ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

# PQI 3403 Análise de Processos da Indústria Química

Ardson dos Santos Vianna Júnior - ASVJ
e-mail: ardson@usp.br

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

# Aula
# Redes Neuronais Artificiais (RNA)

## PQI 3403 Análise de Processos da Indústria Química

Ardson dos S. Vianna. Jr. - USP

# Roteiro

Introdução

Definindo a rede

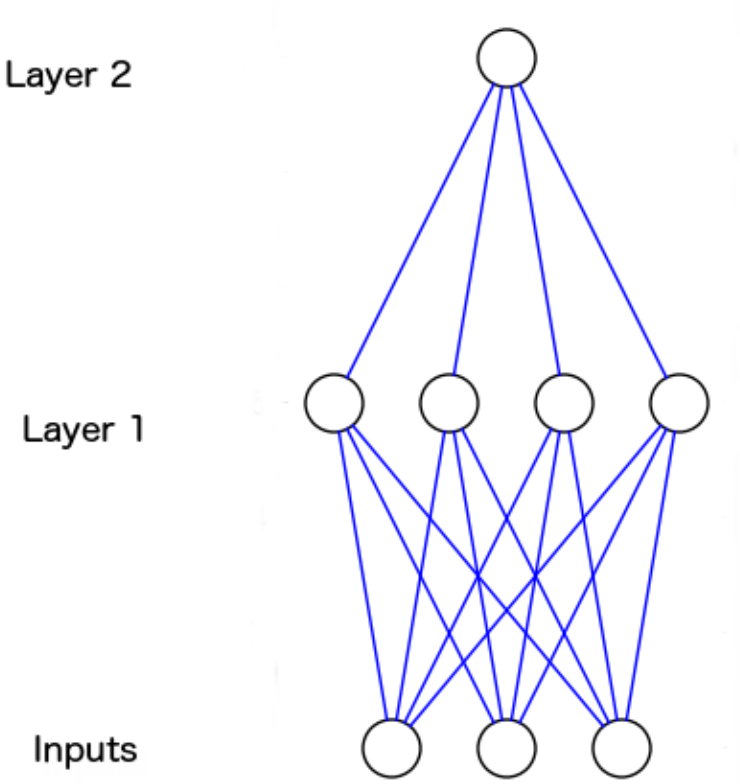
Treinamento

Outras possibilidades

Conclusão

# Componentes de uma RNA

- Uma camada de input  x
- Uma quantidade arbitrária de camadas escondidas (intermediárias)
- Uma camada de saída $\hat{y}$
- Um conjunto de pesos $w_i$ e um erro persistente (bias)
- Escolha de funções de ativação para cada camada escondida.



# Definindo a rede

- Uma camada de input x
- Uma quantidade arbitrária de camadas escondidas (intermediárias)
- Uma camada de saída $\hat{y}$

- `class Neuralayer:`

```
def __init__(self, number_of_neurons,
        number_of_inputs_per_neuron):
```

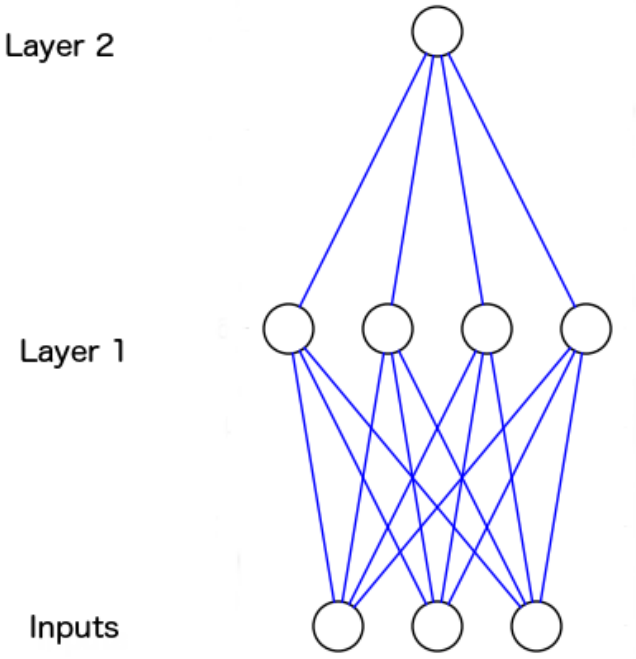
```
class NeuralNetwork:
        def __init__(self, x, y):
                self.input = x
                self.weights1 =
                np.random.rand
                (self.input.shape[1],4)
                self.weights2 =
                np.random.rand(4,1)
                self.y = y
                self.output =
                np.zeros(y.shape)
```

# Definindo a rede

```
# Create layer 1 (4 neurons, each with 3 inputs)
    layer1 = NeuronLayer(4, 3)

# Create layer 2 (a single neuron with 4 inputs)
   layer2 = NeuronLayer(1, 4)

# Combine the layers to create a neural network
   neural_network = NeuralNetwork(layer1, layer2)
```



# Componentes de uma RNA

- Treinamento
- Processo de tentativa e erro

.

- def train(self, training_set_inputs, training_set_outputs, number_of_training_iterations):
- for iteration in xrange(number_of_training_iterations):

# Componentes de uma RNA

Saídas do modelo

```
def think(self, inputs):
        output_from_layer1 = self.__sigmoid(dot(inputs, self.layer1.synaptic_weights))
        output_from_layer2 = self.__sigmoid(dot(output_from_layer1, self.layer2.synaptic_weights))
      return output_from_layer1, output_from_layer2
```

- def __sigmoid(self, x):
      return 1 / (1 + exp(-x))

```
def __sigmoid_derivative(self, x):
      return x * (1 - x)
```

# Função de perda (loss function)

- Soma dos quadrados

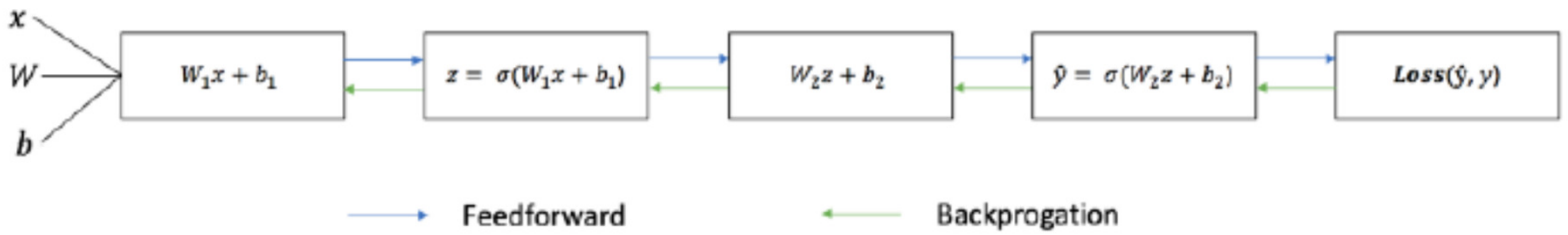
- $SQ = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$

- Minimizar a SQ
- Derivar a função com relação aos pesos e bias
- Gradiente descendente

- def __sigmoid(self, x):
  return 1 / (1 + exp(-x))

def __sigmoid_derivative(self, x):
  return x * (1 - x)

- Treinamento: feedforward e backpropagation



# Conclusão

- Código Python
- Classes - objetos
- Definição da rede
- Treinamento
- resultados

# Bibliografia


- https://github.com/miloharper/multi-layer-neural-network

- https://www.youtube.com/watch?v=VrMHA3yX_QI
- Prof. Patrick Henry Winston