
Aula 07:

Computação Evolutiva e Conexionista – Evoluindo RNA's

Prof. Hugo Puertas de Araújo
hugo.puertas@ufabc.edu.br
Sala: 509.2 (5º andar / Torre 2)

■ Agenda

■ Redes Neurais Artificiais

- ❖ Modelamento matemático
- ❖ Aprendizado
- ❖ Usos

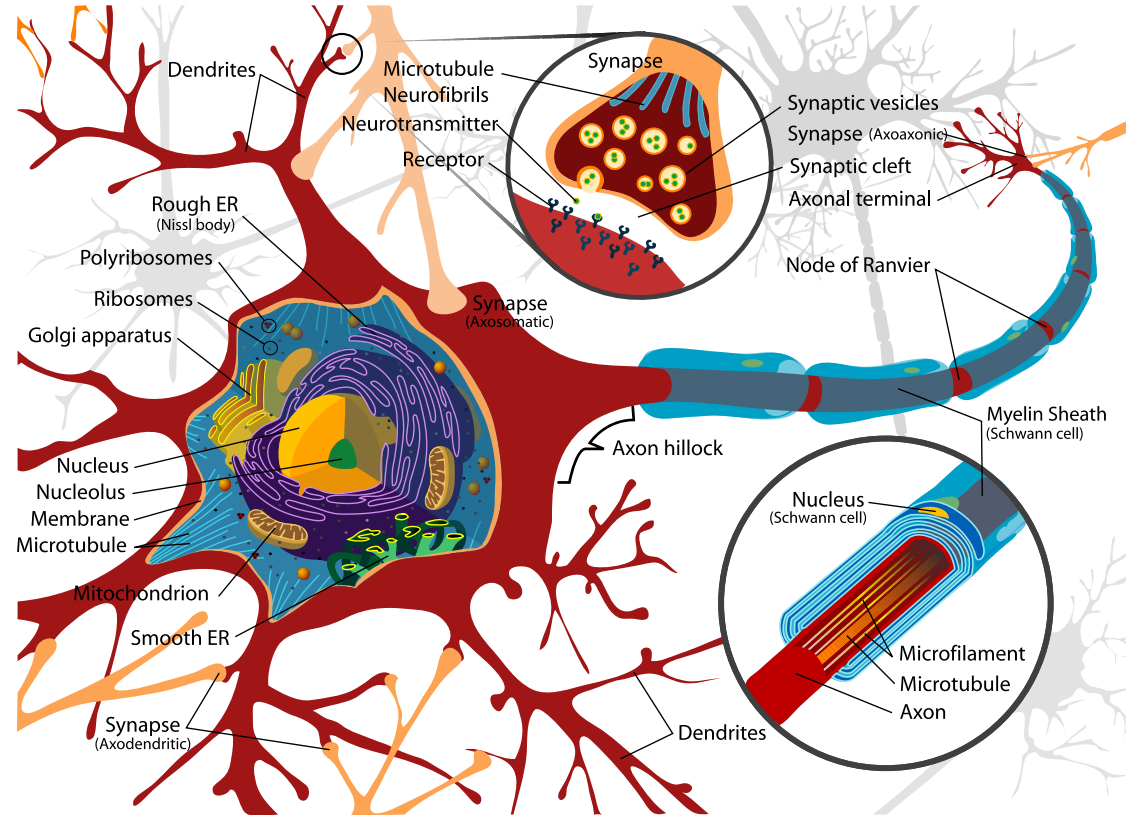
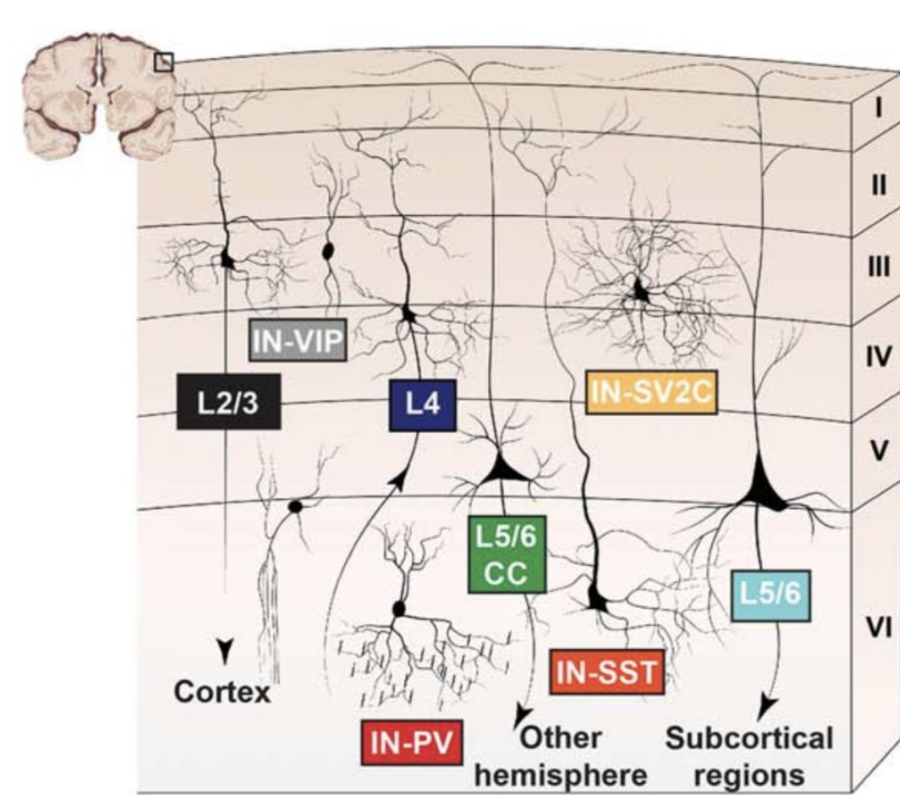
■ Algoritmos Genéticos + RNA's

