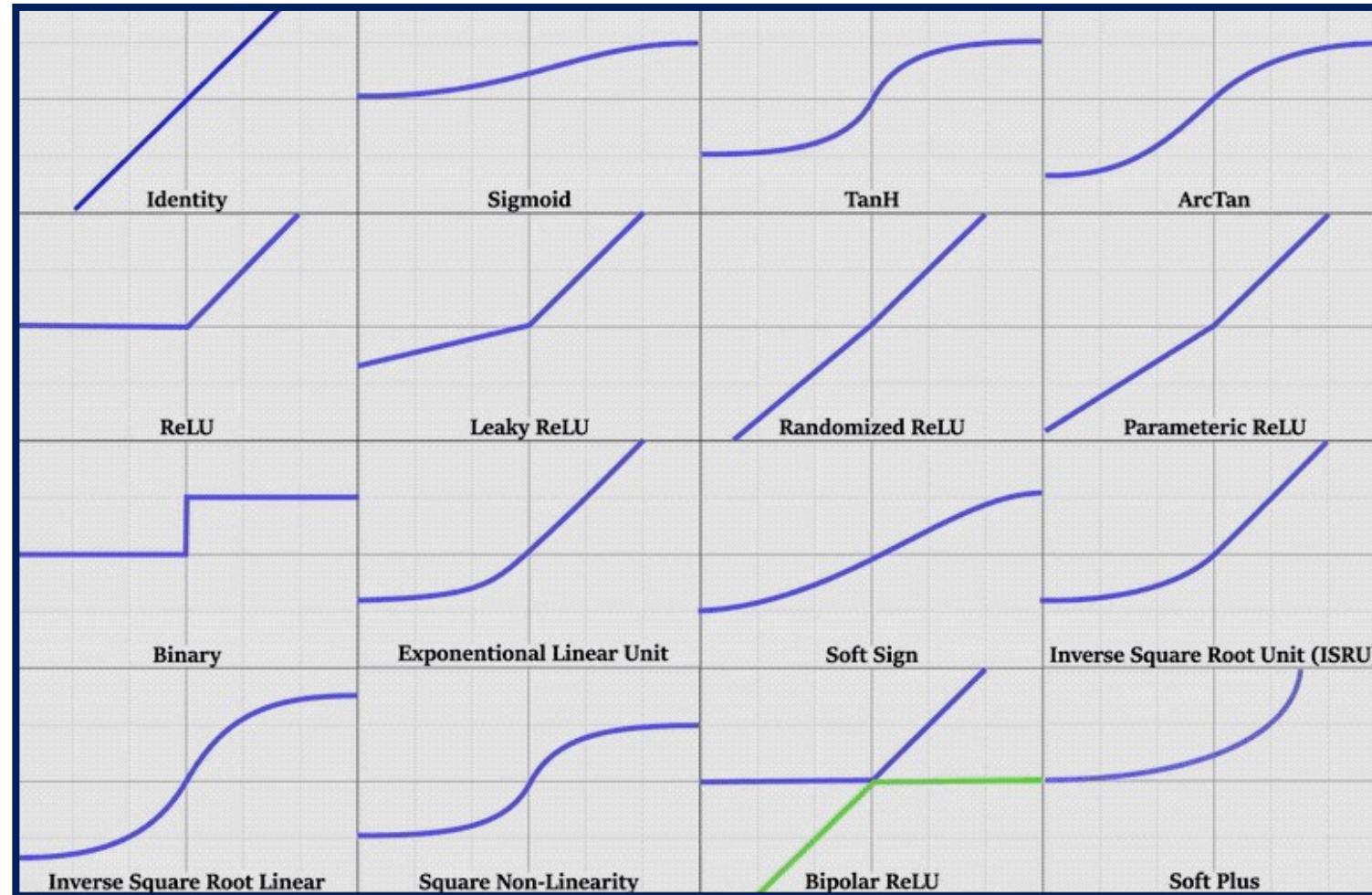
$$a = \text{LReLU}(n) = \begin{cases} \alpha n & n \leq 0 \\ n & n > 0 \end{cases}$$

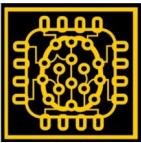
$$\text{LReLU}'(n) = \begin{cases} \alpha & n \leq 0 \\ 1 & n > 0 \end{cases}$$





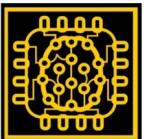
Funções de Ativação





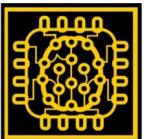
Neurônio Artificial

- A **log-sigmoide (σ)** é a função de ativação mais popular de RNAs, sua saída varia entre [0 , +1];
- A **tanh** também é muito popular e sua saída varia entre [-1 , +1];
- A **ReLU** é equivalente à função identidade, para valores de entrada > 1 , requer poucos recursos computacionais, é de aprendizado rápido, porém apresenta efeito de neurônio morto (*dead neuron*);
- A **L-ReLU** reduz o efeito de dead neuron, introduzindo o fator α com valores entre [+0.01 , +0.2];
- Funções de base radial (**RBF**) tem sido cada vez mais utilizadas em RNAs.
 - **Vantagens:** RBFNN apresenta design fácil, treinamento rápido, boa generalização e robustez para ruído de entrada. São especialmente recomendadas em problemas de aproximação de função, para problemas de aproximação de função, sendo especialmente recomendadas para superfícies com picos e vales regulares.
 - **Desvantagens:** Existem várias funções de base radial, tornando difícil a escolha, a RBFNN tende a aumentar a complexidade conforme a quantidade de neurônios na camada oculta. Outro desafio da RBF está em seu algoritmo e estrutura de treinamento, não permitindo modelar um sistema fortemente não linear.



Aprendizado de RNA

- **Aprendizado Supervisionado:** consiste de um conjunto de treino, com entradas correspondentes e saídas desejadas. A regra de aprendizado é utilizada para ajustar pesos e bias da rede neural de modo que as saídas da RNA se aproximem das saídas desejadas;
- **Aprendizado por Reforço:** Similar ao aprendizado supervisionado, exceto que, em vez de receber a saída correta para cada entrada da rede, o algoritmo recebe apenas uma pontuação (*grade* ou *score*), como uma medida do desempenho da rede para uma determinada sequência de entradas;
- **Aprendizado Não-supervisionado:** A rede atualiza seus pesos e bias apenas em resposta às entradas da rede. Não há saída alvo disponível. Em geral, algoritmos de aprendizado não-supervisionado executam algum tipo de operação de agrupamento (*clustering*). Eles aprendem a categorizar os padrões de entrada em um número finito de classes.

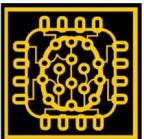


Modos de Treinamento de RNA

- **Modo Sequencial**

- Também conhecido como modo on-line ou Modo Gradiente Estocástico Descendente (*Stochastic Gradient Descent* - SGD)
- A correção das sinapses é aplicada a cada padrão apresentado em cada época (epoch);
- Lida bem com padrões redundantes na base de dados;
- Padrões devem ser apresentados à rede de forma aleatória
- Requer uma quantidade menor de memória quando comparado ao processo em lote (batelada);

$$\Delta w_{ji}(n) = -\eta \sum_{t=0}^n \alpha^{n-t} \frac{\partial \mathcal{E}(n)}{\partial w_{ji}^{(t)}}$$



Modos de Treinamento de RNA

- **Modo Batelada (*Batch Mode*)**

- Todos os dados de treinamento são levados em consideração para dar uma única etapa. A correção das sinapses é aplicada ao final de uma época;
- Toma-se a média dos gradientes de todos os exemplos de treinamento e então usa-se esse gradiente médio para atualizar nossos parâmetros. Então, isso é apenas um passo de descida de gradiente em uma época.
- Algoritmo é ótimo para coletores de erro convexos ou relativamente suaves. Nesse caso, move-se um pouco diretamente em direção a uma solução ótima.
- Permite uma estimativa mais acurada do vetor gradiente local;
- Permite uma paralelização mais fácil da rede.

$$\Delta w_{ji} = -\frac{\eta}{N} \sum_{n=1}^N e_j(n) \frac{\partial e_j(n)}{\partial w_{ji}}$$



Modos de Treinamento de RNA

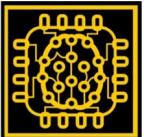
- **Modo Mini-Batelada (*Mini-Batch Mode*)**

- Procura unir as vantagens dos 2 métodos anteriores;
- Usa-se um subconjunto do conjunto de dados para dar mais uma etapa no processo de aprendizado.
- Portanto, o *mini-batch* pode ter um valor maior que um e menor que o tamanho do conjunto de treinamento completo.
- Ao invés de esperar que o modelo calcule todo o conjunto de dados, pode-se atualizar seus parâmetros com mais frequência.
- Reduz o risco de ficar preso em um mínimo local, pois lotes diferentes serão considerados a cada iteração, garantindo uma convergência robusta.

