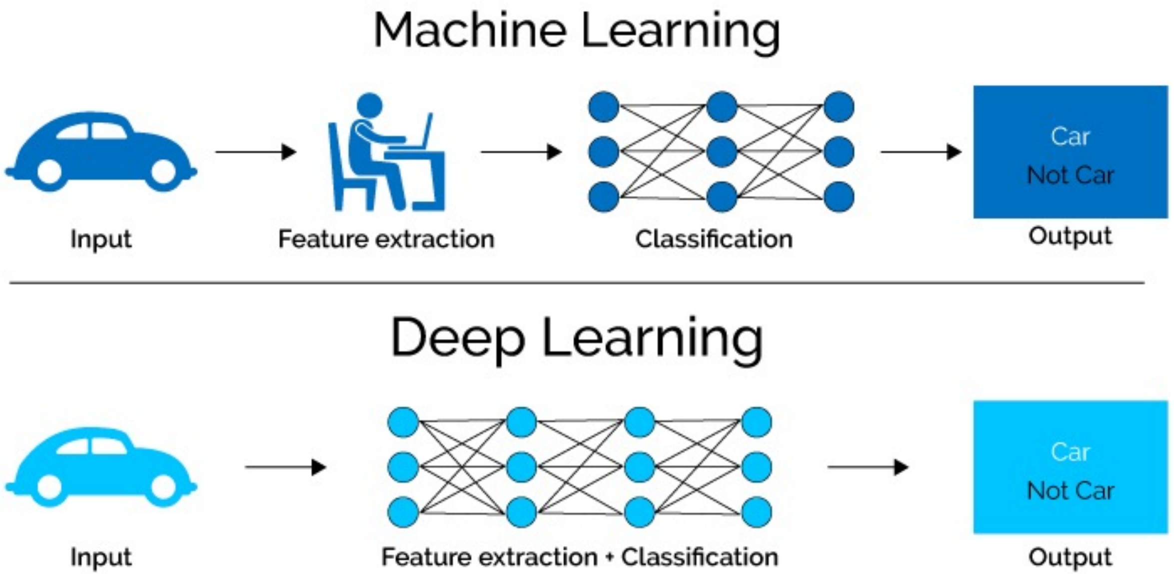
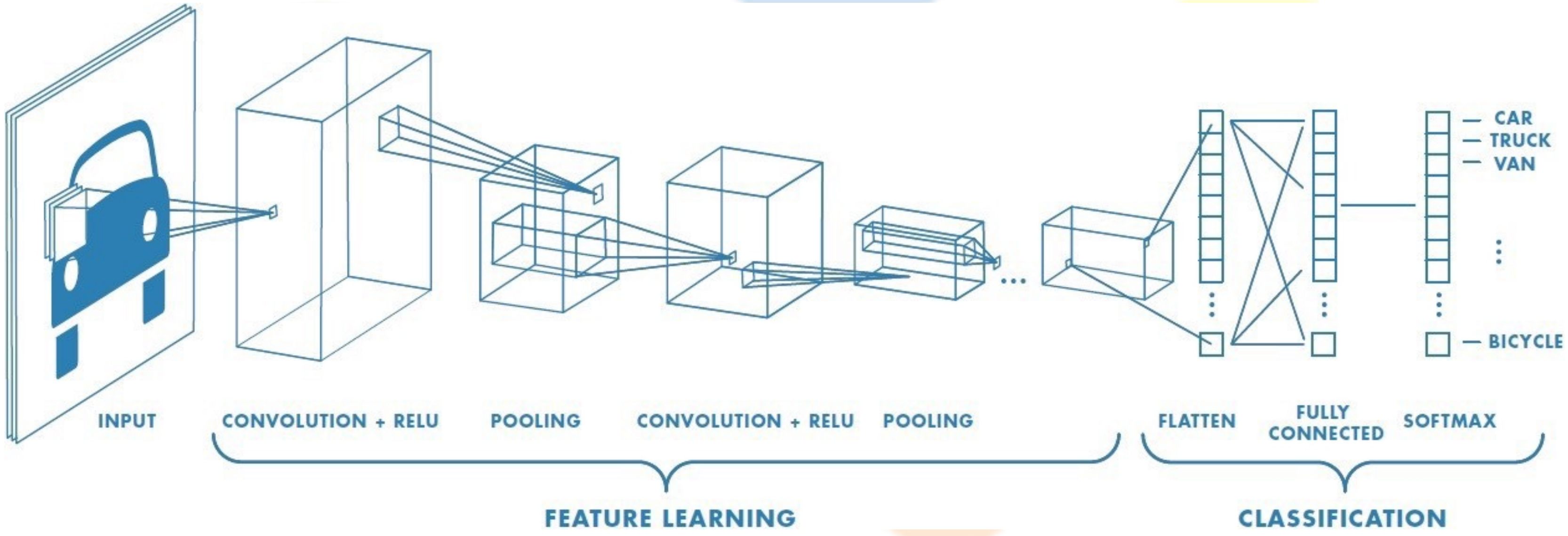
Machine Learning

Data Science Academy

Redes Neurais Artificiais Profundas (Deep Learning)


Machine Learning
Car
Not Car
Input
Feature extraction
Classification
Output
Deep Learning
Car
Not Car
Input
Feature extraction + Classification
Output