- ❖ Evolução x Treinamento
- ❖ Representação cromossomial
- ❖ Neuroevolução

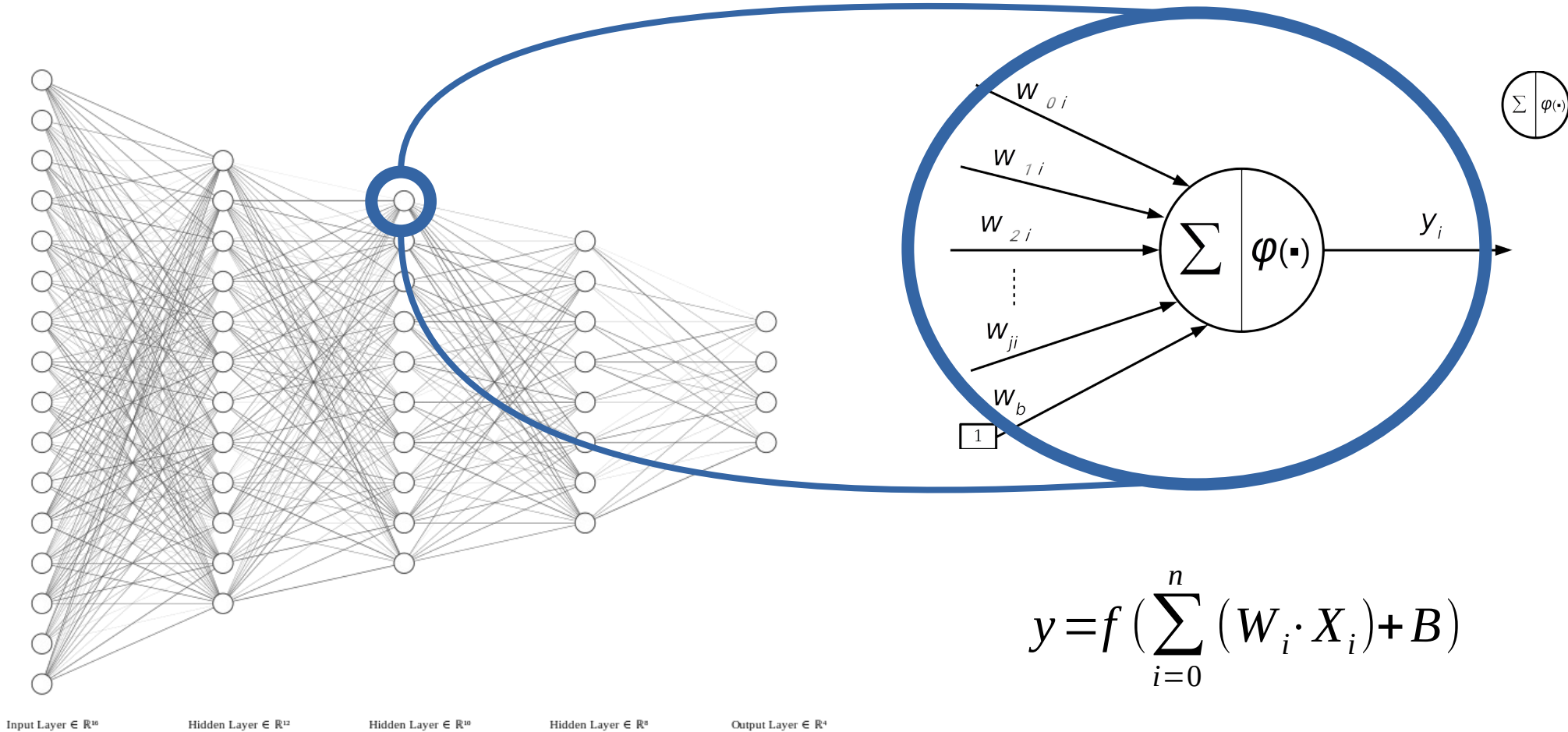
Computação Evolutiva e Conexionista



Redes Neurais – Inspiração biológica

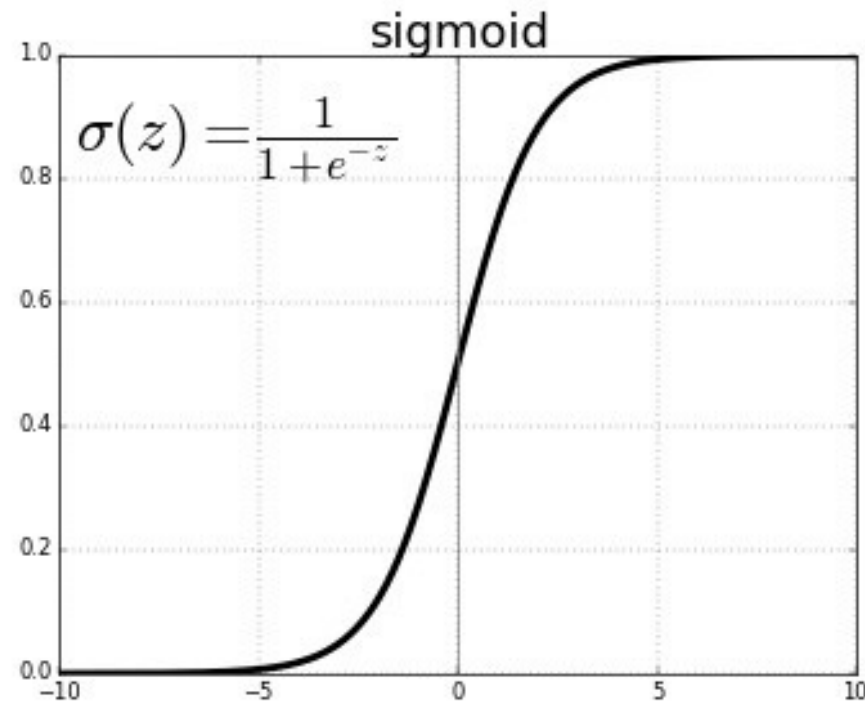
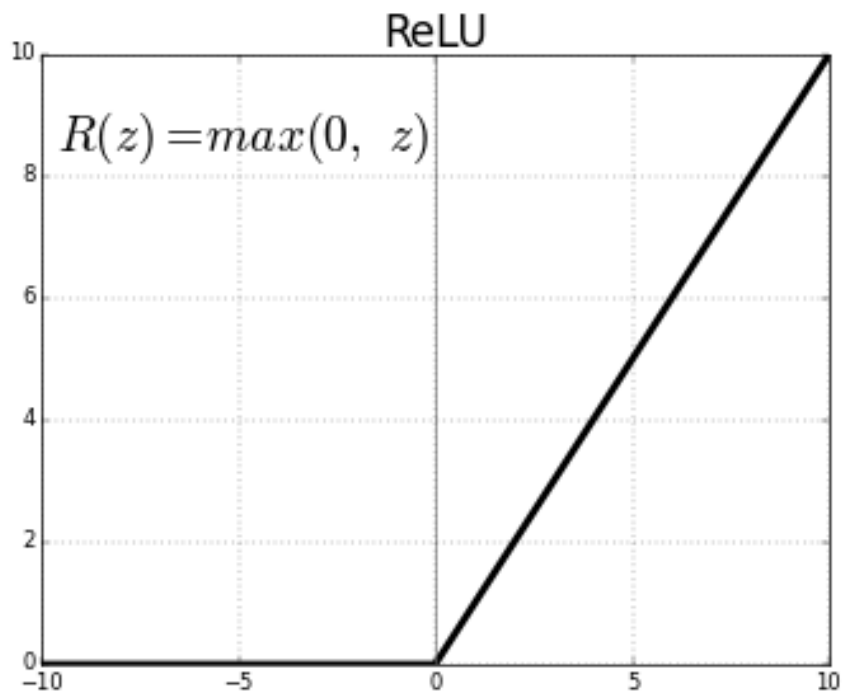