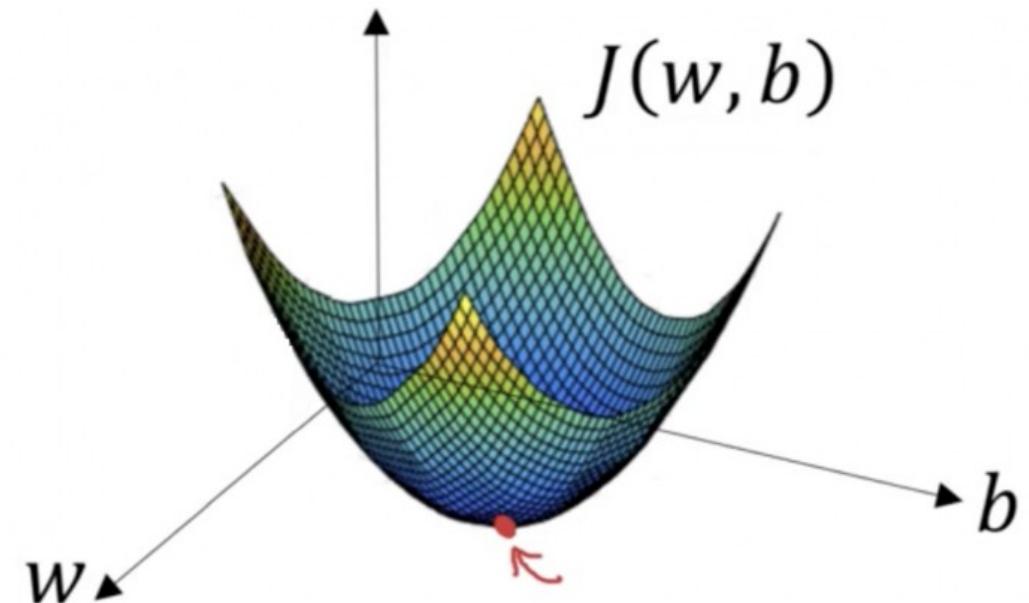
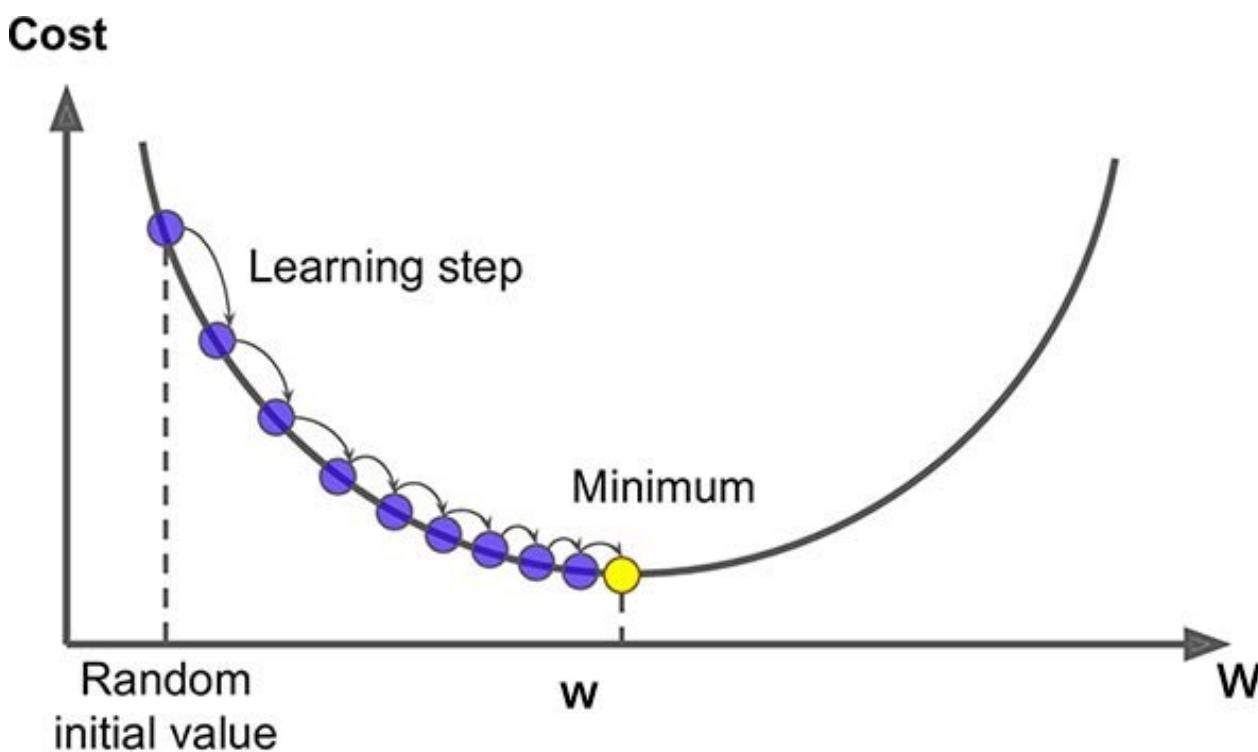


Algoritmos de Aprendizado das RNAs

- Existem diferentes algoritmos de otimização de RNAs.
- Eles diferem em uso de memória, processamento, velocidade e precisão numérica.
- Exemplos de algoritmos de treinamento:
 - Gradiente Descendente (*gradient descent*);
 - Método Newton (*Newton method*);
 - Método Quasi-Newton;
 - **Conjugado do Gradiente (*Conjugate gradient*)**;
 - Levenberg-Marquardt.
- Estudar post NeuralDesigner: “5 algorithms to train a neural network”, disponível em: https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network/



Aprendizado por Gradiente Descendente



<https://builtin.com/data-science/gradient-descent>

Dr. Prof. Eng. Arnaldo de Carvalho Junior

adecorvalhojr@ifsp.edu.br 32



Aprendizado de RNA

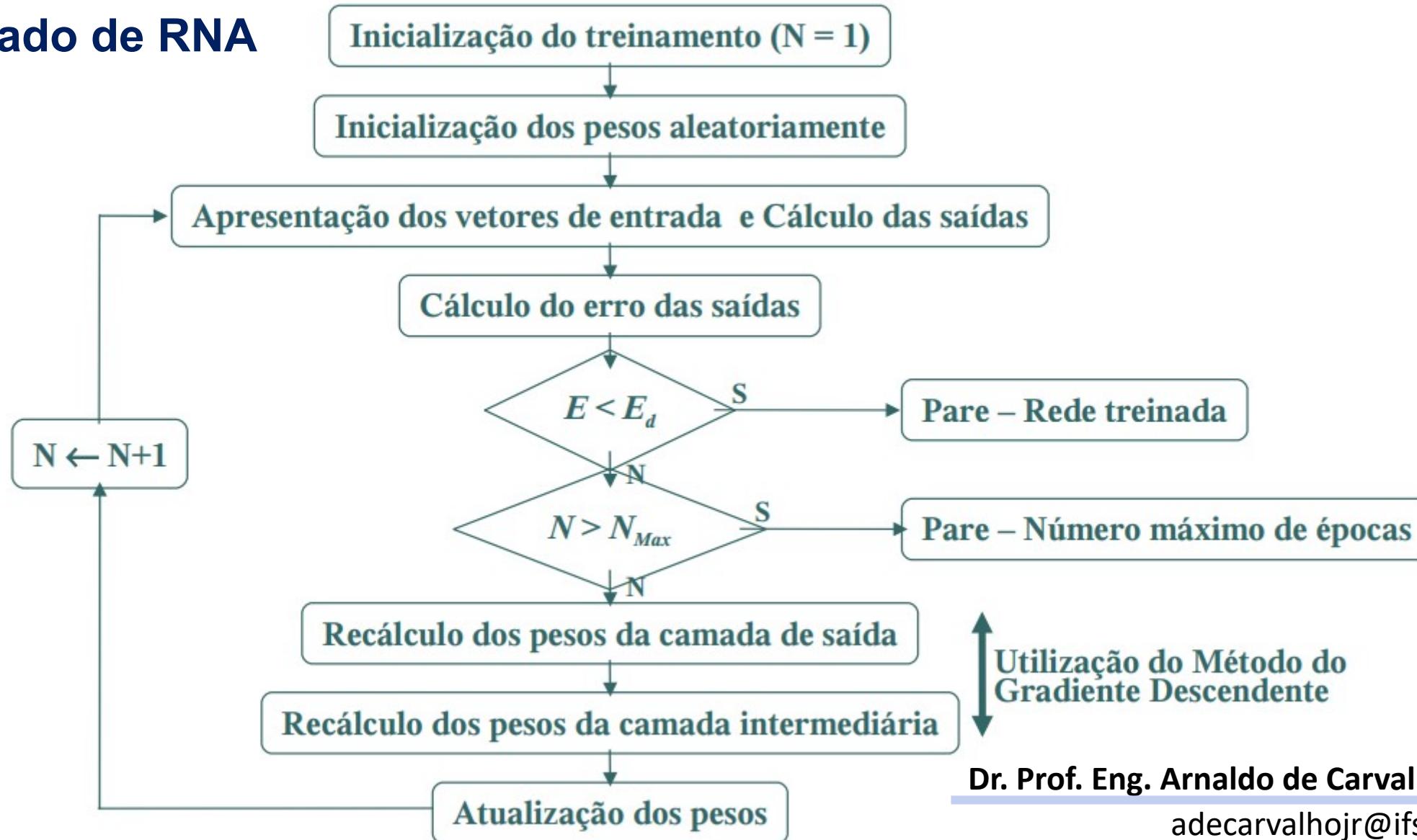
Método Clássico: *Feed-forward* e *Backpropagation*

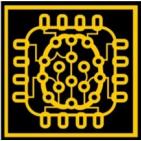
1. Seleciona-se os valores de entrada e saída;
2. Atribui pesos randomicamente;
3. Executa a rede no modo sentido *feed-forward*;
4. Calcula-se o Erro Médio Quadrático (*mean square error* - MSE) entre a saída alvo (target) e a calculada pela RNA;
5. Propaga-se a derivada do erro no sentido inverso (*backpropagation*);
6. Retorna ao passo 3 até atingir o erro mínimo (*mean square error* - MSE);
7. Memoriza-se os pesos e bias finais da RNA “treinada”.



Aprendizado de RNA

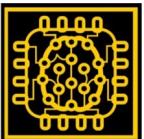
Exemplo:





Critérios de parada

- Finalizar o treinamento após n ciclos
- Finalizar o treinamento após o MSE ficar abaixo de uma constante α
- Finalizar o treinamento quando a porcentagem de classificações corretas estiver acima de uma constante α (mais indicado para saídas binárias)
- Combinação dos métodos acima