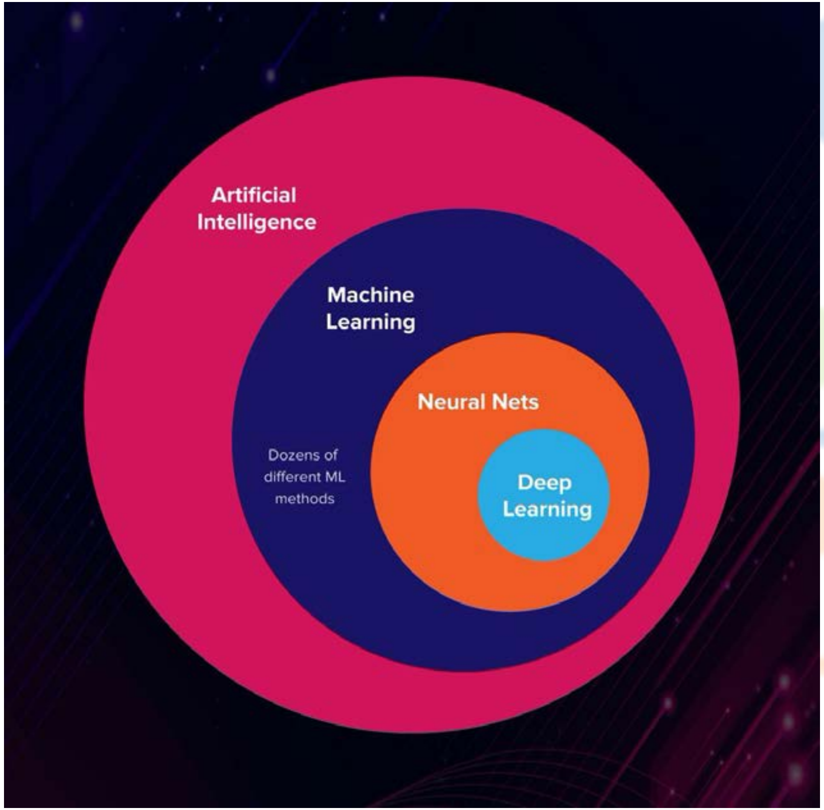

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Este capítulo será uma introdução a Deep Learning, pois esse é um tema bastante extenso!

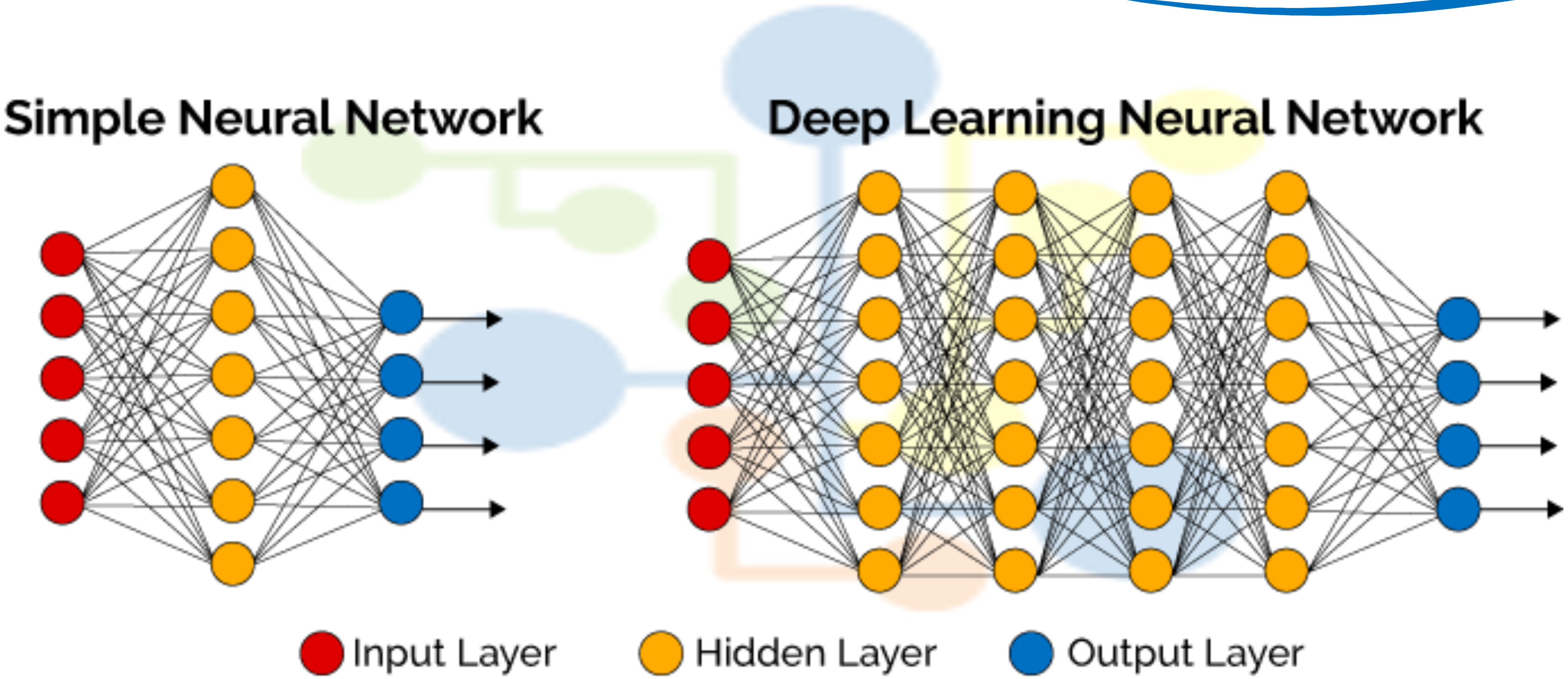
Data Science Academy

O Que é Deep Learning?