Modelamento matemático

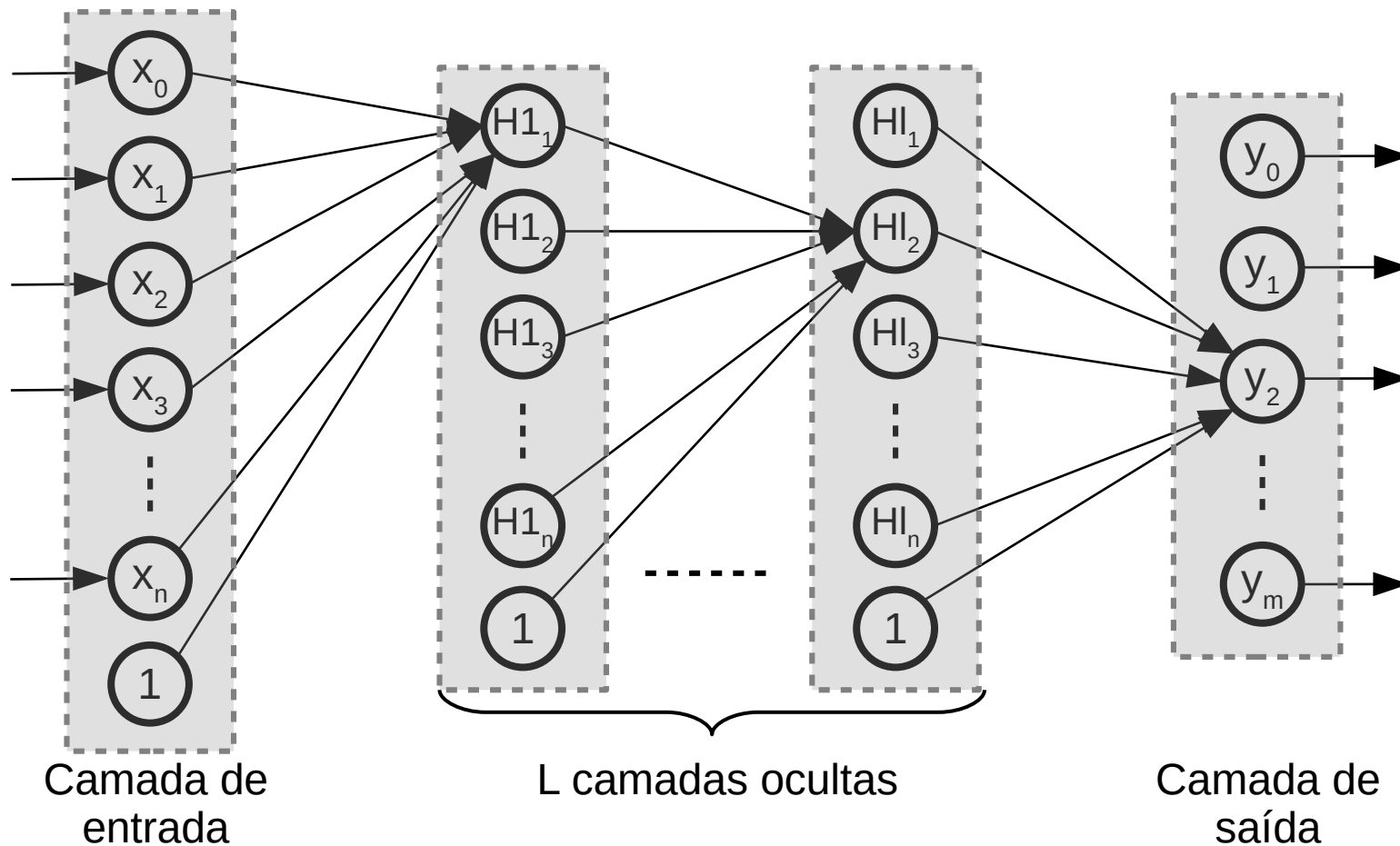


$$y = f\left(\sum_{i=0}^n (W_i \cdot X_i) + B\right)$$

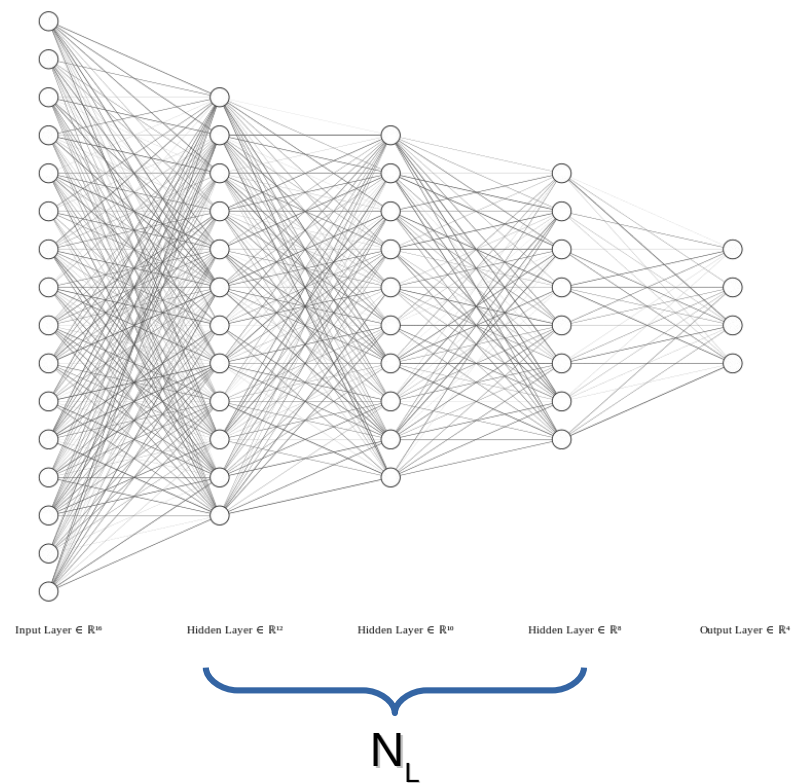
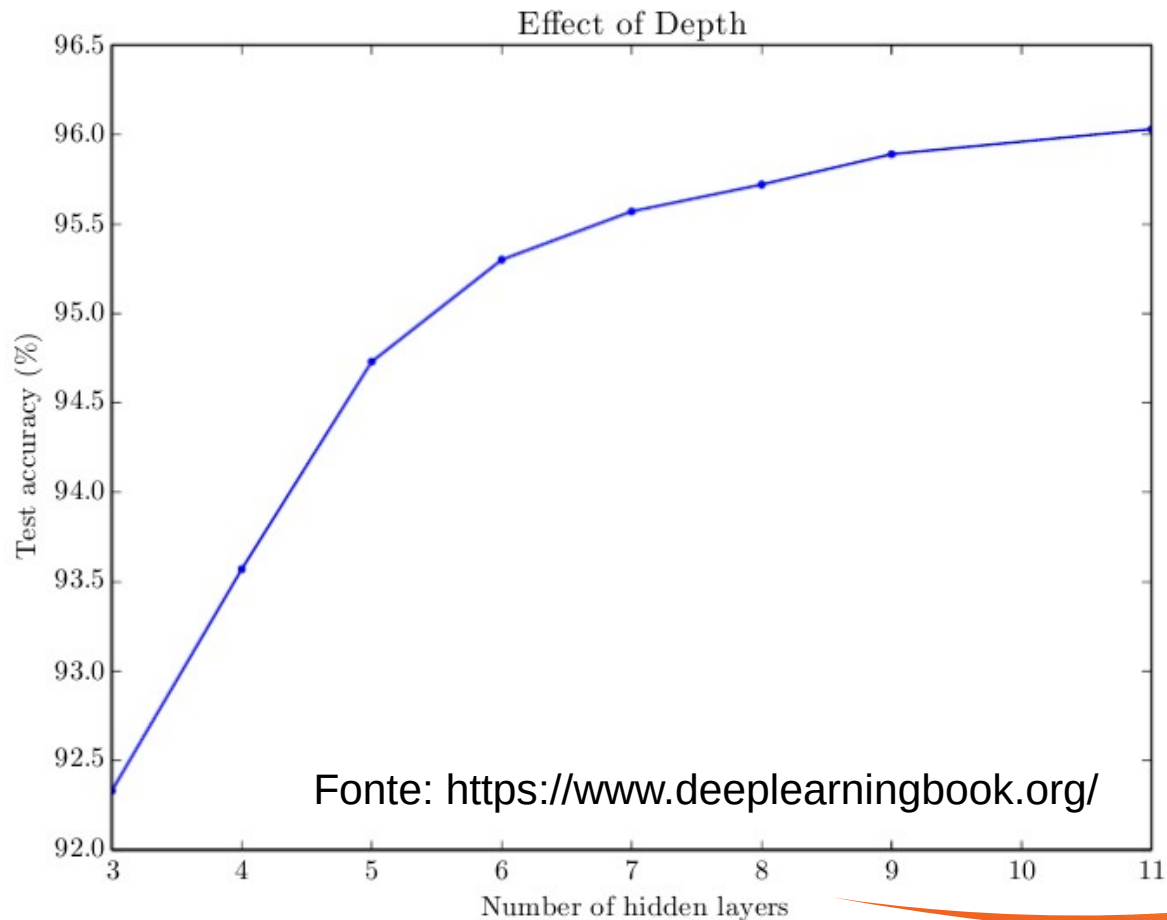
Função de ativação



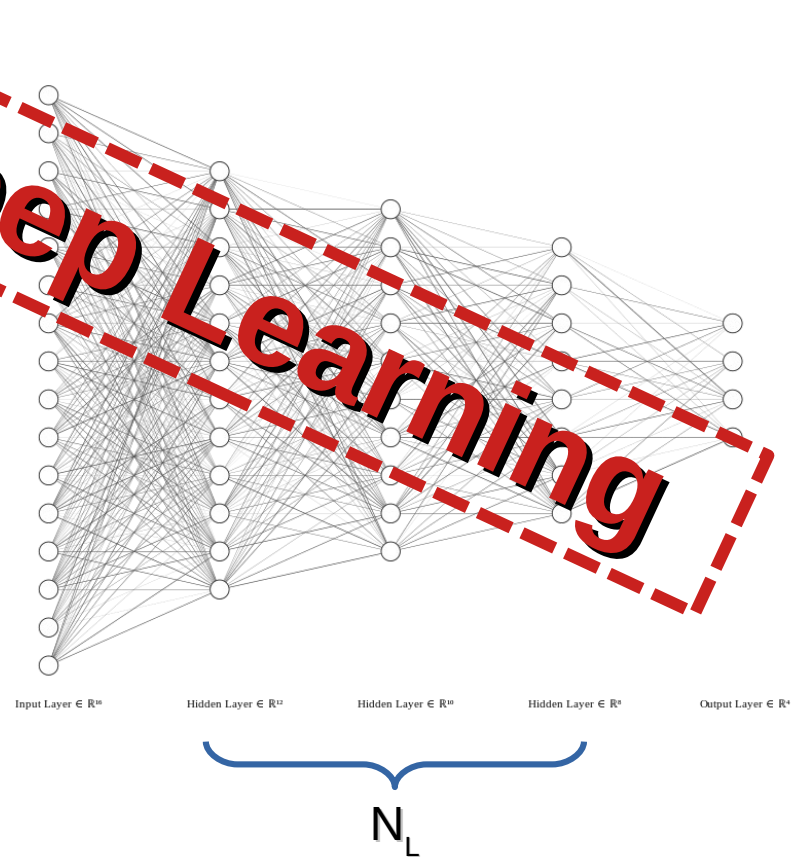
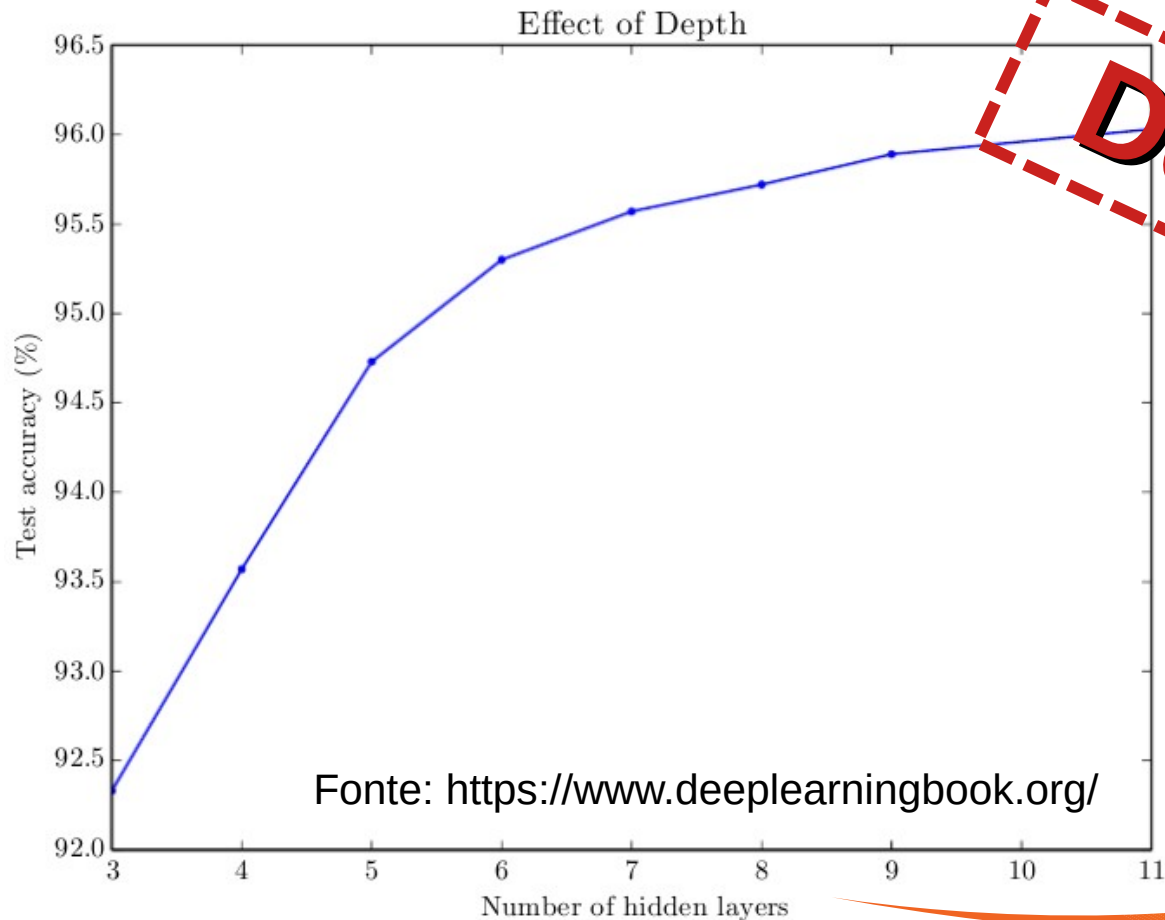
Modelamento matemático