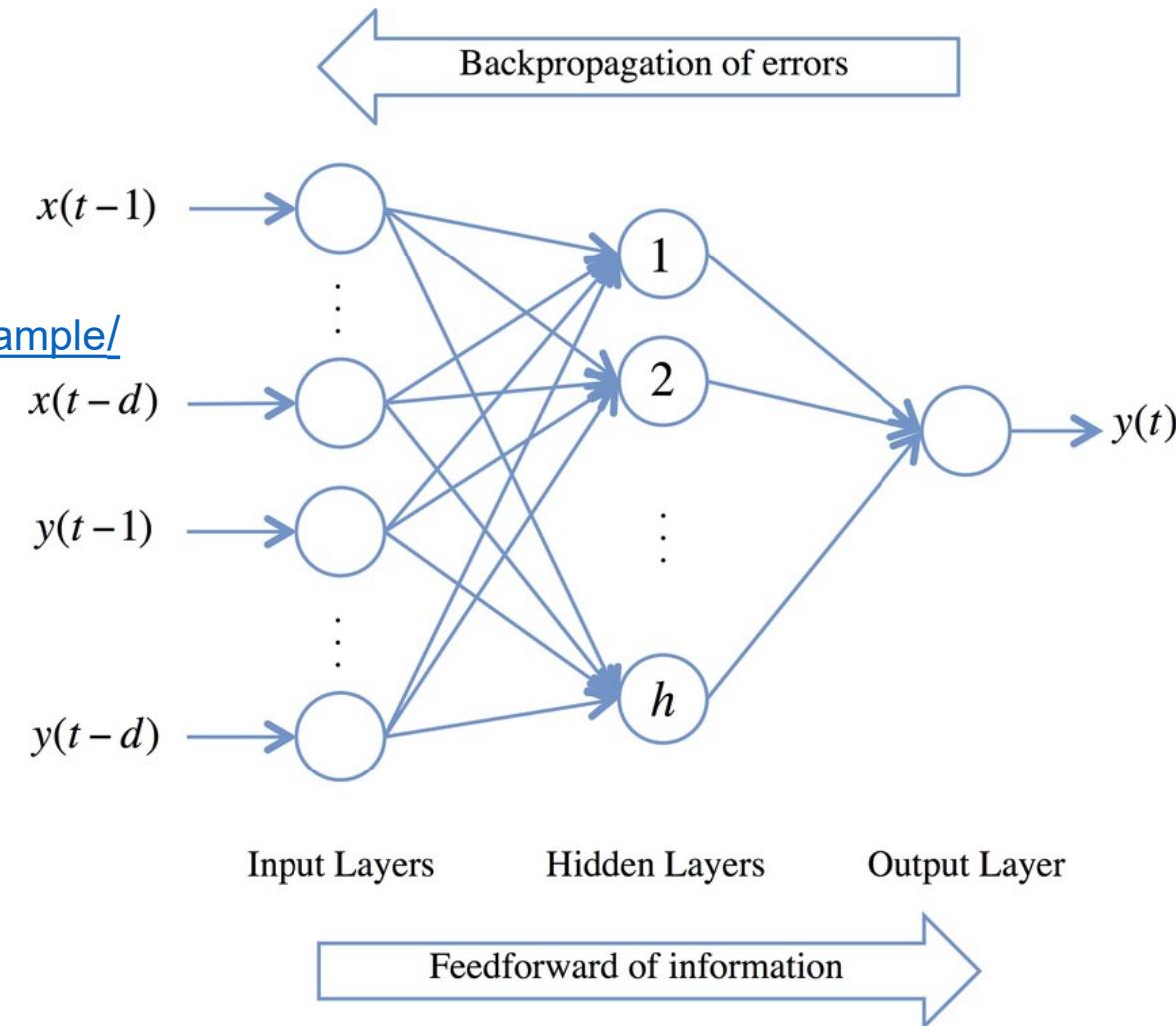
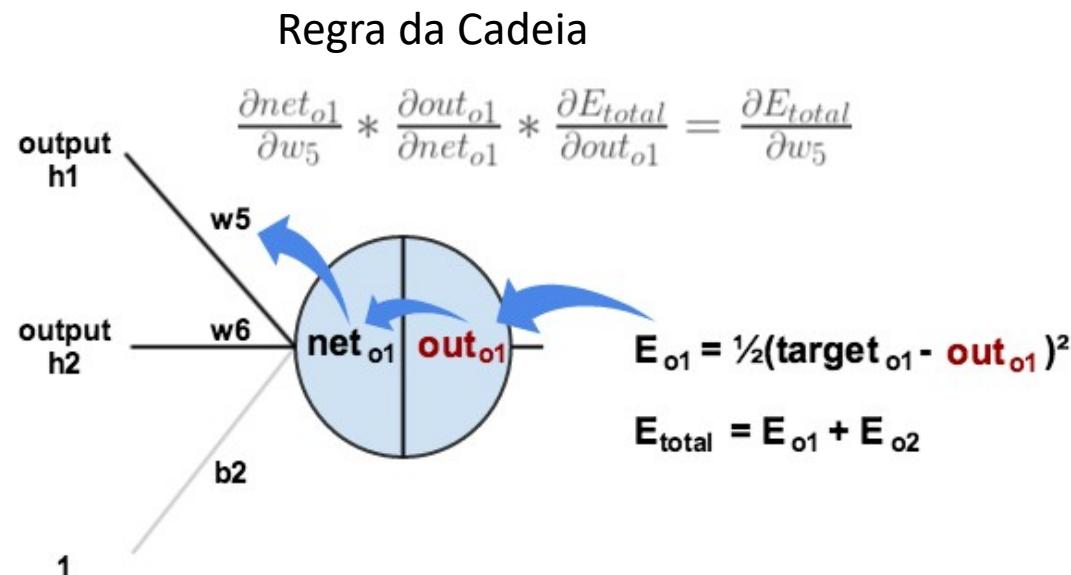
Aprendizado de RNA

Método Clássico: *Feed-forward e Backpropagation*

- Estudar Exemplos nos links a seguir:

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

<https://theneuralblog.com/forward-pass-backpropagation-example/>

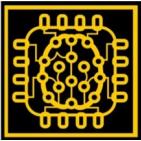




Aprendizado de RNA

Melhorias de Desempenho:

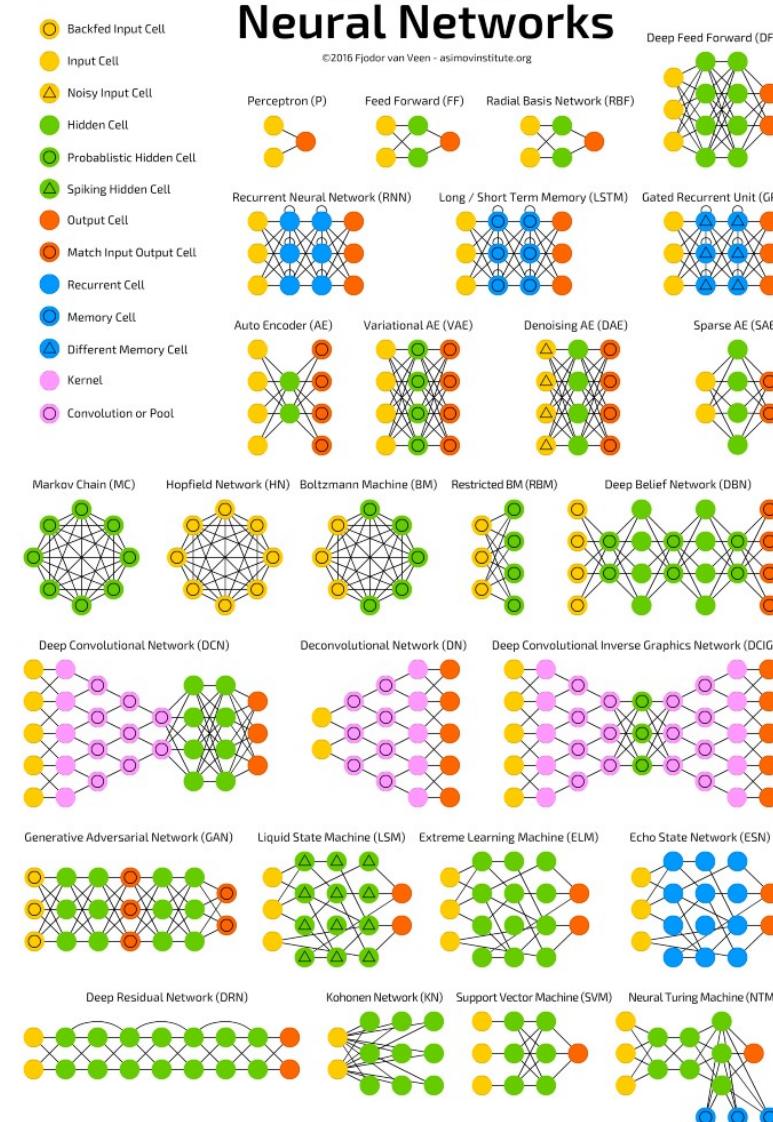
- Atualização sequencial versus batelada;
- Maximizando o conteúdo de informação;
- Função de ativação;
- Valores de saída desejada;
- Normalização das entradas;
- Inicialização;
- Aprendizado a partir de “dicas”;
- Taxas de aprendizado (*learning rate*);
- Algoritmos de Aprendizado mais eficazes



Mapa das RNAs

- Perceptron
- Feed forward
- MLP
- RNN
- CNN
- RBFNN
- DCN
- GAN
- LSTM,...

Tem para todos os gostos.
Escolha a sua!



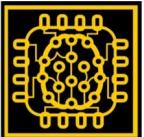
<https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>

Exemplos de RNAs para Aprendizado de Caracteres:

- <https://www.youtube.com/watch?v=RcgfwZ-FFBI>
- https://www.youtube.com/playlist?list=PLZHQBObOWTQDNU6R1_6700Dx_ZCJB-3pi

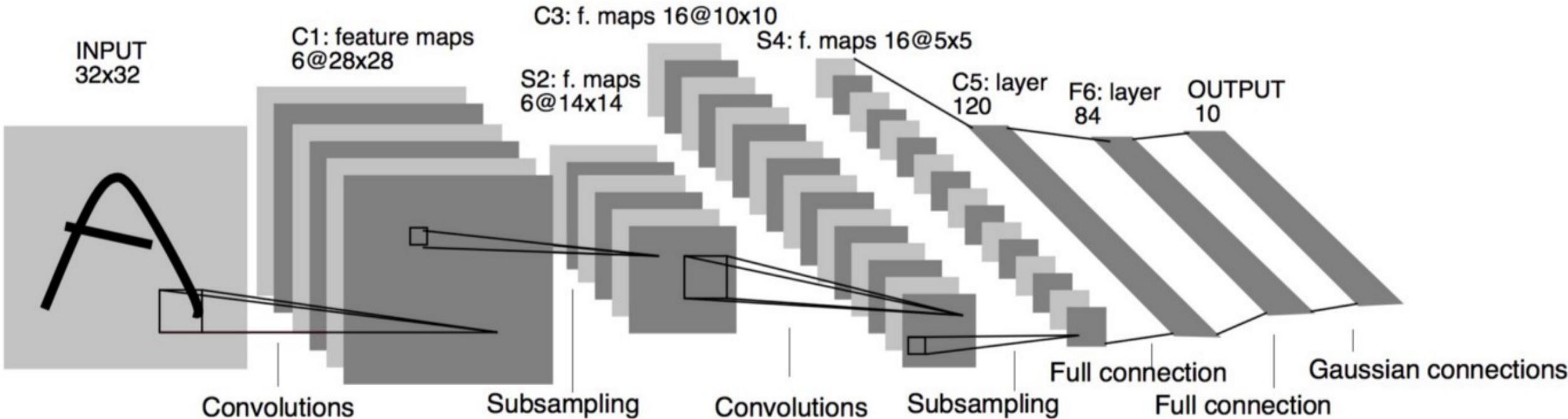
Dr. Prof. Eng. Arnaldo de Carvalho Junior

adecorvalhojr@ifsp.edu.br 38



Redes Neurais Convolucionais

- Redes profundas;
- Utilizadas para o reconhecimento de imagens devido à sua alta eficácia.