www.deeplearningbook.com.br

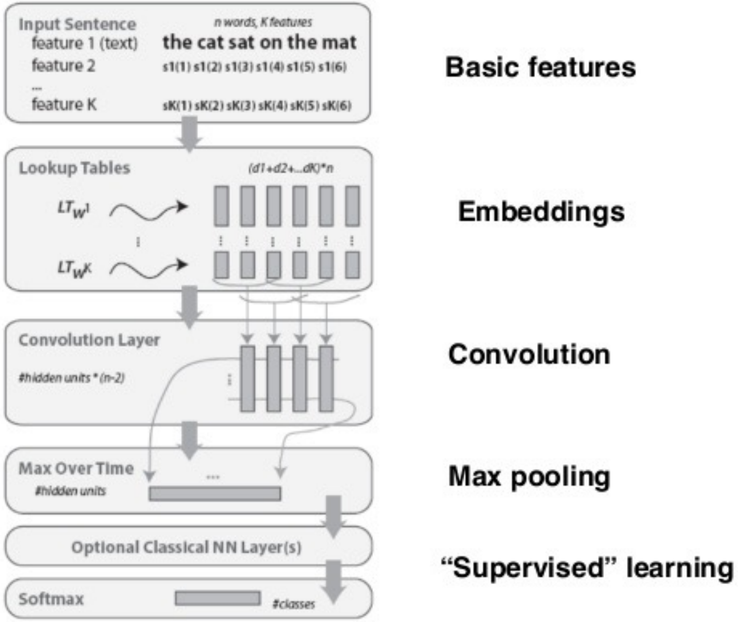


Deep Learning é uma sub-categoria de Machine Learning.


Simple Neural Network
Deep Learning Neural Network
Input Layer
Hidden Layer
Output Layer

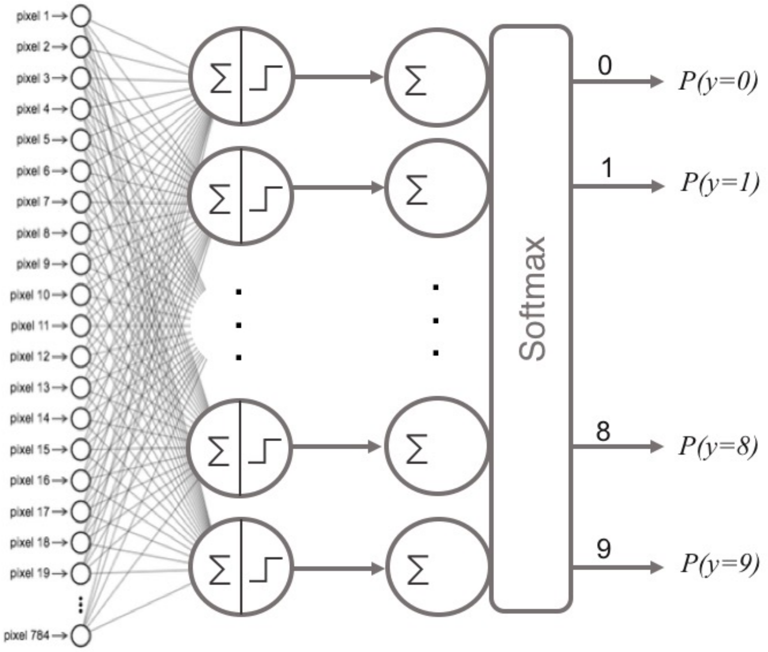
Data Science Academy

O Processo de Aprendizagem de Modelos de Deep Learning

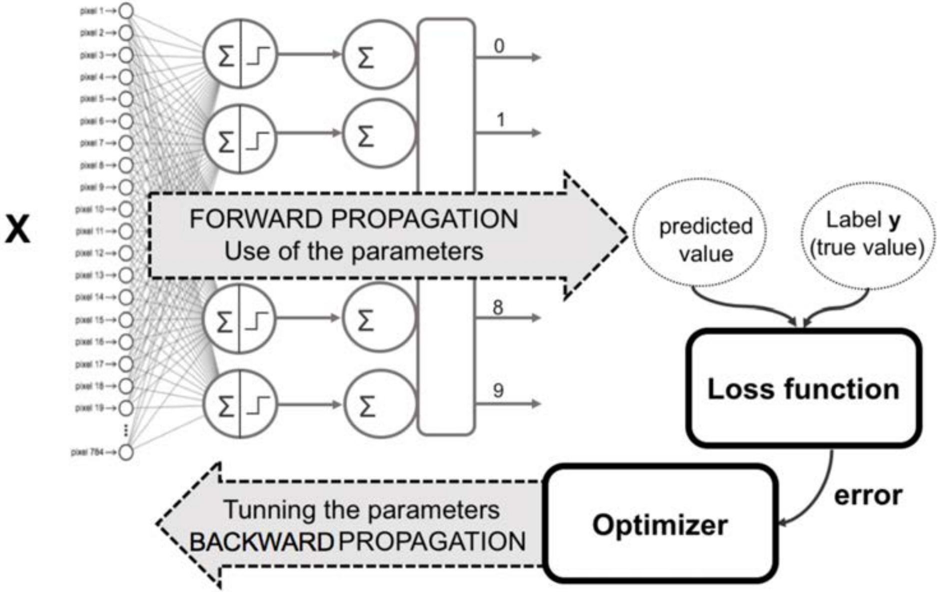


Em Deep Learning nosso trabalho é preparar as estruturas de dados.