Efeito da quantidade de camadas






Efeito da quantidade de camadas



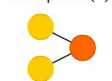
Neural Nets Zoo

Neural Networks

©2016 Fjodor van Veen - asimovinstitute.org

-  Backfed Input Cell
-  Input Cell
-  Noisy Input Cell
-  Hidden Cell
-  Probabilistic Hidden Cell
-  Spiking Hidden Cell
-  Output Cell
-  Match Input Output Cell
-  Recurrent Cell
-  Memory Cell
-  Different Memory Cell
-  Kernel
-  Convolution or Pool

Perceptron (P)



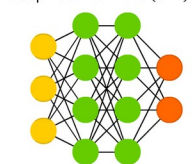
Feed Forward (FF)



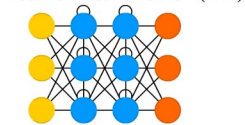
Radial Basis Network (RBF)



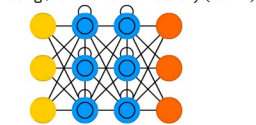
Deep Feed Forward (DFF)



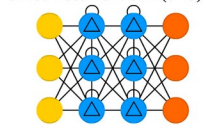
Recurrent Neural Network (RNN)



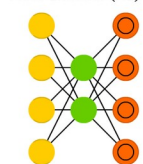
Long / Short Term Memory (LSTM)



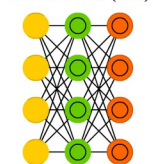
Gated Recurrent Unit (GRU)



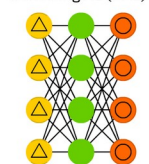
Auto Encoder (AE)



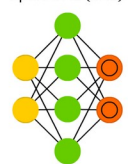
Variational AE (VAE)



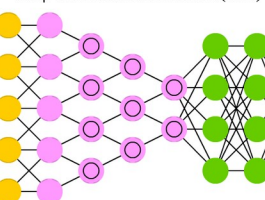
Denoising AE (DAE)



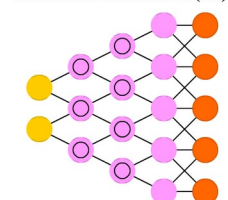
Sparse AE (SAE)



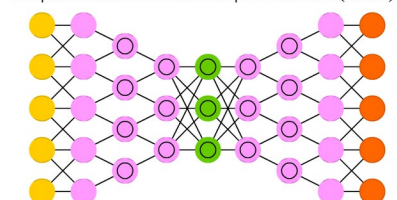
Deep Convolutional Network (DCN)



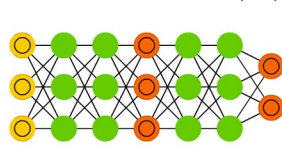
Deconvolutional Network (DN)



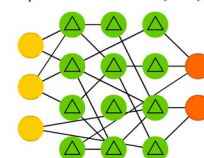
Deep Convolutional Inverse Graphics Network (DCIGN)



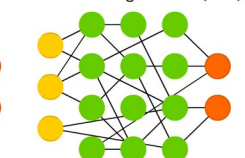
Generative Adversarial Network (GAN)



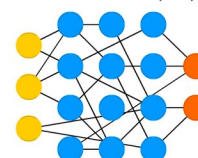
Liquid State Machine (LSM)



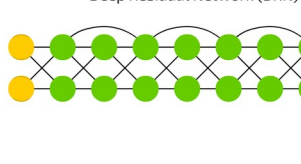
Extreme Learning Machine (ELM)



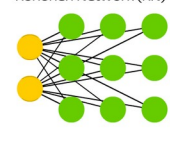
Echo State Network (ESN)



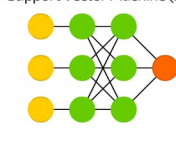
Deep Residual Network (DRN)



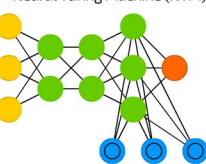
Kohonen Network (KN)



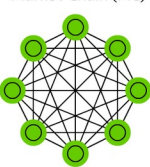
Support Vector Machine (SVM)



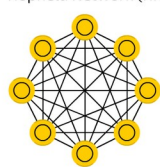
Neural Turing Machine (NTM)



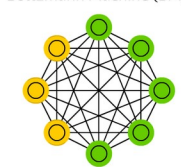
Markov Chain (MC)



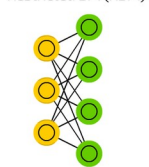
Hopfield Network (HN)



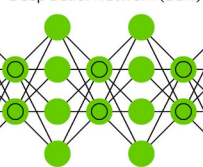
Boltzmann Machine (BM)



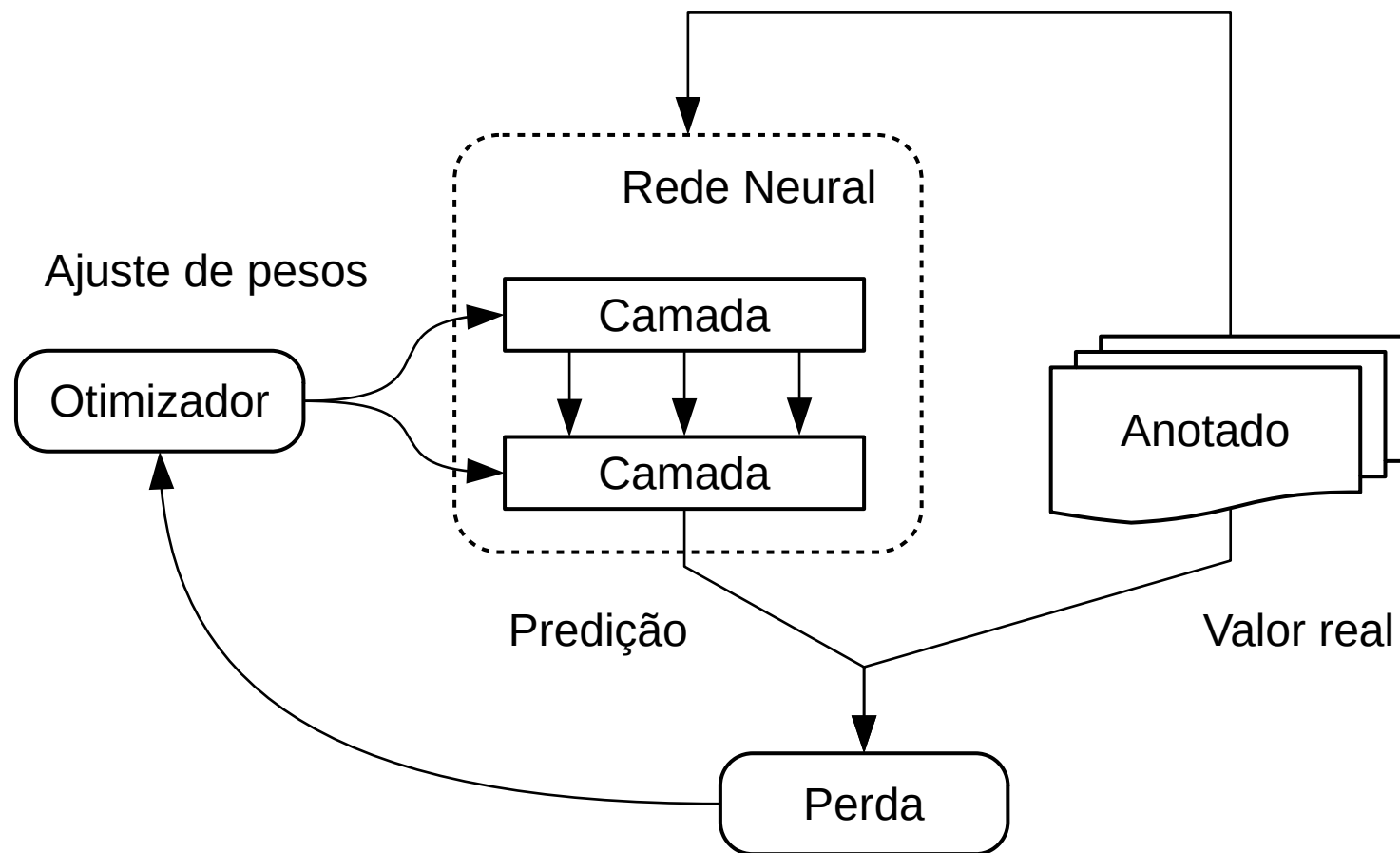
Restricted BM (RBM)