<https://medium.com/neuronio-br/entendendo-redes-convolucionais-cnns-d10359f21184>

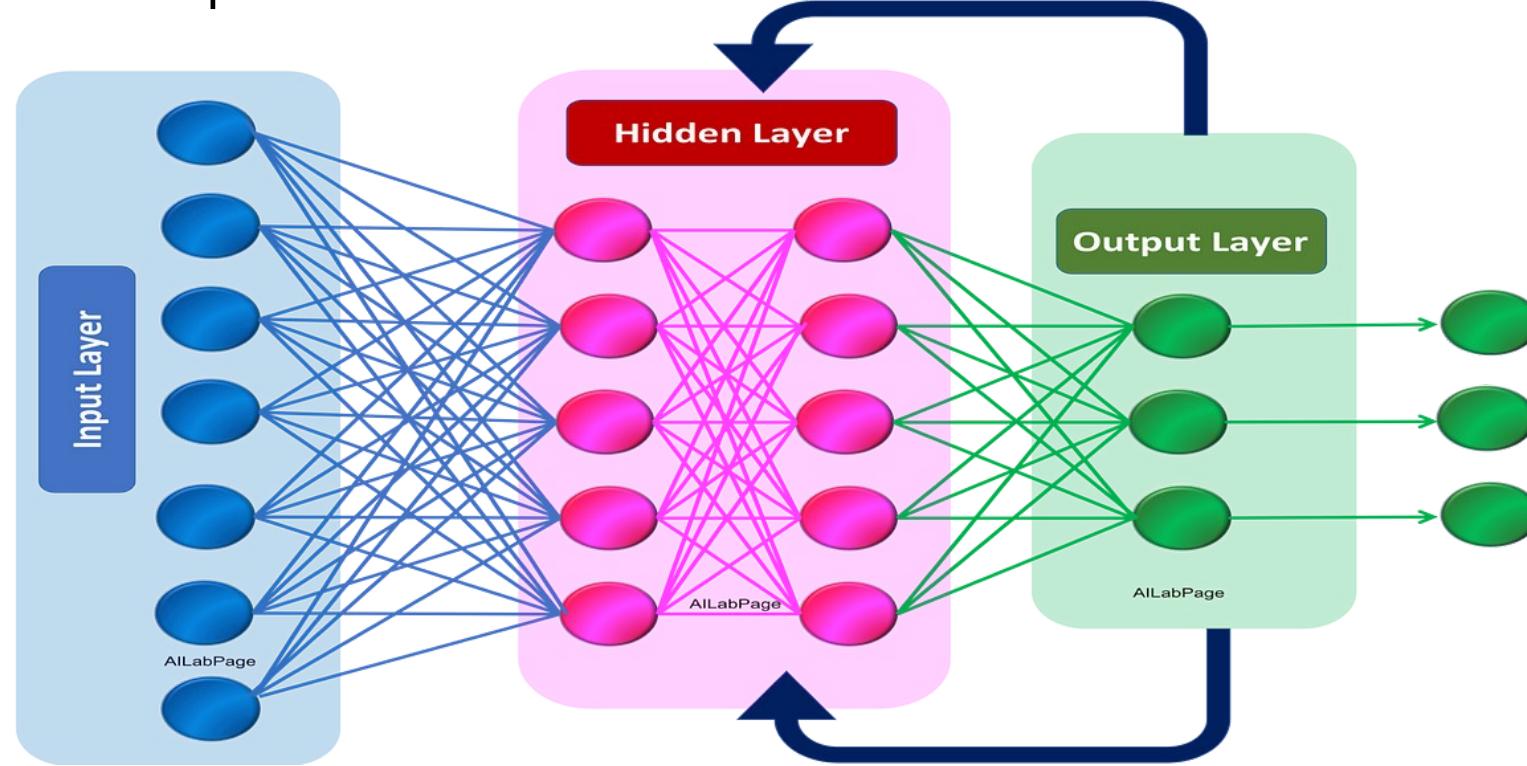
Dr. Prof. Eng. Arnaldo de Carvalho Junior

adecorvalhojr@ifsp.edu.br 39



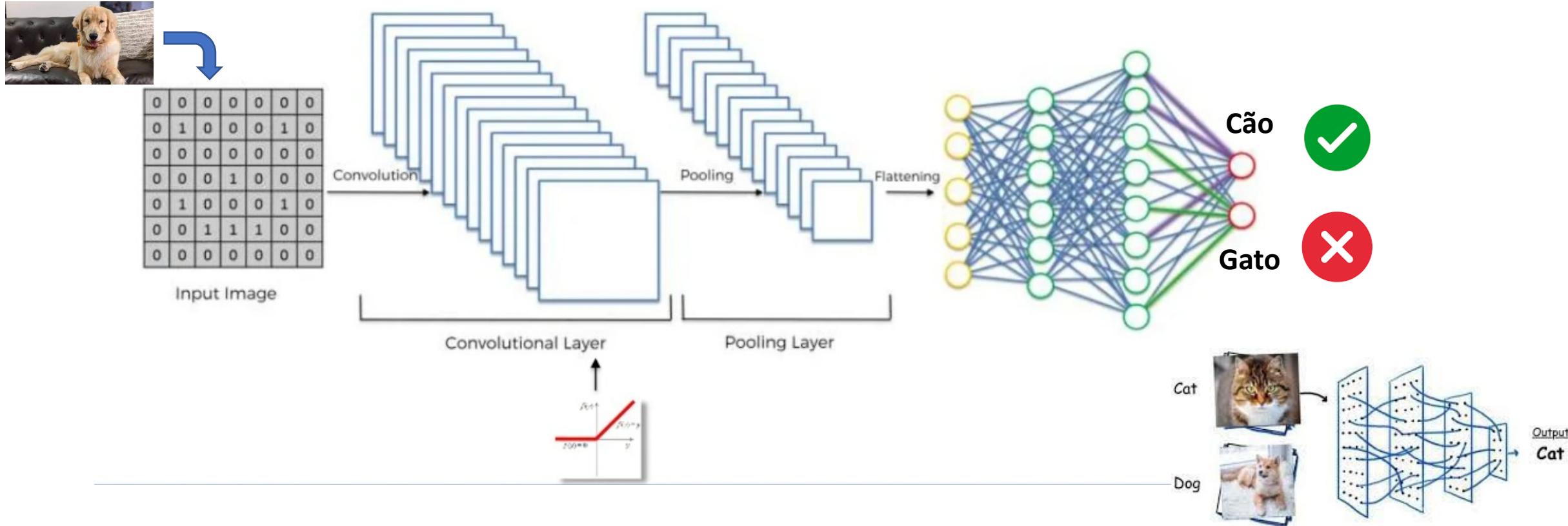
Redes Neurais Recorrentes

- Efeito memória;
- Ideais para séries temporais de dados.





Redes Neurais Convolucionais.

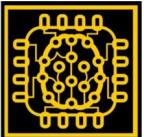


<https://www.slideshare.net/KirillEremenko/deep-learning-az-convolutional-neural-networks-cnn-module-2>

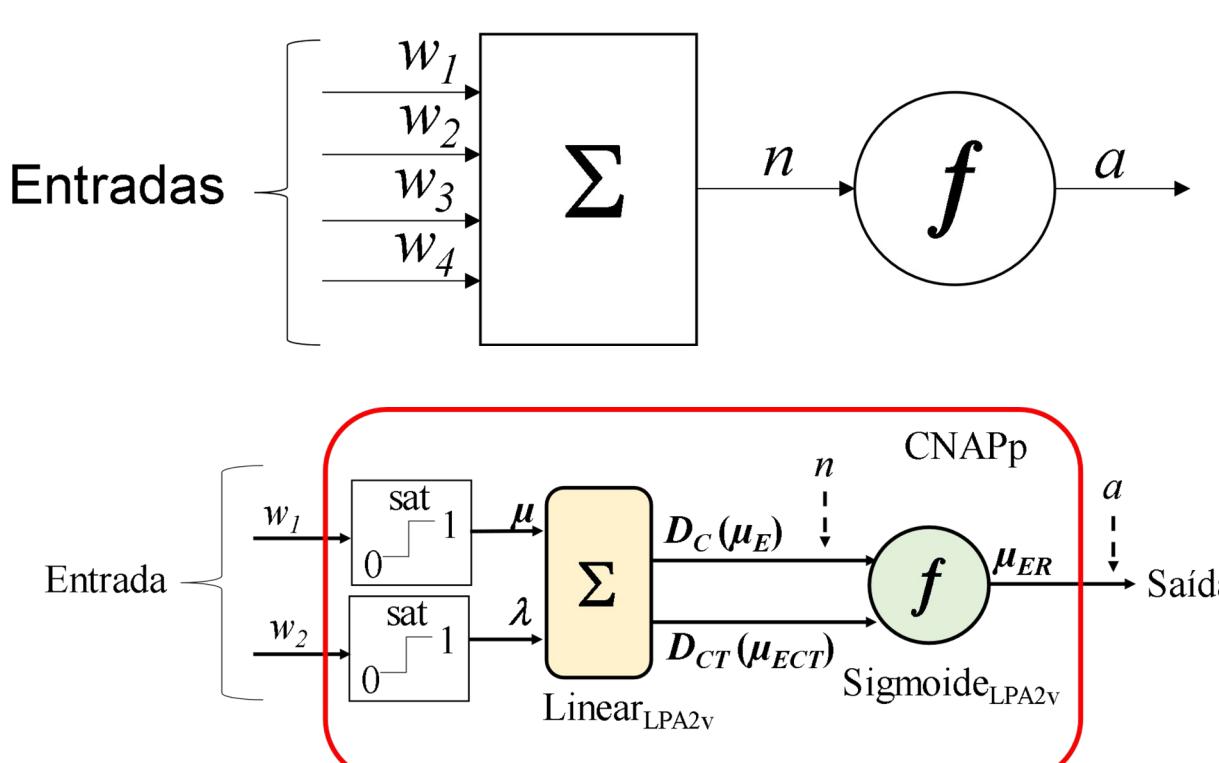
<https://www.superdatascience.com/blogs/the-ultimate-guide-to-convolutional-neural-networks-cnn>

Dr. Prof. Eng. Arnaldo de Carvalho Junior

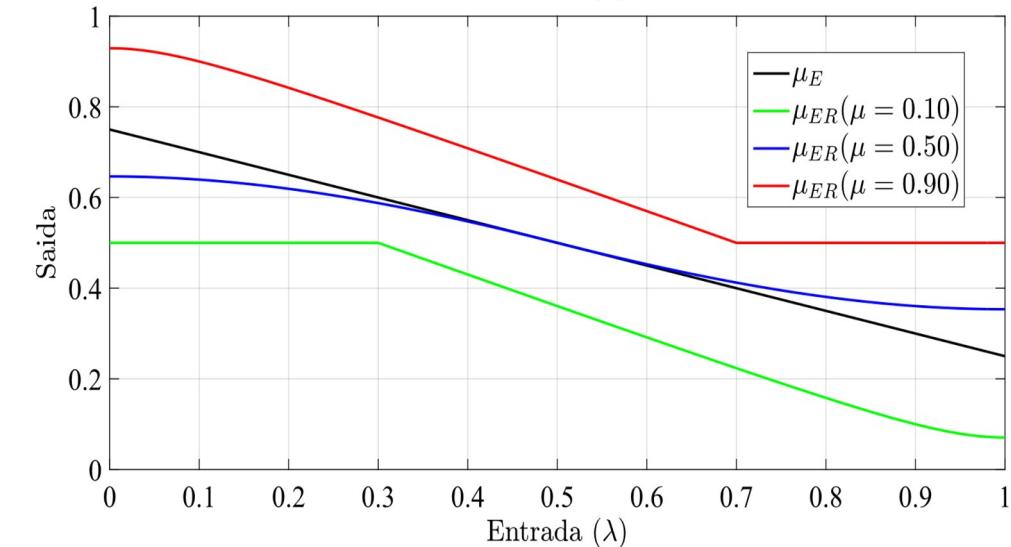
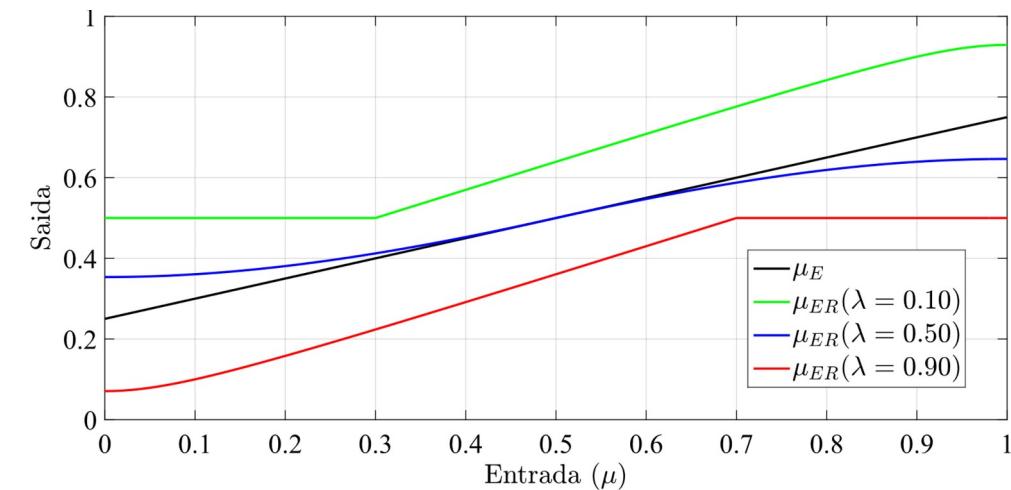
aedcarvalhojr@ifsp.edu.br 41