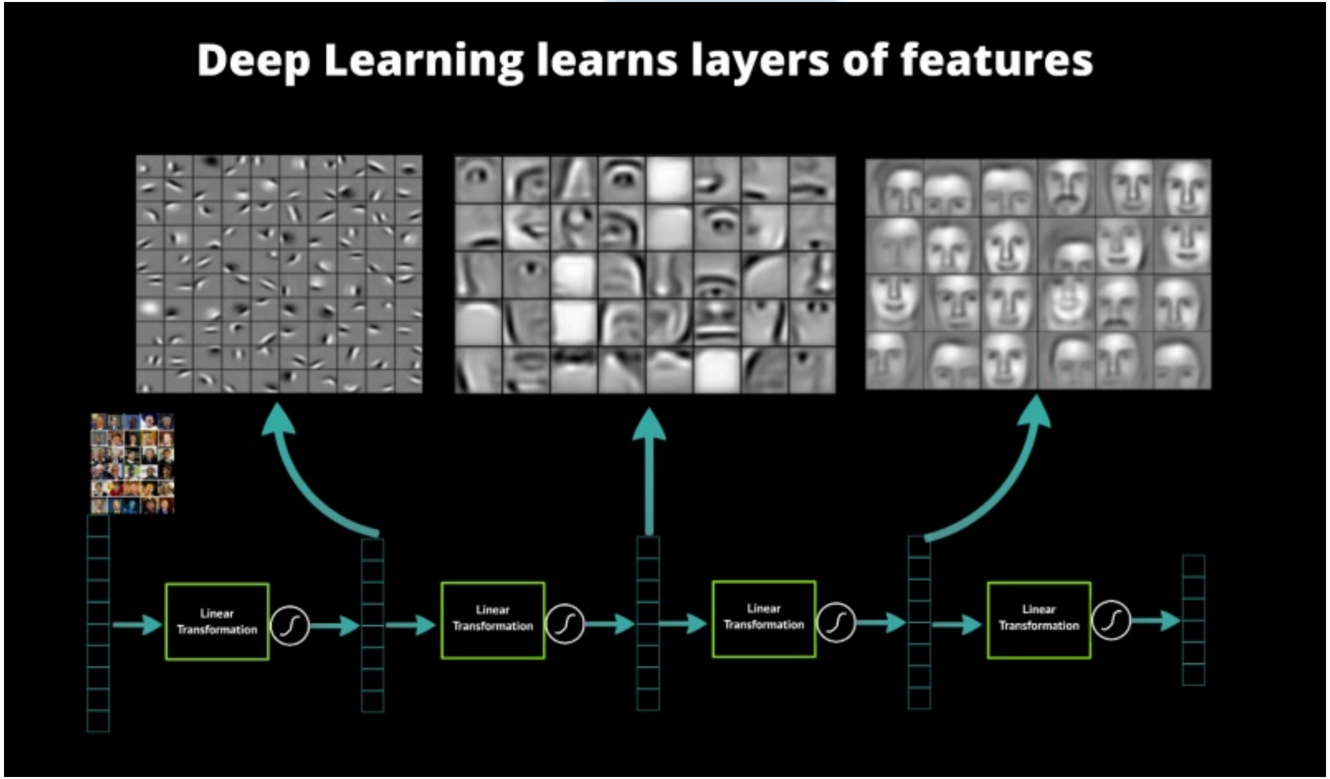
O aprendizado de recursos e a classificação/regressão são ambos feitos pelo modelo.



Em Visão Computacional, primeiro a rede realiza o processo de aprendizagem dos recursos e depois faz a classificação.

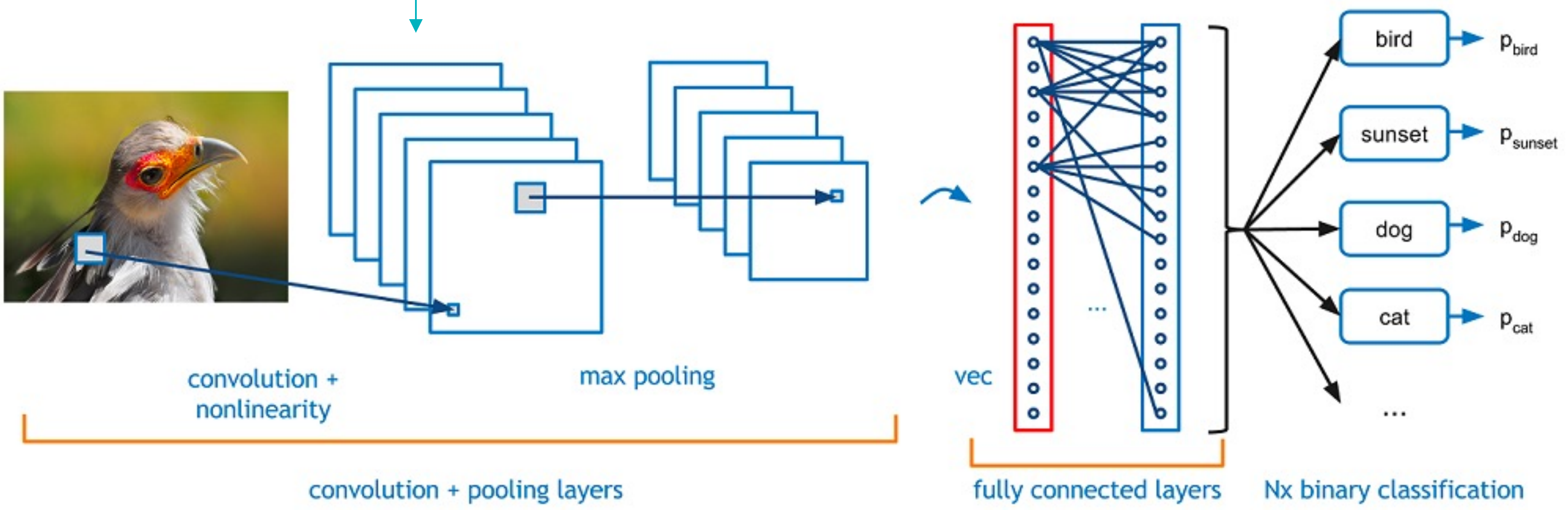


A aprendizagem ocorre em duas etapas: **Forward Propagation** e **Backward Propagation**!