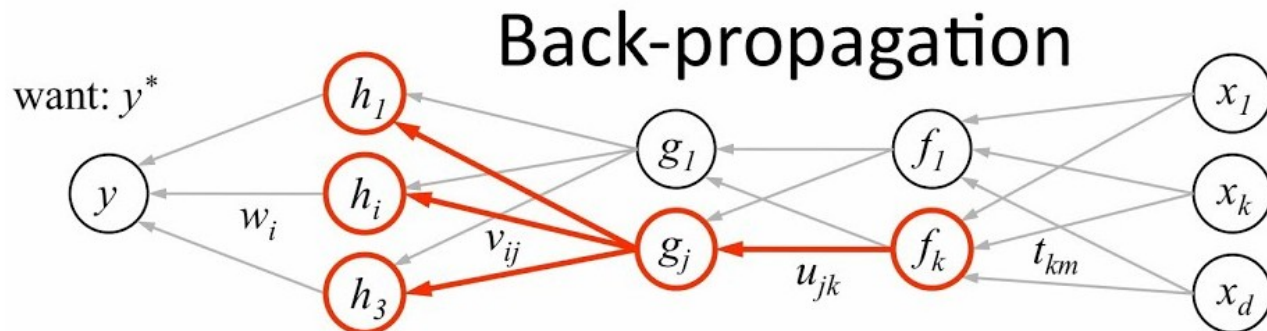
Deep Belief Network (DBN)



Aprendizado supervisionado



Aprendizado supervisionado



1. receive new observation $\mathbf{x} = [x_1 \dots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer $1 \dots L$
compute g_j based on units f_k from previous layer: $g_j = \sigma \left(u_{j0} + \sum_k u_{jk} f_k \right)$
3. get prediction y and error $(y - y^*)$
4. **back-propagate error:** for each unit g_j in each layer $L \dots 1$

(a) compute error on g_j

$$\underbrace{\frac{\partial E}{\partial g_j}}_{\text{should } g_j \text{ be higher or lower?}} = \sum_i \underbrace{\sigma'(h_i)}_{\text{how } h_i \text{ will change as } g_j \text{ changes}} \underbrace{v_{ij}}_{\text{was } h_i \text{ too high or too low?}} \underbrace{\frac{\partial E}{\partial h_i}}_{\text{was } h_i \text{ too high or too low?}}$$

(b) for each u_{jk} that affects g_j

(i) compute error on u_{jk}

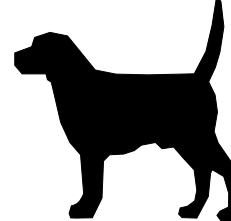
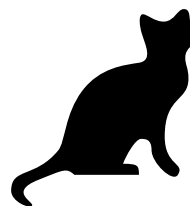
$$\frac{\partial E}{\partial u_{jk}} = \underbrace{\frac{\partial E}{\partial g_j}}_{\text{do we want } g_j \text{ to be higher/lower}} \underbrace{\sigma'(g_j) f_k}_{\text{how } g_j \text{ will change if } u_{jk} \text{ is higher/lower}}$$

(ii) update the weight

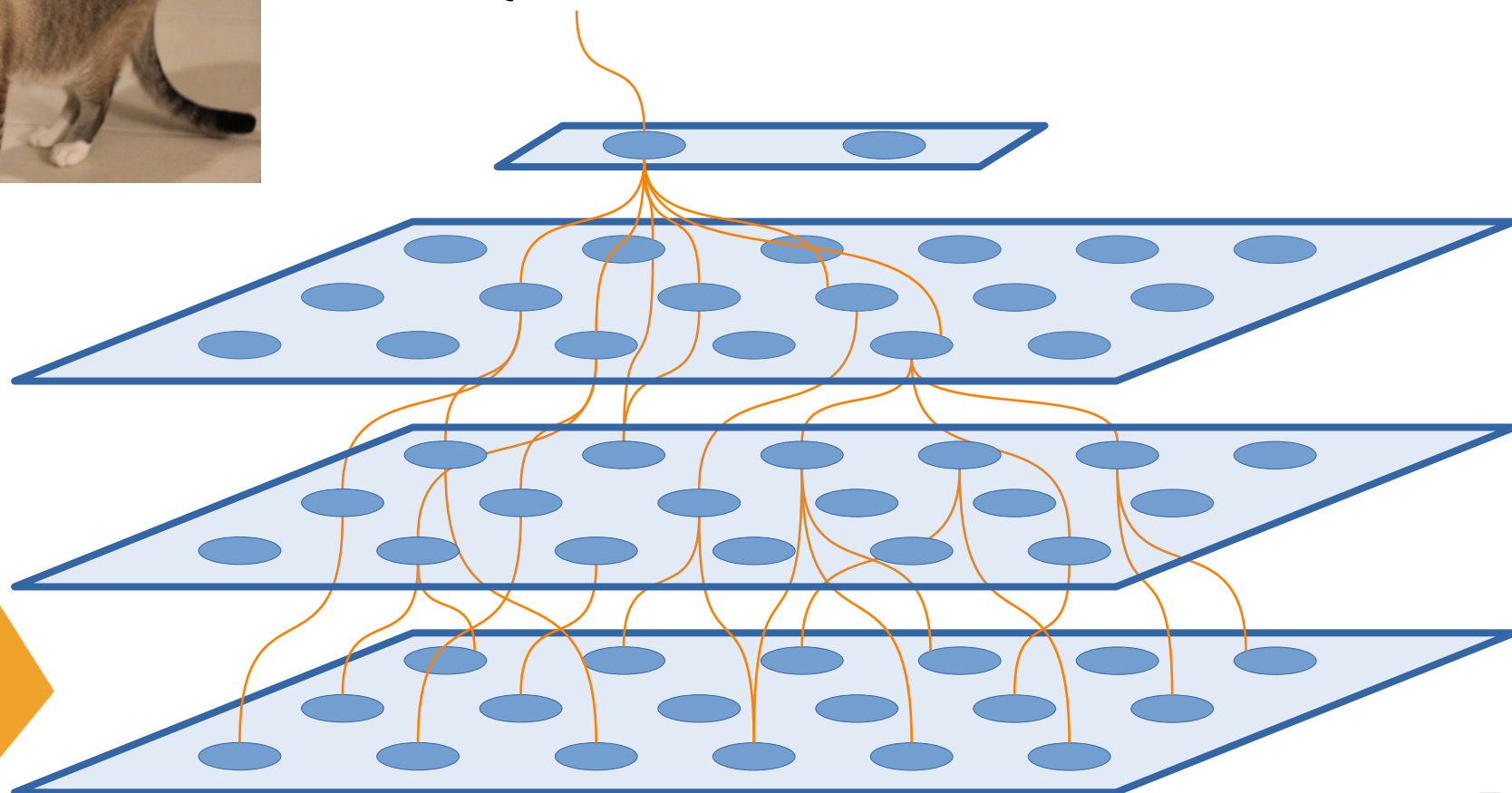
$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$$

■ Usos para RNA's

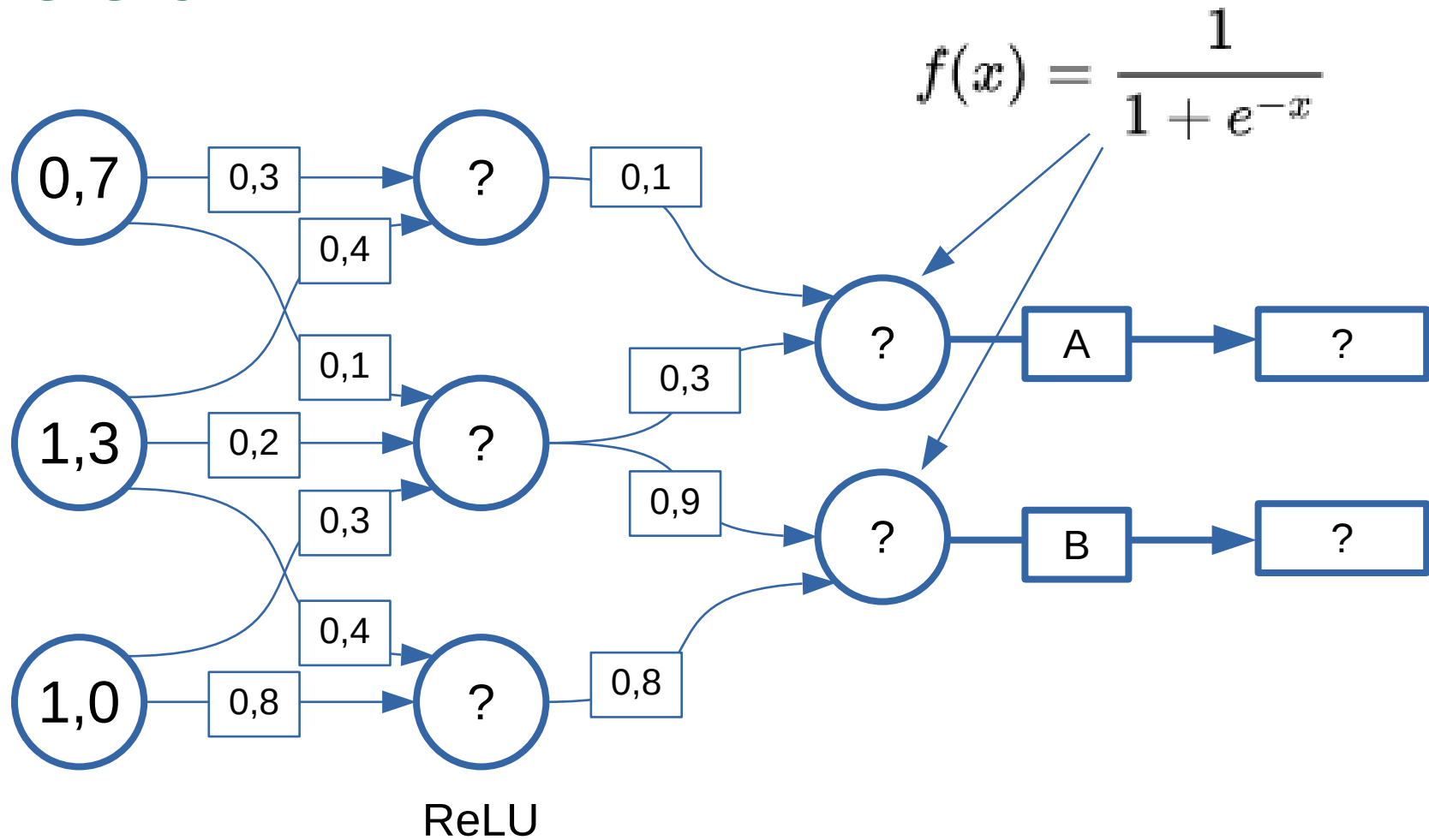
- Classificação
- Regressão
- Redução de dimensionalidade