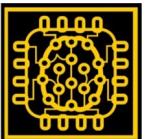


Neurônio Paraconsistente: Função de Ativação LPA2v

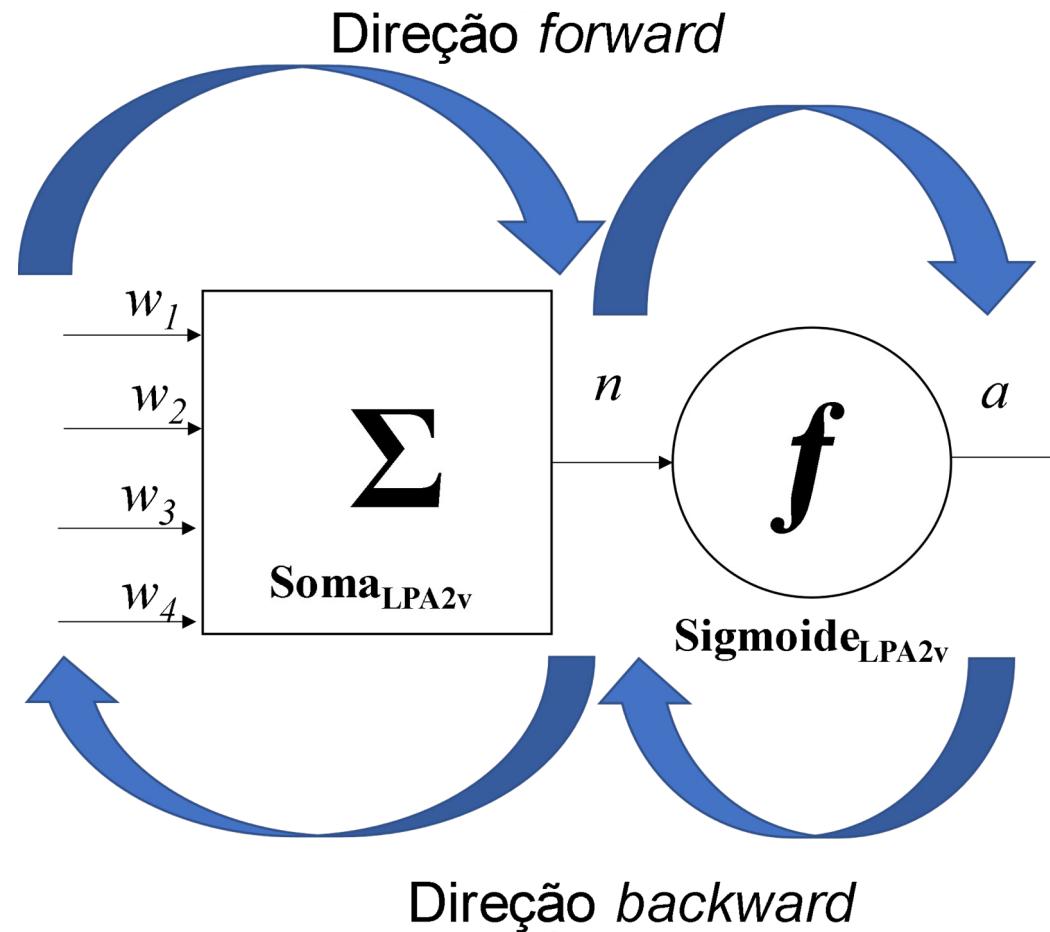


Comparação entre saídas μ_E e μ_{ER} , para diferentes valores fixos de λ (a) e μ (b)



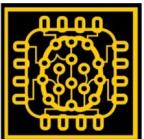


Método *backpropagation* para o neurônio paraconsistente.



$$\frac{\partial \text{Erro}_{total}}{\partial w_1} = \frac{\partial \text{Erro}_{total}}{\partial a} * \frac{\partial a}{\partial n} * \frac{\partial n}{\partial w_1}$$

$$\text{Erro}_{total} = \frac{1}{2} (S_{Target} - S_{PAL})^2$$



Derivadas para CNAPP com saída não-linear (μ_{ER}):

Para $D_c > 0; \mu > \lambda$:

$$\mu_{ER} = \frac{(2 - \sqrt{2\mu^2 + 2\lambda^2 - 4\mu + 2})}{2}$$

$$\frac{\partial \mu_{ER}}{\partial \mu} = \frac{(1 - \mu)}{(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{0.5}}$$

$$\frac{\partial \mu_{ER}}{\partial \lambda} = -\frac{\lambda}{(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{0.5}}$$

Para $D_c < 0; \mu < \lambda$:

$$\mu_{ER} = \frac{(\sqrt{2\mu^2 + 2\lambda^2 - 4\lambda + 2})}{2}$$

$$\frac{\partial \mu_{ER}}{\partial \mu} = \frac{\mu}{(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}}$$

$$\frac{\partial \mu_{ER}}{\partial \lambda} = \frac{\lambda - 1}{(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}}$$

Derivadas para CNAPP com saída linear (μ_E):

$$\frac{\partial \mu_E}{\partial \mu} = 0.5; \quad \frac{\partial \mu_E}{\partial \lambda} = -0.5$$

Para $D_c = 0 \Rightarrow \mu = \lambda$:

$$\mu_{ER} = 0.5 \quad \text{Indefinido (LPA2v)}$$

Derivada pela média

$$\frac{\partial \mu_{ER}}{\partial \mu} = \frac{1}{2(2\mu^2 + 2\lambda^2 - 4\mu + 2)^{0.5}}$$

$$\frac{\partial \mu_{ER}}{\partial \lambda} = -\frac{1}{2(2\mu^2 + 2\lambda^2 - 4\lambda + 2)^{0.5}}$$

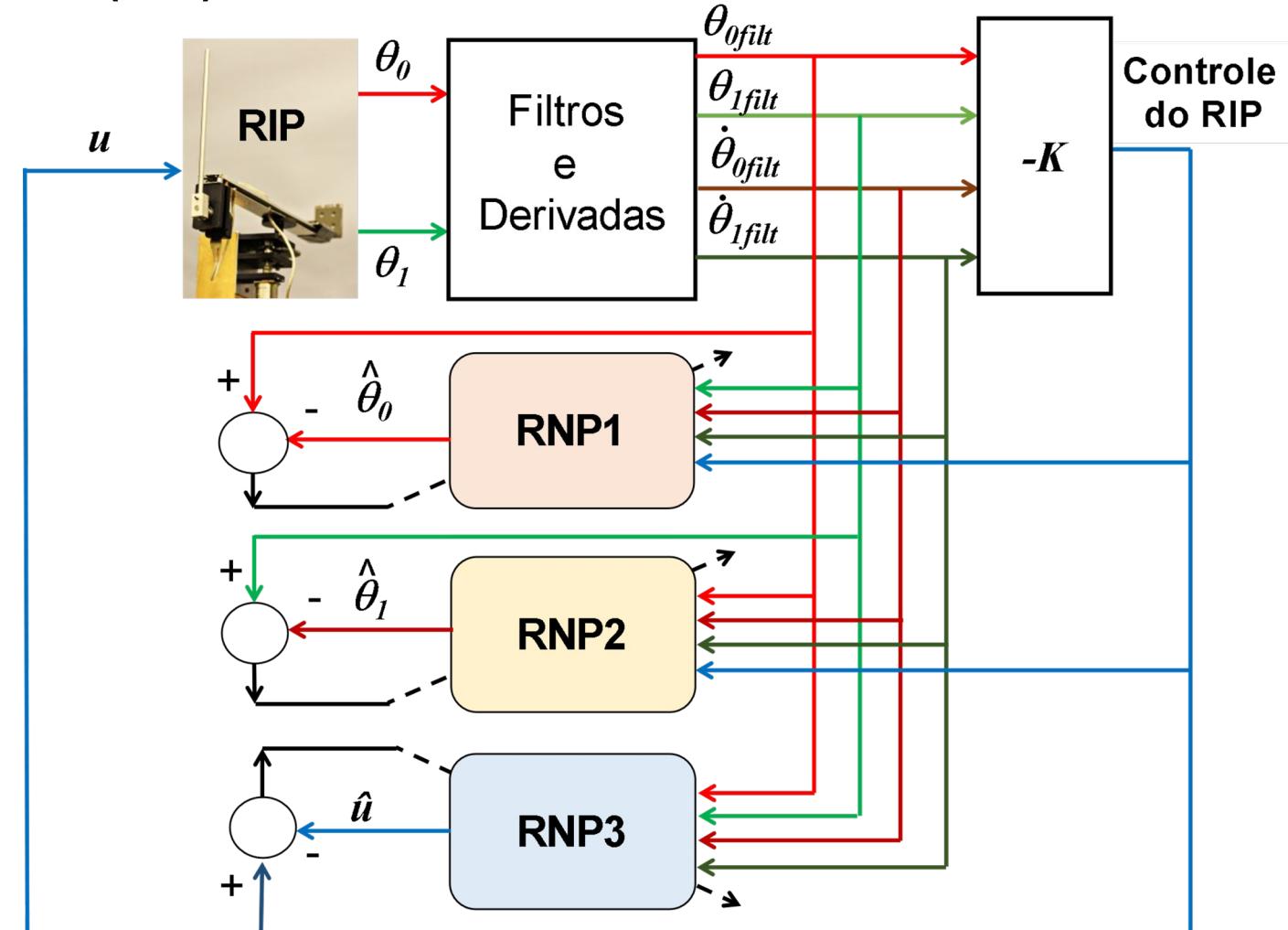
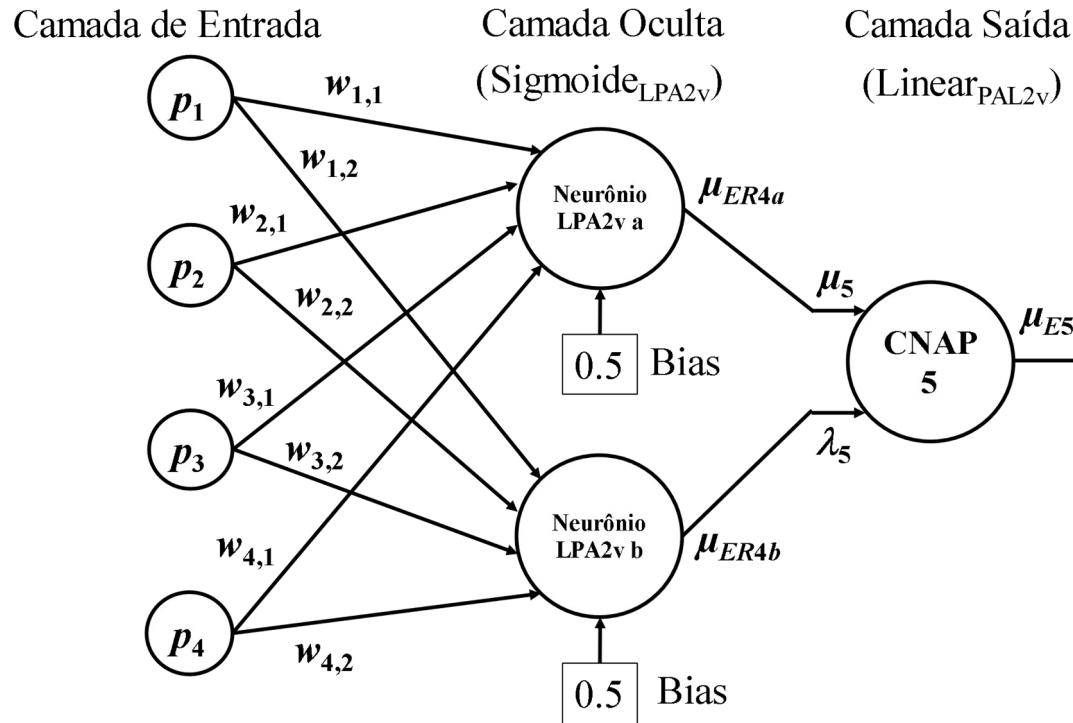
Para $D > 1 \Rightarrow D = 1 \Rightarrow \mu_{ER} = 0.5$