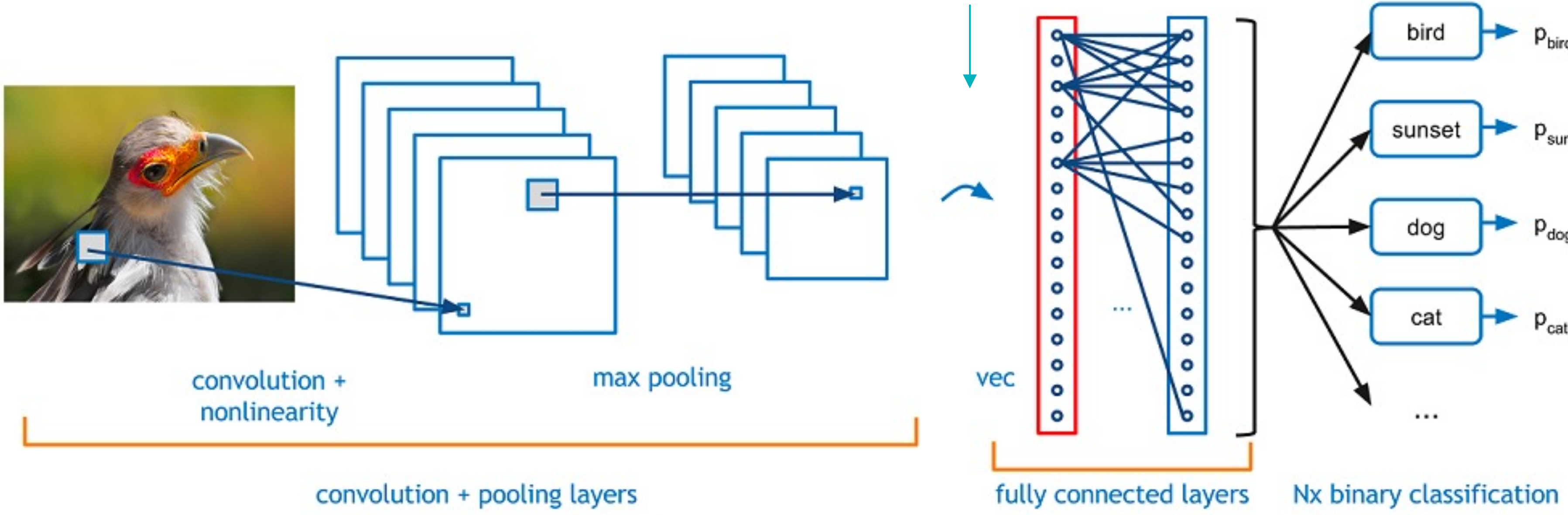

Deep Learning learns layers of features
Linear Transformation
Linear Transformation
Linear Transformation
Linear Transformation

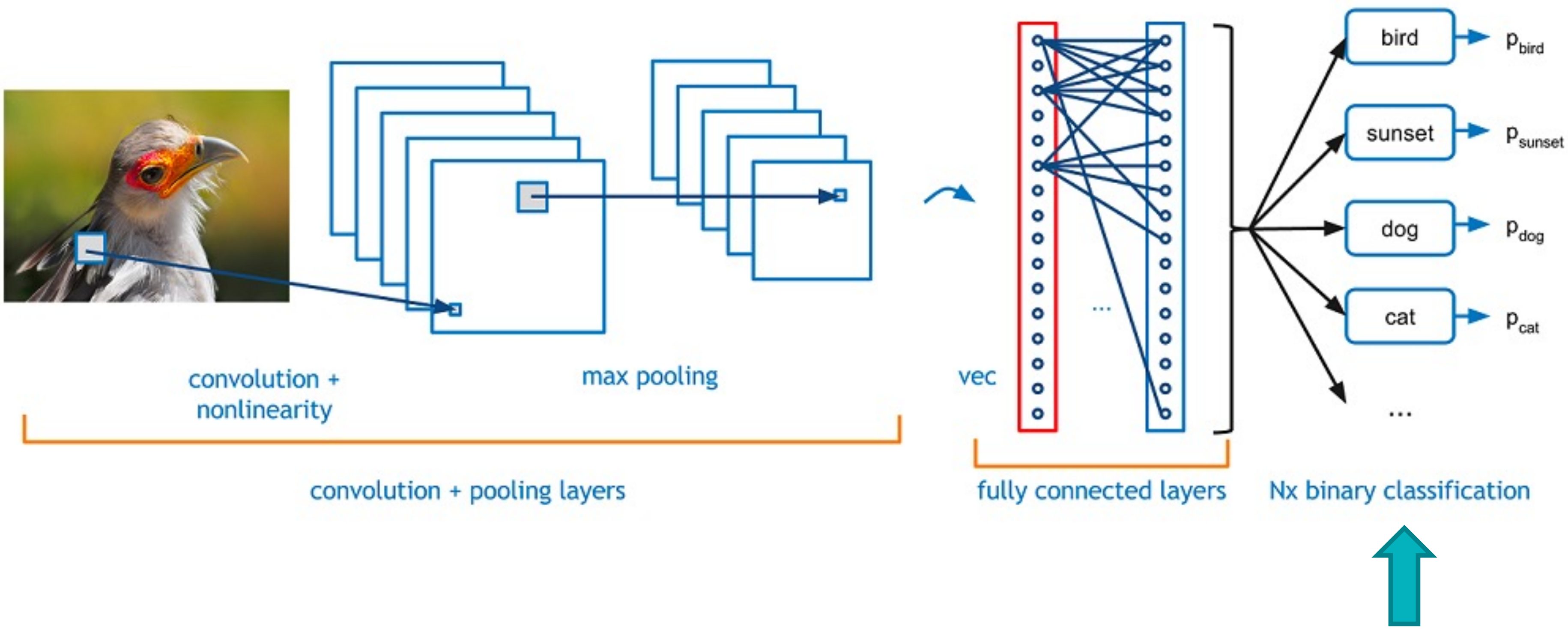
Arquitetura de Redes Neurais Convolucionais