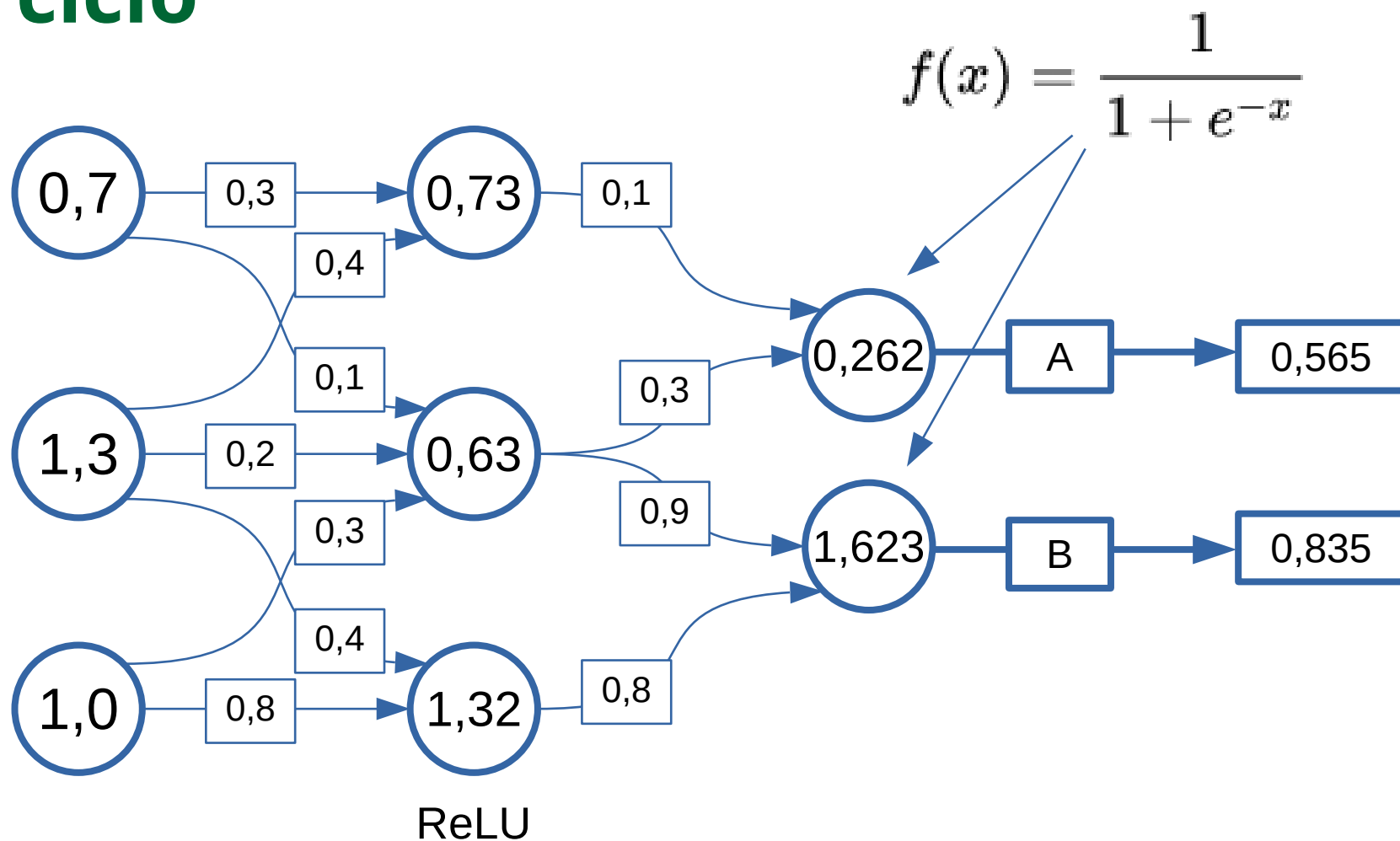
Classificação



Exercício



Exercício



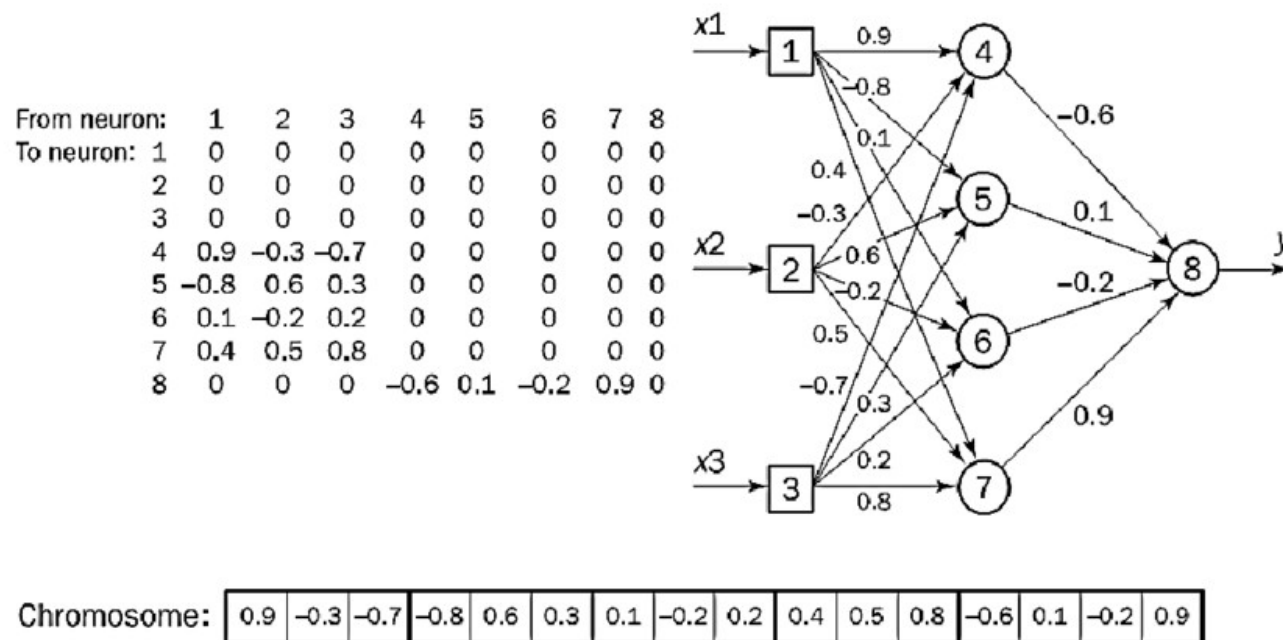
■ Algoritmos Genéticos + RNA's

- Evoluir certos aspectos de uma RNA:
 - ❖ Pesos sinápticos (p/ estrutura fixa)
 - ❖ Estrutura (quantidades de camadas e neurônios por camada)
 - ❖ Regra de aprendizado (Learning rule)
 - ❖ Hiperparâmetros

■ Algoritmos Genéticos + RNA's - Características

- Evoluir os parâmetros da RNA implica em substituir o processo de aprendizado
 - ❖ As RNA's geradas são testadas em situações de uso e ranqueadas de acordo com algum critério
 - ❖ As melhores candidatas passam para a próxima geração via algum mecanismo de GA
 - Crossover
 - Mutação
 - Elitismo