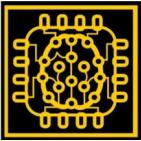
$$\frac{\partial \mu_E}{\partial \mu} = \frac{\partial \mu_E}{\partial \lambda} = 0$$

Derivada arbitrária



Identificação do Pêndulo Invertido Rotativo (RIP)





Rotary Inverted Pendulum Identification for Control by Paraconsistent Neural Network

A. De Carvalho, J. F. Justo, B. A. Angélico, A. M. De Oliveira and J. I. Da Silva Filho, "Rotary Inverted Pendulum Identification for Control by Paraconsistent Neural Network," in *IEEE Access*, doi: 10.1109/ACCESS.2021.3080176.

<https://ieeexplore.ieee.org/document/9430548>

Received April 21, 2021, accepted May 8, 2021, date of publication May 13, 2021, date of current version May 25, 2021.
Digital Object Identifier 10.1109/ACCESS.2021.3080176

Rotary Inverted Pendulum Identification for Control by Paraconsistent Neural Network

ARNALDO DE CARVALHO, JR.^{1,2}, JOÃO FRANCISCO JUSTO²,
BRUNO AUGUSTO ANGÉLICO², ALEXANDRE MANICOBÁ DE OLIVEIRA³, (Member, IEEE),
AND JOÃO INÁCIO DA SILVA FILHO³, (Member, IEEE)

¹Federal University of Paraná, School of Technology, São Paulo, Brazil 11533-160, Brazil

²Faculdade Politécnica, Universidade de São Paulo, São Paulo 05508-010, Brazil

³Laboratory of Applied Paraconsistent Logic, Department of Electronic Engineering and Computation, Santa Cecília University (UNISANTA), Santos 11045-907, Brazil

Corresponding author: Arnaldo de Carvalho, Jr. (adecarvalhojr@ifsp.edu.br)

ABSTRACT Artificial neural networks (ANNs) have been used over the last few decades to perform tasks by learning with comparisons. Fitting input-output models, system identification, control, and pattern recognition are some fields for ANN applications. However, problems involving uncertain situations could be challenging for them. The family of paraconsistent logics (PL) is a powerful tool that can deal with uncertainty and contradictory information, so getting attention from researchers for its implications and applications in artificial intelligence. This investigation describes a novel activation function reasoned on the paraconsistent annotated logic by two-value annotations (PAL2v) rules, a variation of PL, allowing the design of a new paraconsistent neural net (PNN), applied in model identification for control (I4C) of a closed-loop rotary inverted pendulum (RIP) system.

INDEX TERMS Paraconsistent logic, neural net, model identification, pattern analysis, rotary inverted pendulum.

I. INTRODUCTION

Artificial neural networks are mathematical algorithms that can learn a specific function or pattern [1], [2], could be applied in a wide range of applications [3]. Because of those characteristics, the ANN is a powerful tool for identification and system control [1], [2]. In recent years, the ANN application for identification and control has been investigated, for online and offline environments [4], [5]. The model identification approach oriented to control a system, when the model is valid only in the boundary conditions of this purpose, is called identification for control (I4C) [6]–[8].

Noise, disturbances, and uncertainties are inherent aspects of real-world problem description [9], not usually considered by the state estimation methods for nonlinear control techniques [10]. The ANN attractiveness in identification and control derives from its intrinsic ability to use experimental data to model unknown systems, even with perturbations or uncertainties, enabling a notable alternative in situations

where conventional methods could fail in determining the appropriate control laws [11].

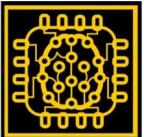
Uncertainty, inconsistency, and contradictory signals are also aspects considered by the paraconsistent logic (PL), which deals with them without resulting in triviality [12], [13]. PL is a family of alternative logics that repeats the principle of non-contradiction of classical logic. PL attracted interest not only philosophical but by its implications and applications in artificial intelligence too [14]. Through the use of pieces of evidence, the PL can model human knowledge and reasoning, allowing applications in specialist systems that consider uncertain information in decision-making [13], [15], [16].

The Paraconsistent Annotated Logic with 2-value annotations (PAL2v), belonging to the PL family, uses two variables, or evidence, which allows greater representation power in a lattice of the Real plane, when expressing knowledge about a proposition P [15], [17]. PAL2v allows an expert system to deal with concepts like uncertain, inconsistent, and incomplete data, such as those obtained from sensors, without the risk of trivialization [17], [18]. A hybrid proportional-integral (PI) control approach, using PAL2v cells to handle the

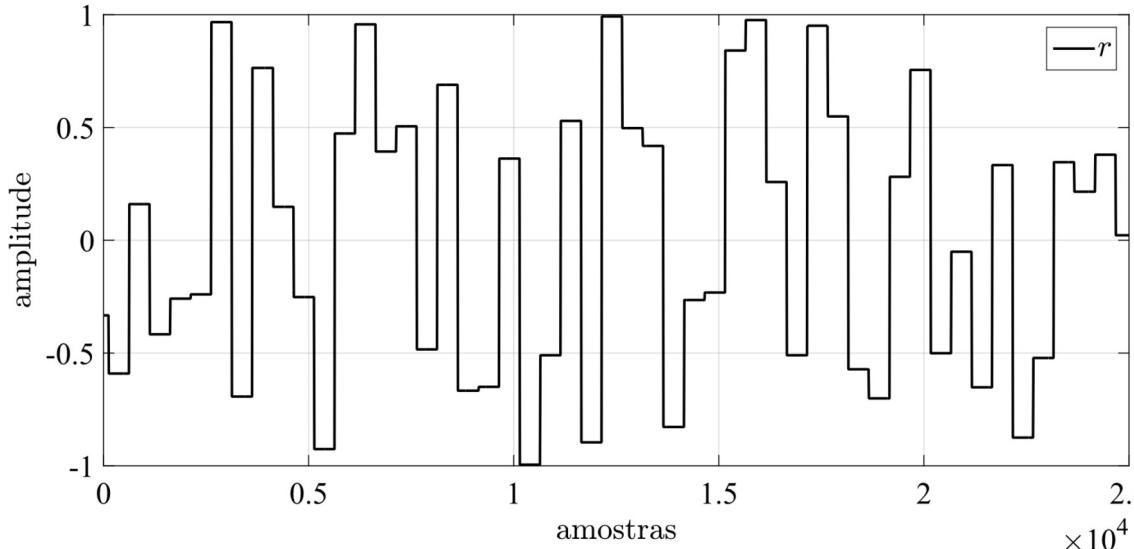
The associate editor coordinating the review of this manuscript and approving it for publication was Shen Yin.

VOLUME 9, 2021 This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see <https://creativecommons.org/licenses/by/4.0/>

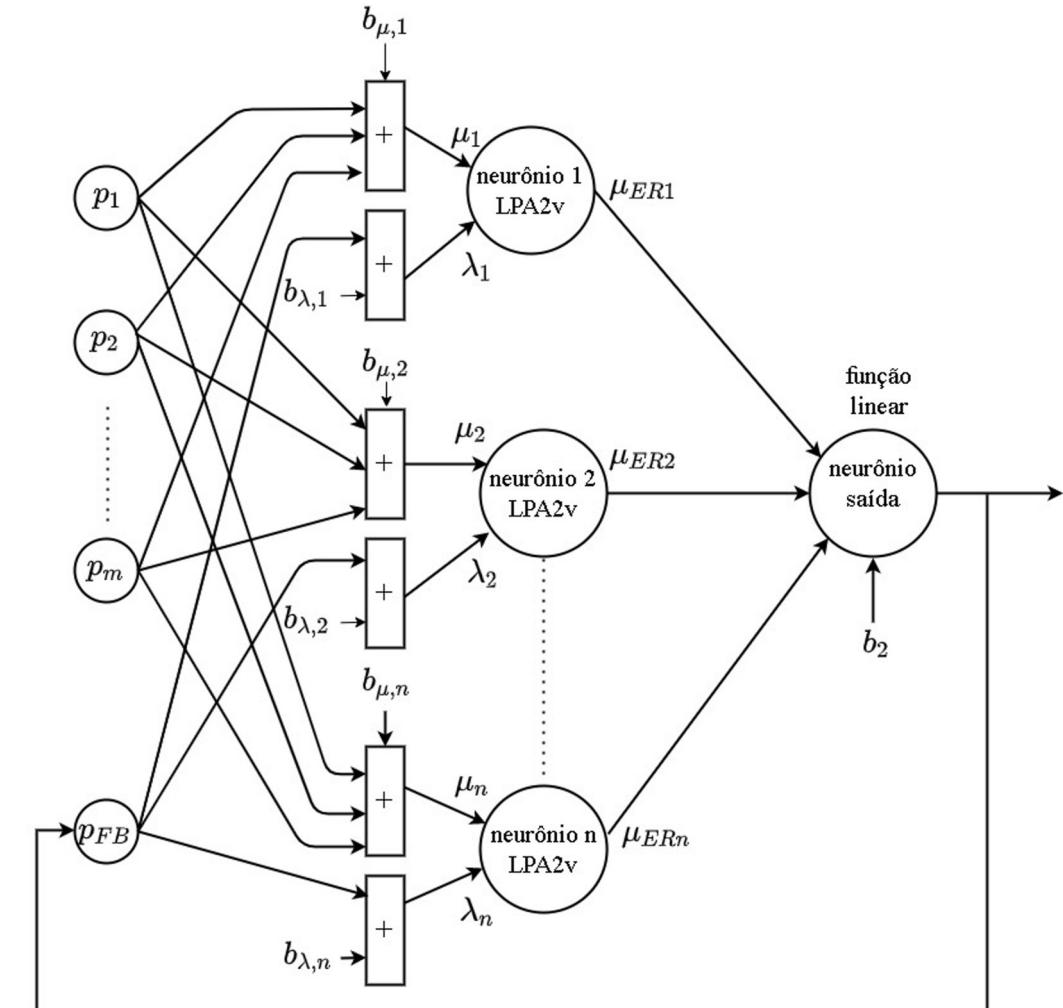
74155

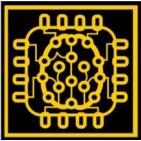


Sinal pseudoaleatório de referência (r) para a identificação do braço e pêndulo do RIP.