Mapas de Recursos



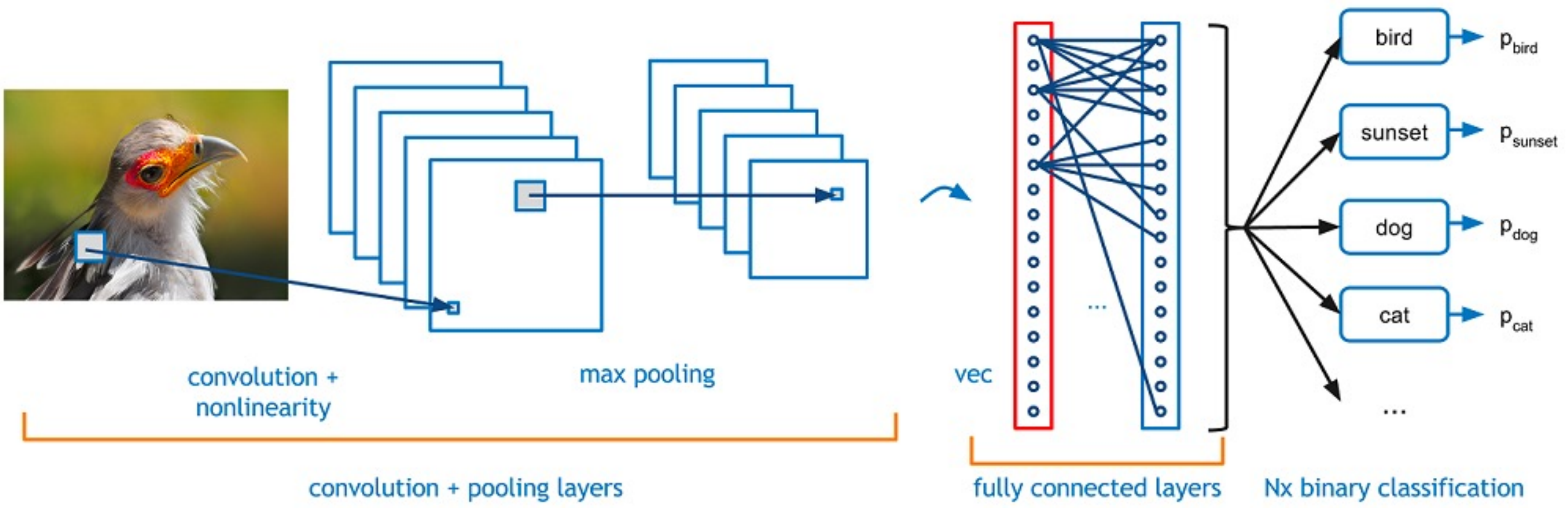
Transformamos a Matriz em Vetor

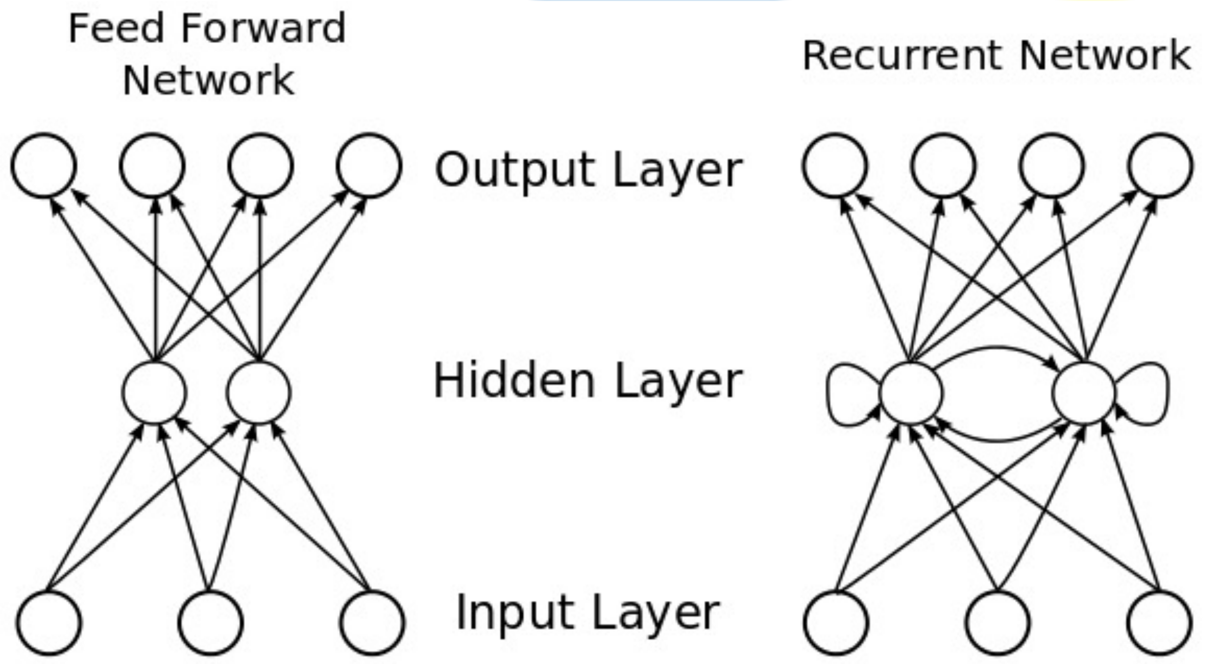


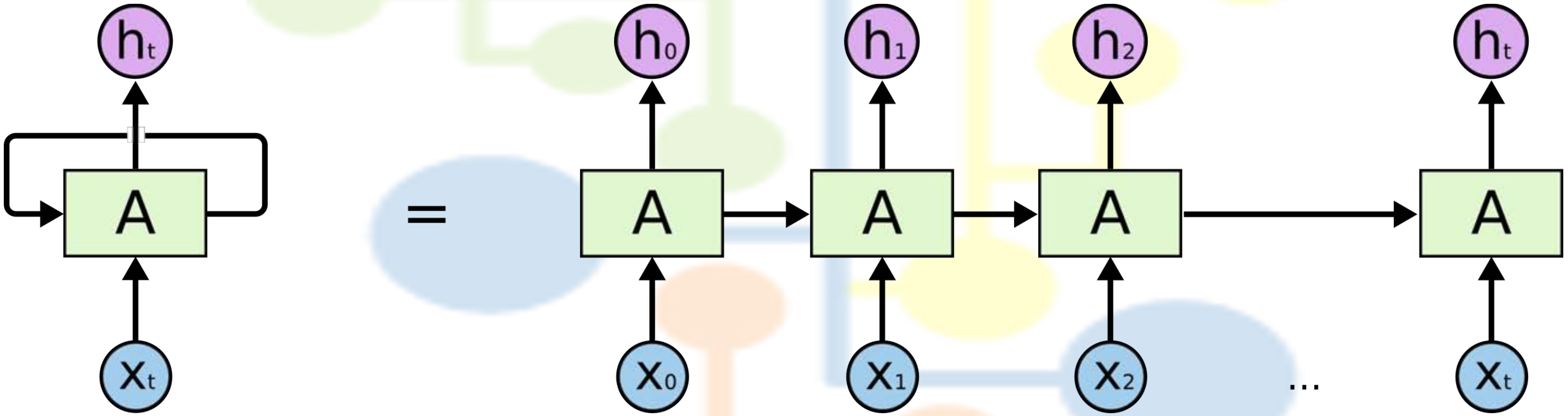

bird
p_{bird}
sunset
p_{sunset}
dog
p_{dog}
cat
p_{cat}
...
convolution + nonlinearity
max pooling
vec
convolution + pooling layers
fully connected layers