Representação cromossomal direta



Topologia fixa,
pesos variáveis

Representação cromossomial

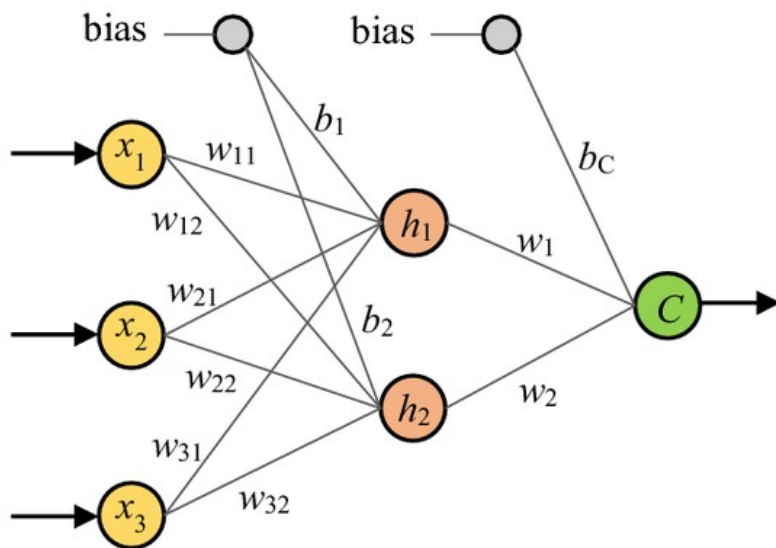
Topology part:

2

Weight part:

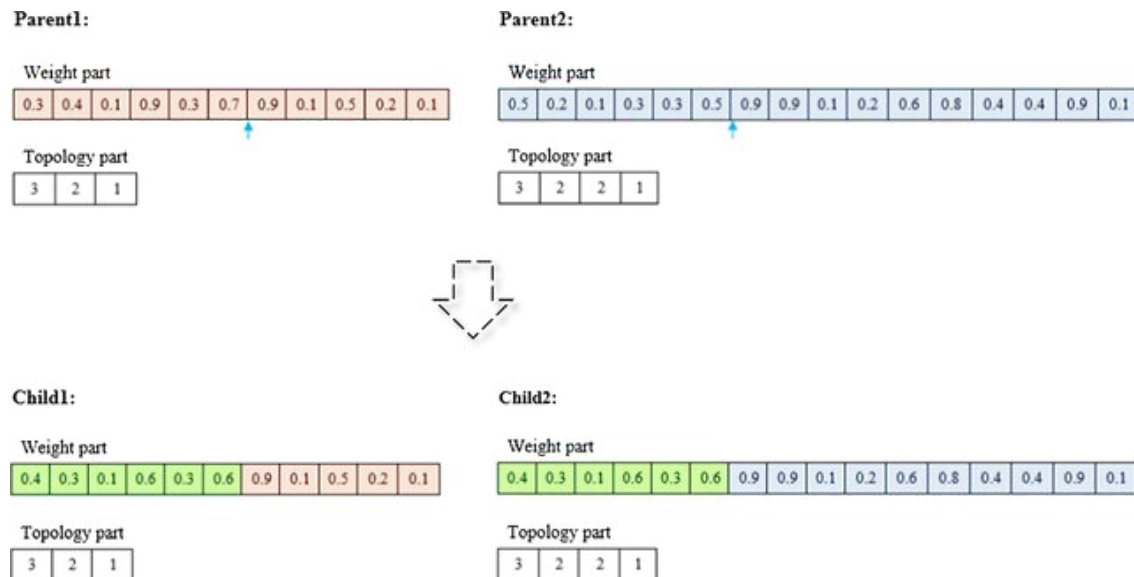
b_1	w_{11}	w_{21}	w_{31}	b_2	w_{12}	w_{22}	w_{32}	b_C	w_1	w_2
-------	----------	----------	----------	-------	----------	----------	----------	-------	-------	-------

Topologia fixa,
pesos variáveis

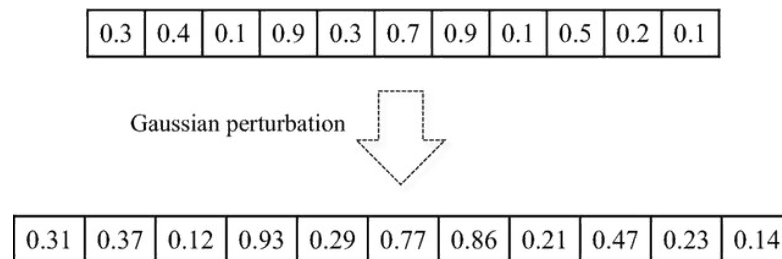


Operadores genéticos

Crossover



Mutação



■ Representação cromossomial gramatical

- Evolui topologias a partir de regras (gramática)
 - ❖ Gramática é o genótipo
 - ❖ RNA gerada é o fenótipo
 - ❖ RNA precisa ser treinada (backpropagation) e depois testada para determinar o seu fitness

■ Evolução de hiperparâmetros

- Evolui a partir de topologias fixas
 - ❖ Cromossomos codificam hiperparâmetros
 - ❖ As RNA's são geradas, treinadas e testadas
 - ❖ Resultado serve p/ determinação do fitness

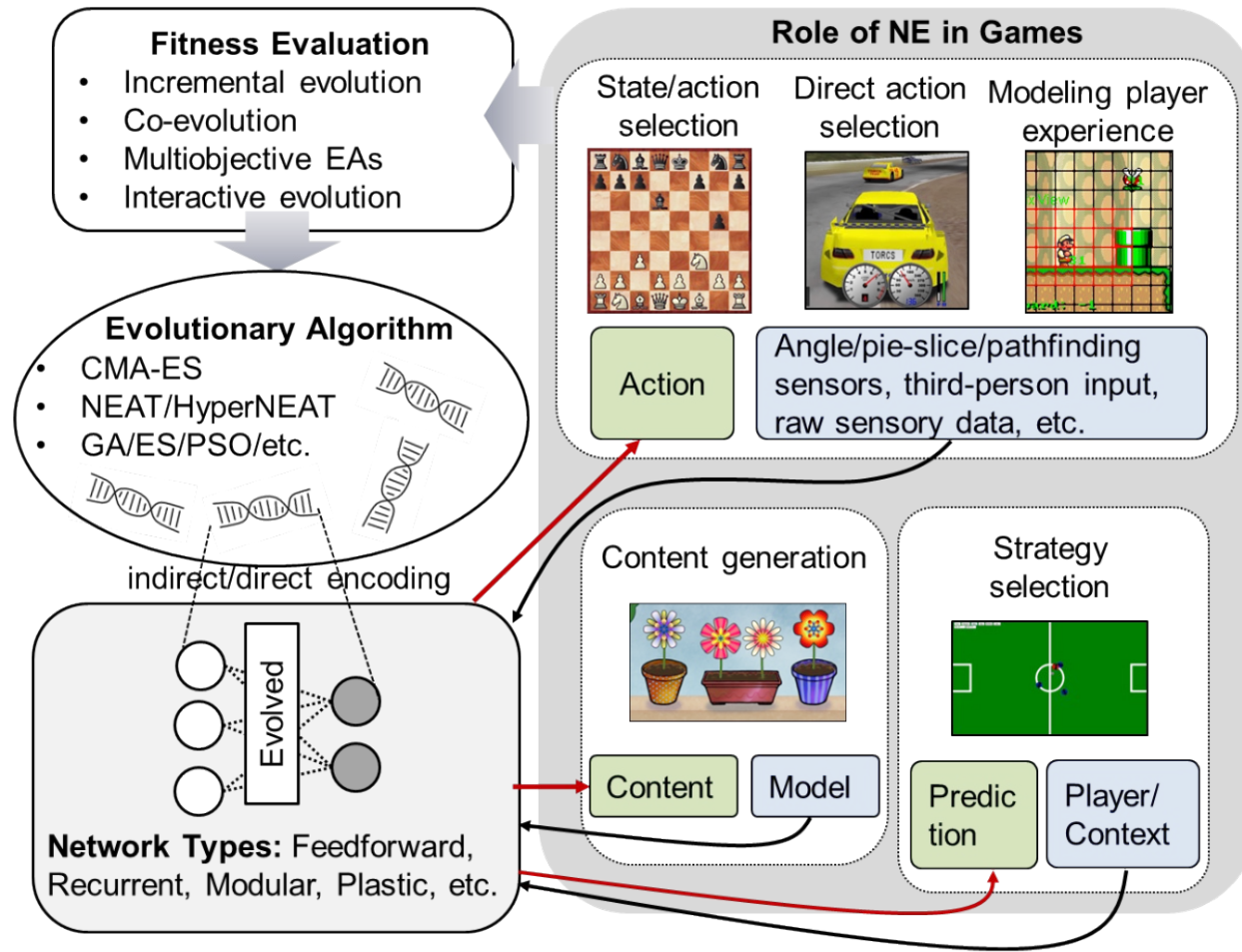
■ Neuroevolução

- Mecanismos de evolução baseados em:
 - ❖ Algoritmos Genéticos (GA)
 - ❖ Estratégias Evolucionárias (ES)
 - ❖ Programação Genética (GP)