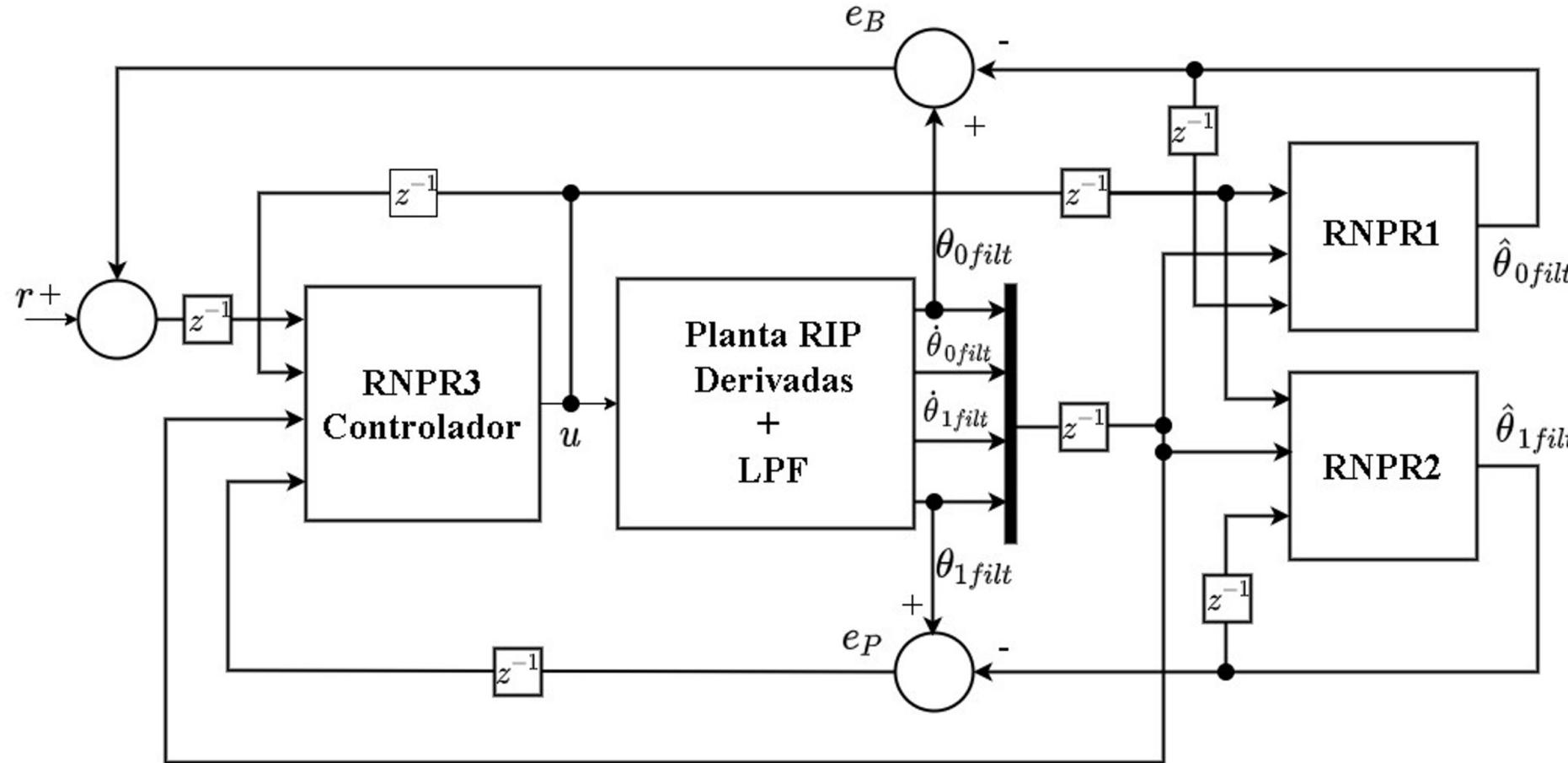


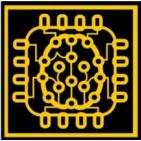
Rede Neural Paraconsistente Recorrente (RNPR).





Controle do RIP por Rede Neural Paraconsistente Recorrente.





Model reference control by recurrent neural network built with paraconsistent neurons for trajectory tracking of a rotary inverted pendulum

Carvalho, A., Justo, J.F., Angélico, B.A. et al. Model reference control by recurrent neural network built with paraconsistent neurons for trajectory tracking of a rotary inverted pendulum, *Applied Soft Computing*, 2022, 109927, ISSN 1568-4946, DOI:10.1016/j.asoc.2022.109927.

<https://www.sciencedirect.com/science/article/abs/pii/S1568494622009760?via%3Dihub>

Applied Soft Computing 133 (2023) 109927



Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc



Model reference control by recurrent neural network built with paraconsistent neurons for trajectory tracking of a rotary inverted pendulum

Arnaldo de Carvalho Junior^{a,*}, Bruno Augusto Angelico^b, João Francisco Justo^b,

Alexandre Maniquiba de Oliveira^a, João Inacio da Silva Filho^c

^a Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP), São Paulo SP, Brazil

^b Escola Politécnica, Universidade de São Paulo, São Paulo SP, Brazil

^c Santa Cecília University (UNISANTA), Santos SP, Brazil

ARTICLE INFO

Article history:

Received 30 March 2022

Received in revised form 7 November 2022

Accepted 4 December 2022

Available online 13 December 2022

Keywords:

Paraconsistent logic

Neural models

Recurrent networks

Paraconsistent neural networks

Nonlinear control systems

ABSTRACT

This investigation presents a recurrent paraconsistent neural network (RPNN), as the main element of the model reference control (MRC) strategy for the rotary inverted pendulum (RIP). The RIP characteristics, such as nonlinearity, two degrees-of-freedom (2DoF) motion, and under-actuated system, make it an ideal design for applying and testing MRC. The RPNN is composed of three controllers: the reference controller (MRC) to use RPNNs; two of them to model the arm and pendulum angles; and the third one to control the system while tracking a reference trajectory. The hidden neurons of the RPNN use the paraconsistent annotated logic by 2value annotations (PAL2v) rules as an activation function. PAL2v, as a member of the paraconsistent logics family, deals with uncertain and contradictory data, representing a potentially robust alternative to applications of artificial neural networks in control. The PAL2v neuron is detailed and compared with other activation functions in recurrent neural networks (RNN). With real-time experiments, the PNMRC strategy is compared with classical control methodology, presenting excellent performance.

© 2022 Elsevier B.V. All rights reserved.

1. Introduction

Several features of the modern control theory can be investigated using complex physical systems, such as the rotary inverted pendulum (RIP), also labeled Furuta pendulum [1]. The RIP is appropriate for such a task due to its intrinsic characteristics, such as nonlinearity, single input multiple output (SIMO) variables, fast dynamics, and two-degree-of-freedom (2DoF) motion [2–4]. Inverted Pendulums are the basis of technologies such as missiles, aircraft, satellite control, and walking robotics [3].

Among the various control engineering research, soft computing methods (SCM) are receiving much attention. SCM is in the state-of-the-art of the control of nonlinear systems and includes the application of knowledge-based systems, genetic algorithms (GA), machine learning, fuzzy logic, artificial neural network (ANN), or a combination of them [5].

ANN computing models can recognize patterns and learn functions, finding a wide range of applications [6–8]. Features, such as self-learning, self-organizing, nonlinearity, input-to-output mapping advances, and fault tolerance, make the ANN an ideal resource for system identification and modeling [8]. In recent years, several investigations have explored the ANN capabilities in offline and online environments to model and control a certain system [9–12]. The capability to model unknown systems, even under disturbances or uncertainties, makes the ANN an excellent alternative in finding the proper control law where traditional methods could fail [11]. Manifold control strategies based on models implement ANNs, including observed-based, internal model control (IMC), model predictive control (MPC), and model reference control (MRC) [6,11,13,14].

ANN has observed accelerating progress, with diverse topologies proposed in the literature, such as the recurrent neural network (RNN) [8]. RNNs, which use temporal or sequential data to model and recognize patterns, are good candidates to model a nonlinear system, despite the difficulties of training [14]. One SCM-based control strategy is to implement the MRC with RNN. In this case, one RNN models the plant while the other one learns the controller [14].

* Corresponding author.
E-mail address: aedecarvalhojr@ifsp.edu.br (A. de Carvalho Junior).