Nx binary classification

Data Science Academy

Arquitetura de Redes Neurais Recorrentes

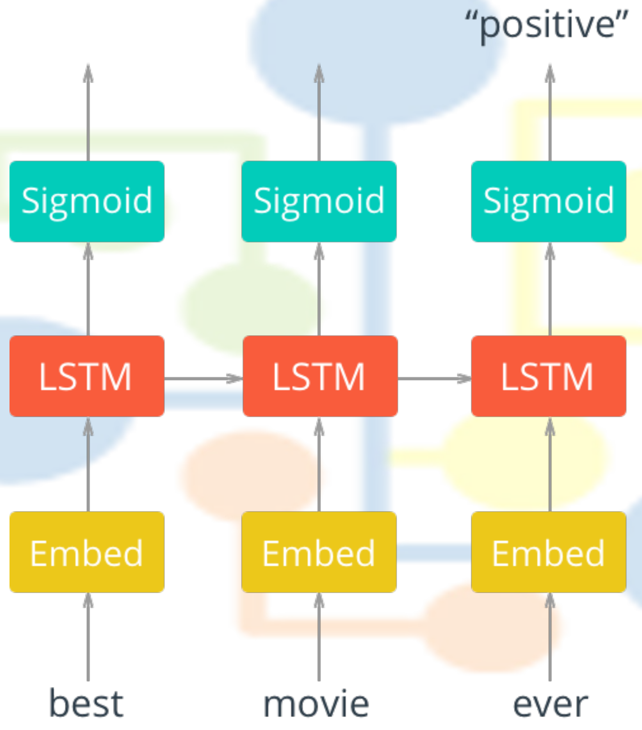

convolution +
nonlinearity
max pooling
vec
...
bird
sunset
dog
cat
...
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + pooling layers
fully connected layers
Nx binary classification