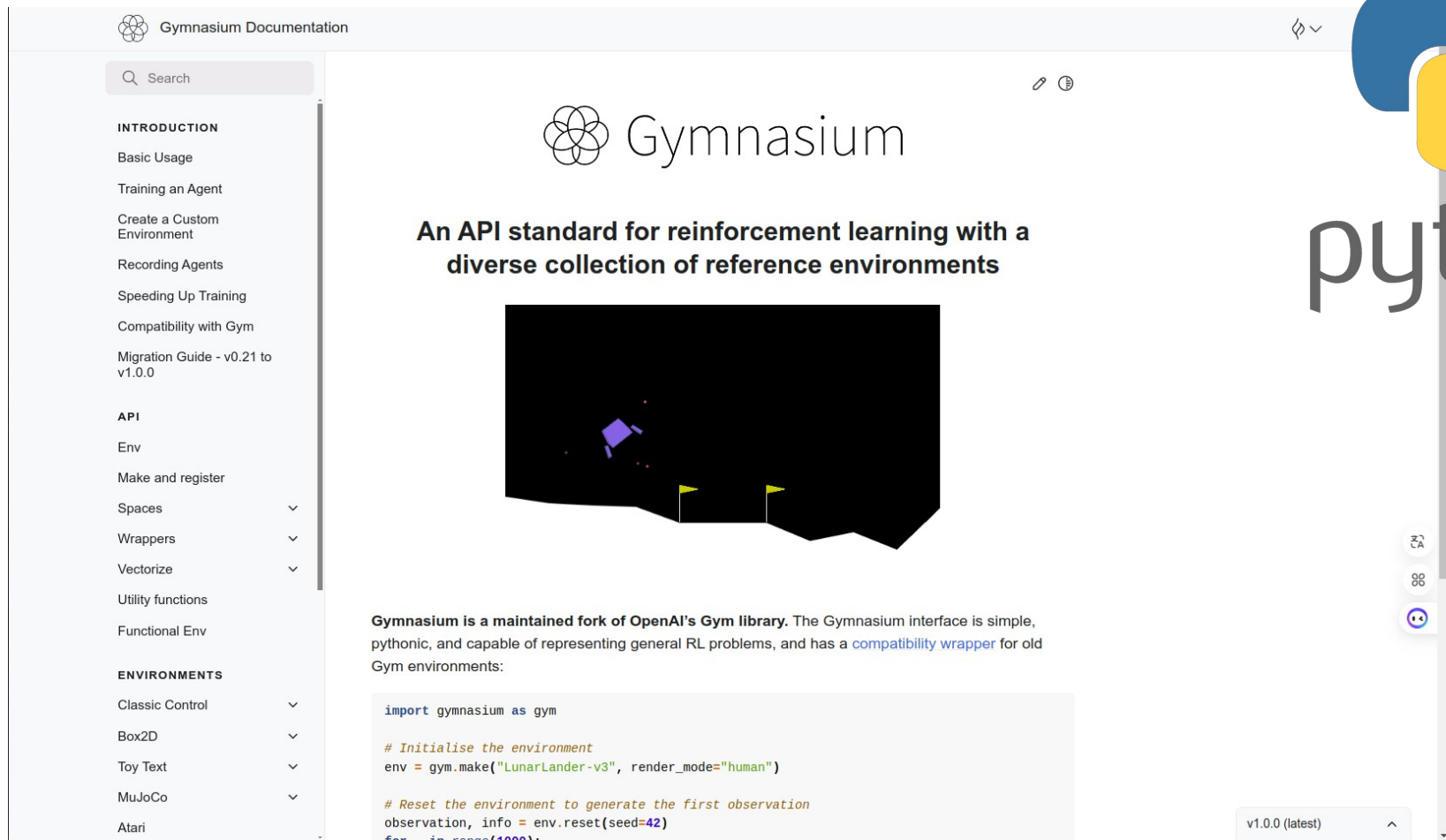
■ Neuroevolução – Estratégias Evolutivas

- Ajuste do desvio padrão de cada perturbação gaussiana (mecanismo de mutação) de acordo com a função de perda (geralmente MSE). Desse modo, o passo de adaptação é grande enquanto o erro for alto (maior exploração) e diminui conforme o erro cai (maior exploração). Assim, a busca da solução fica mais próxima dos pontos de menor erro.

Neuroevolução em jogos



Neuroevolução em jogos



Gymnasium Documentation

Search

INTRODUCTION

- Basic Usage
- Training an Agent
- Create a Custom Environment
- Recording Agents
- Speeding Up Training
- Compatibility with Gym
- Migration Guide - v0.21 to v1.0.0

API

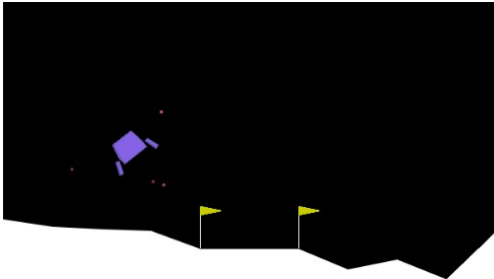
- Env
- Make and register
- Spaces
- Wrappers
- Vectorize
- Utility functions
- Functional Env

ENVIRONMENTS

- Classic Control
- Box2D
- Toy Text
- MuJoCo
- Atari

Gymnasium

An API standard for reinforcement learning with a diverse collection of reference environments



Gymnasium is a maintained fork of OpenAI's Gym library. The Gymnasium interface is simple, pythonic, and capable of representing general RL problems, and has a [compatibility wrapper](#) for old Gym environments:

```
import gymnasium as gym

# Initialise the environment
env = gym.make("LunarLander-v3", render_mode="human")

# Reset the environment to generate the first observation
observation, info = env.reset(seed=42)

for _ in range(1000):
```

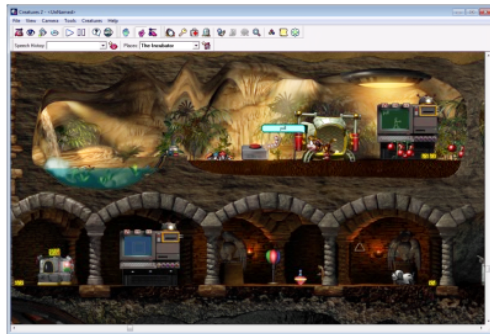
v1.0.0 (latest)



Neuroevolução em jogos



(a)



(b)



(c)



(d)

Fig. 2. **Neuroevolution in Existing Games.** (a) NE is able to discover high-performing controllers for racing games such as TORCS [11]. (b) NE has also been successfully applied to commercial games, such as Creatures [44]. Additionally, NE enables new types of games such as GAR (c), in which players can interactively evolve particular weapons [46], or NERO (d), in which players are able to evolve a team of robots and battle them against other players [119].

Neuroevolução em jogos

The Role of Neuroevolution in Selected Games. ES = evolutionary strategy, GA = genetic algorithm, MLP = multi-layer perceptron, MO = multiobjective, TP = third-person (input not tied to a specific frame of reference, e.g. number edible ghosts) , UD = user-defined network topology, PA = performance alone

NE Role (Section III)	Game	ANN Type (Section IV)	NE Methods (Section V)	Fitness Evaluation (Section VI)	Input Representation (Section VII)
State/action evaluation	Checkers [32] Chess [32] Othello [79] Go (7×7) [38] Ms. Pac-Man [71] Simulated Car Racing [74]	MLP MLP MLP CPPN (MLP) MLP MLP	UD, GA UD, GA Marker-based [34] HyperNEAT UD, ES UD, ES	Coevolution PA (positional values) Cooperative coevolution PA (score+board size) PA (average score) PA (waypoints visited)	TP (piece type) TP (piece type) TP (piece type) TP (piece type) Path-finding Speed, pos, waypoints
Direct action selection	Quake II [85] [86] Unreal Tournament [135] Go (7×7) [118] Simulated Car Racing [124] Keepaway Soccer [122] Battle Domain [105] NERO [119] Ms. Pac-Man [106] Simulated Car Racing [29] Atari [48] Creatures [44]	MLP Recurrent, LSTM MLP MLP MLP MLP MLP Modular MLP MLP CPPN (MLP) Modular MLP	UD, GA UD, GA, NSGA-II NEAT UD, ES NEAT NEAT, NSGA-II NEAT NEAT, NSGA-II UD, GA HyperNEAT GA	PA (kill count) MO (damage&accuracy) Transfer Learning Incremental Evolution Transfer Learning MO+Incremental Interactive Evolution MO (pills&ghosts eaten) PA (distance) PA (game score) Interactive Evolution	Visual Input (14×2) Pie-slice, way point, etc. Roving Eye (3×3) Rangefinders, waypoints Distances Angle, straight line Rangefinders, pie-slice Path-finding Roving Eye (5×5) Raw input (16×21) TP (e.g. type of object)
Selection between strategies	Keepaway Soccer [142] [143] EvoCommander [56]	MLP MLP	NEAT NEAT	PA (hold time) Interactive Evolution	Angle and distance Pie-slice, rangefinder
Modelling opponent strategy	Texas Hold'em Poker [66]	MLP	NEAT	PA (%hands won)	TP (e.g. size of pot, cost of a bet, etc.)
Content generation	GAR [46] Petalz [97]	CPPN (MLP) CPPN (MLP)	NEAT NEAT	Interactive Evolution Interactive Evolution	Model Model
Modelling player experience	Super Mario Bros [87]	MLP, Perceptron	UD, GA	PA (player preference)	TP (e.g. gap width, number deaths, etc.)