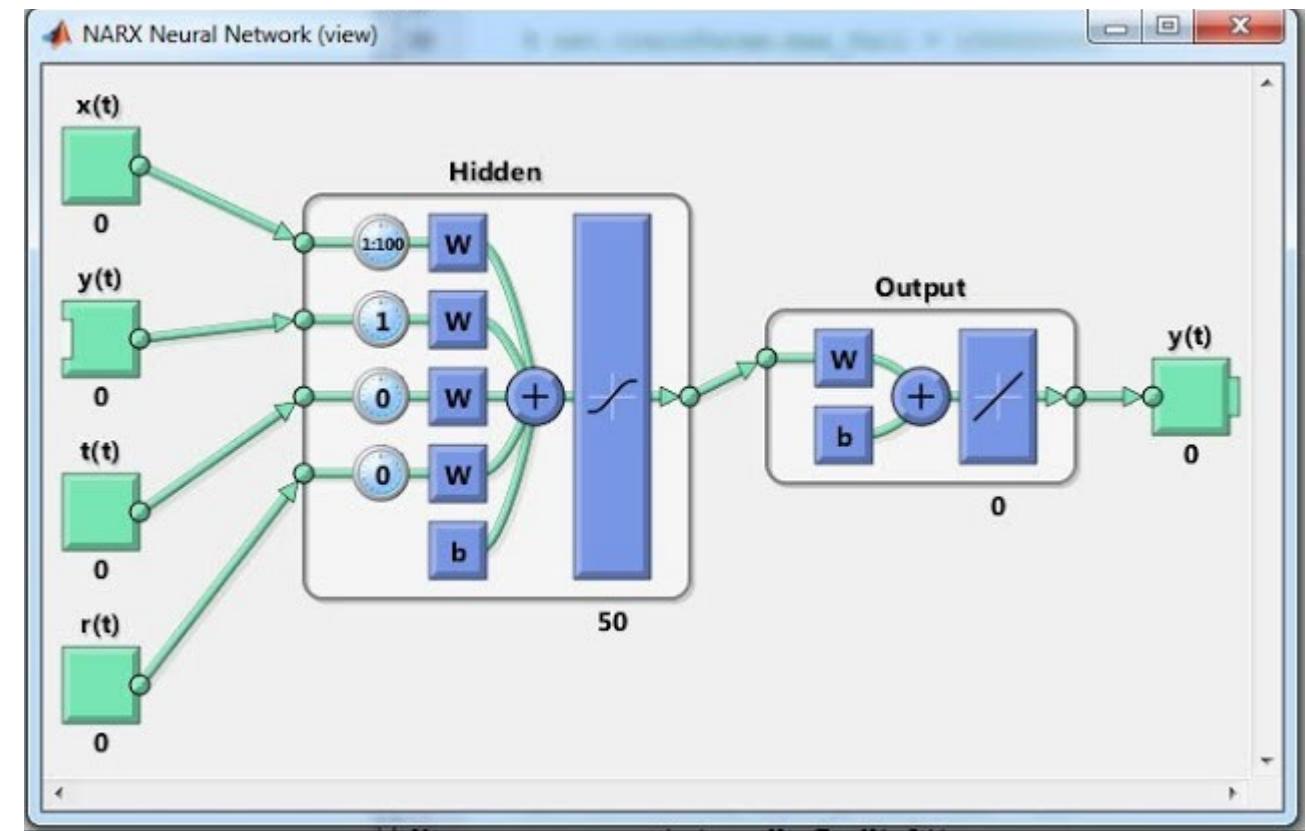
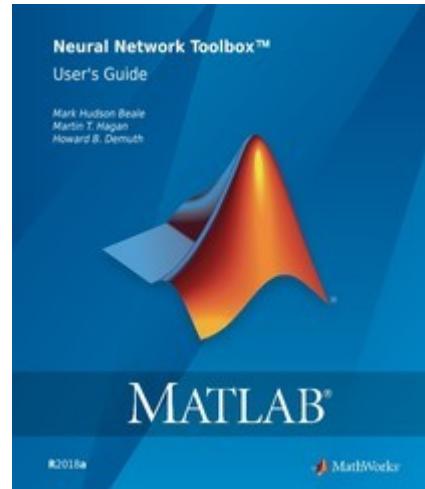
<https://doi.org/10.1016/j.asoc.2022.109927>
1568-4946/© 2022 Elsevier B.V. All rights reserved.

Dr. Prof. Eng. Arnaldo de Carvalho Junior

aedecarvalhojr@ifsp.edu.br 49



Simulações em Matlab



http://cda.psych.uiuc.edu/matlab_pdf/nnet.pdf



Exercício 1

a. Utilize o script Matlab disponível a seguir:

<https://www.mathworks.com/matlabcentral/fileexchange/130739-paraconsistent-neural-network-pnn>

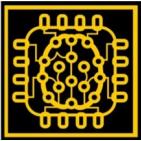
b. Analise o script, com especial atenção para o cálculo dos pesos e treinamento das RNAs utilizando diferentes funções de ativação (σ , tanh, ReLU, L-ReLU e PAL2v).

c. Execute o script para diferentes valores de entrada e de saída, conforme tabela verdade XOR e valores analógicos e analise os resultados obtidos para cada RNA.

a	b	s
0	0	0
0	1	1
1	0	1
1	1	0

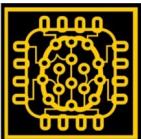
a	b	s
0.2	0.8	0.5
0.1	0.3	0.7
0.6	0.3	0.8
0.7	0.8	0.3

d. Teste também para diferentes valores de taxa de aprendizagem: $lr = 1; 0.5; 0.1; \dots$



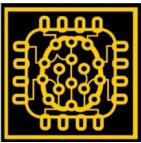
Exercício 2

- a. Abra a ferramenta nftool do Matlab.
- b. Carregue o exemplo “*Simple Fitting Problem*”
- c. Selecione as % de treinamento, validação e teste (70-15-15), (60-20-20) e (50-25-25)
- d. Execute primeiramente a rede com 2 neurônios na camada oculta. Analise os resultados tanto numéricos como gráficos.
- e. Aumente para 4 neurônios na camada oculta e refaça o treinamento da rede. O que aconteceu?
- f. Aumente para 10 neurônios e refaça o treinamento. O que aconteceu?
- g. Refaça novamente o exercício utilizando diferentes algoritmos de treinamento. Compare os resultados.



Exercício 3

- a. Crie uma base de dados de 100 valores de L e C fictícia para cálculo de Frequência de Ressonância de circuito LC .
 - I. Exemplo: 10 a 1000pF e 10 a 1000nH ; $F_r = \frac{1}{2\pi\sqrt{LC}}$
- b. Use essa base de dados para treinar uma rede neural capaz de calcular a frequência de ressonância.
- c. Utilize o aplicativo do Matlab para criar a rede neural e utilizar o script para quaisquer valores de L e C dentro da faixa acima indicada.
- d. Compare os resultados fornecidos pela saída da RNA e do cálculo real de frequência de ressonância.

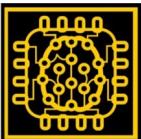


Redes Neurais Artificiais

- COSTA, L. “*Redes Neurais Artificiais: criando um Perceptron de uma camada*”, BLOG, acessado em 27/04/2022, disponível em: <https://growiz.com.br/redes-neurais-artificiais-criando-um-perceptron-de-uma-camada-em-c/>
- HAGAN, M. T.; DEMUTH, H. B.; BEALE, M. H.. “Neural Network Design”, Martin Hagan, 2º edition, 2014, 802 p. Disponível em: <https://hagan.okstate.edu/NNDesign.pdf>
- HAGAN, M. T.; DEMUTH, H. B.; DE JESÚS, O. “*An introduction to the use of neural networks in control systems*”. International Journal of Robust and Nonlinear Control, Vol.12, Issue 11, 2002, p. 959-985.
- APICELLA, A.; DONNARUMMA, F.; ISGRÒ, F.; Prevete, R. “*A survey on modern trainable activation functions*”, Neural Networks, Volume 138, p. 14-32, 2021.
- SONODA, S.; MURATA, N. “*Neural network with unbounded activation functions is universal approximator.*” Applied and Computational Harmonic Analysis, Vol. 43, Issue 2, 2017, p. 233-268.
- WIKIPEDIA, “*Radial Basis Function*”, acessado em 27/04/2022, disponível em: https://en.wikipedia.org/wiki/Radial_basis_function.
- NIELSEN, M. “*Neural Networks and Deep Learning*”, eBOOK online, 2019. Acessado em 27/04/2022, disponível em: <http://neuralnetworksanddeeplearning.com/index.html>.
- FACURE, M. “*Funções de Ativação: Entendendo a importância da ativação correta nas redes neurais*”, BLOG, acessado em 27/04/2022, disponível em: <https://matheusfacure.github.io/2017/07/12/activ-func/>
- FACURE, M. “*Redes Neurais Feedforward Densas*”, BLOG, acessado em 27/04/2022, disponível em: <https://matheusfacure.github.io/2017/05/15/deep-ff-ann/>



- MAZUR, M. “**A Step by Step Backpropagation Example**”, BLOG, acessado em 27/04/2022, disponível em: <https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/>
- TCH, A. “**The mostly complete chart of Neural Networks, explained**”, BLOG, acessado em 27/04/2022, disponível em: <https://towardsdatascience.com/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464>
- ABIODUN, O. I.; et al. “**State-of-the-art in artificial neural network applications: A survey**”, Heliyon, Vol. 4, Issue 11, 2018, 41 p.
- DE CARVALHO, A.; et al. “**Rotary Inverted Pendulum Identification for Control by Paraconsistent Neural Network**,” in IEEE Access, doi: 10.1109/ACCESS.2021.3080176.
- DE CARVALHO JUNIOR, A. “**Identification and Control of Dynamic Systems with Paraconsistent Neural Network**”. 2021. 196 p. Tese (Doutorado) – Programa de Engenharia Elétrica, Escola Politécnica, Universidade de São Paulo, São Paulo, 2021. DOI: 10.11606/T.3.2021.tde-08102021-100149. Disponível em: <https://www.teses.usp.br/teses/disponiveis/3/3142/tde-08102021-100149/pt-br.php>
- DEMUTH, H. and BEALE, M.” **Neural Network Toolbox: For Use with MATLAB®**”. Free Online Ebook, 846 p. Acessado em 27/04/2022. Disponível em: http://cda.psych.uiuc.edu/matlab_pdf/nnet.pdf



- ROSA, J. L. G. R. Artificial Neural Networks – Models and Applications, In-Tech, 416 p., 2016. ISBN-13: 978-9535127055. Disponível em: https://mts.intechopen.com/storage/books/5191/authors_book/authors_book.pdf
- Kerschbaumer, R. Redes Neurais - Tópicos m Inteligência Artificial, Instituto Federal Catarinense, Campus Luzerna. Disponível em: <https://ricardokers.github.io/Anexos/Slides%20Redes%20Neurais.pdf>. Acessado em Maio 10, 2024.
- Carvalho, A. Redes Neurais Artificiais: Algoritmos poderosos para aplicações de IA e ML. EAILab, Fevereiro 2024. Disponível em: <https://eailab.labmax.org/2024/04/03/redes-neurais-artificiais-algoritmos-poderosos-para-aplicacoes-de-ia-e-ml/>. Acessado em Maio 13, 2024.
- Carvalho, A. Função de Ativação, o Núcleo da Composição de Neurônios Artificiais. EAILab, Abril 2024. Disponível em: <https://eailab.labmax.org/2024/02/28/funcao-de-ativacao-o-nucleo-da-composicao-de-neuronios-artificiais/>. Acessado em Maio 13, 2024.
- DE CARVALHO JUNIOR, A. Identification and Control of Dynamic Systems with Paraconsistent Neural Network. 2021. 196 p. Tese (Doutorado) – Programa de Engenharia Elétrica, Escola Politécnica, Universidade de São Paulo, São Paulo, 2021. Disponível em: <https://doi.org/10.11606/T.3.2021.tde-08102021-100149> Acessado em Maio 13, 2024.
- DSA, Função de Ativação, Deep Learning Book, Data Science Academy. Disponível em <https://www.deeplearningbook.com.br/funcao-de-ativacao>, acessado em fevereiro 27, 2024.



Redes Neurais Artificiais



Perguntas?