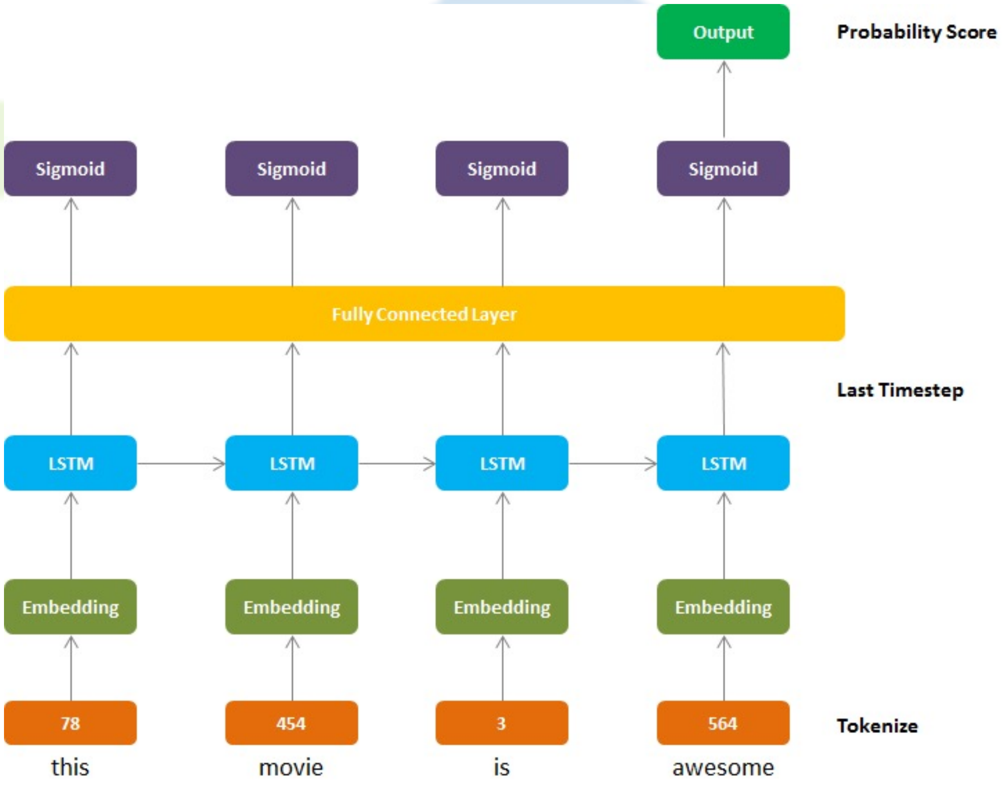

Feed Forward Network
Recurrent Network
Output Layer
Hidden Layer
Input Layer

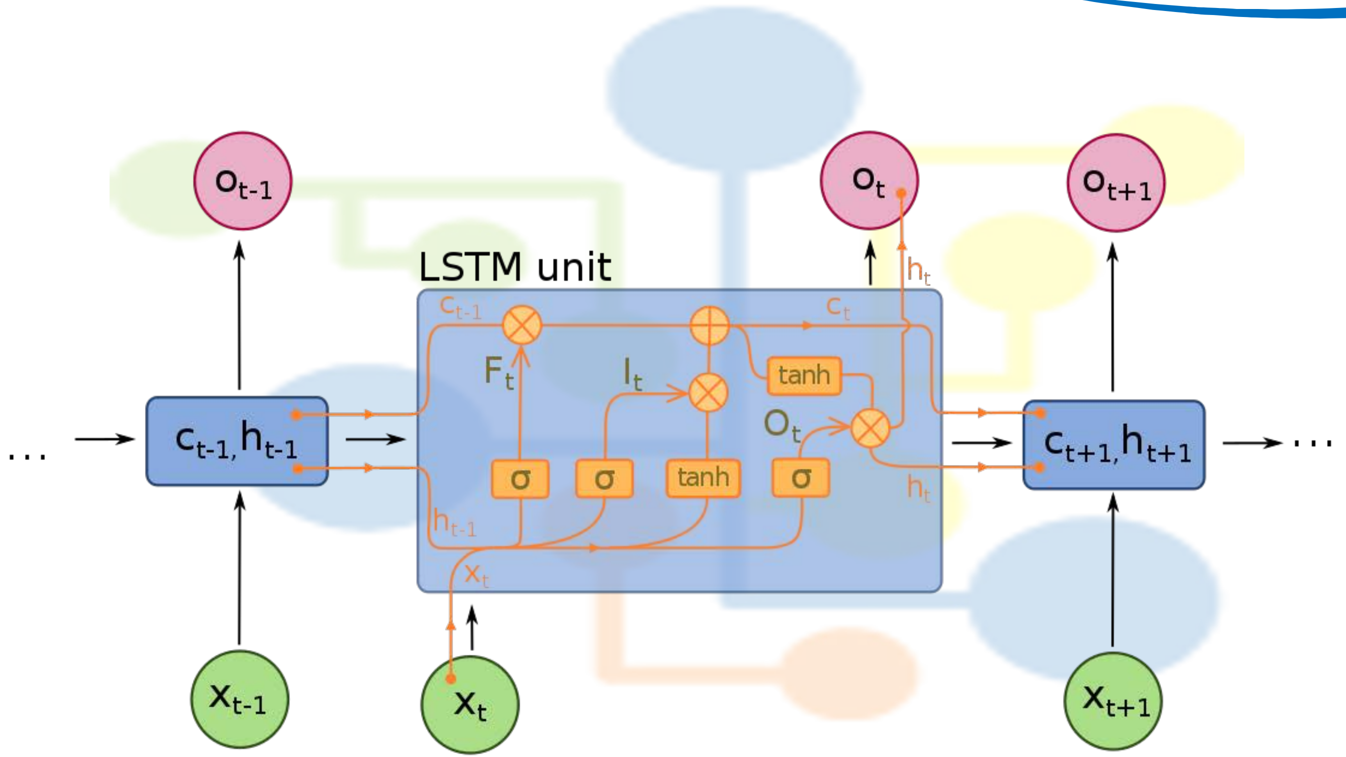

h_t
A
x_t
=
h_0 h_1 h_2 h_t
A A A A
x_0 x_1 x_2 ... x_t

Data Science Academy

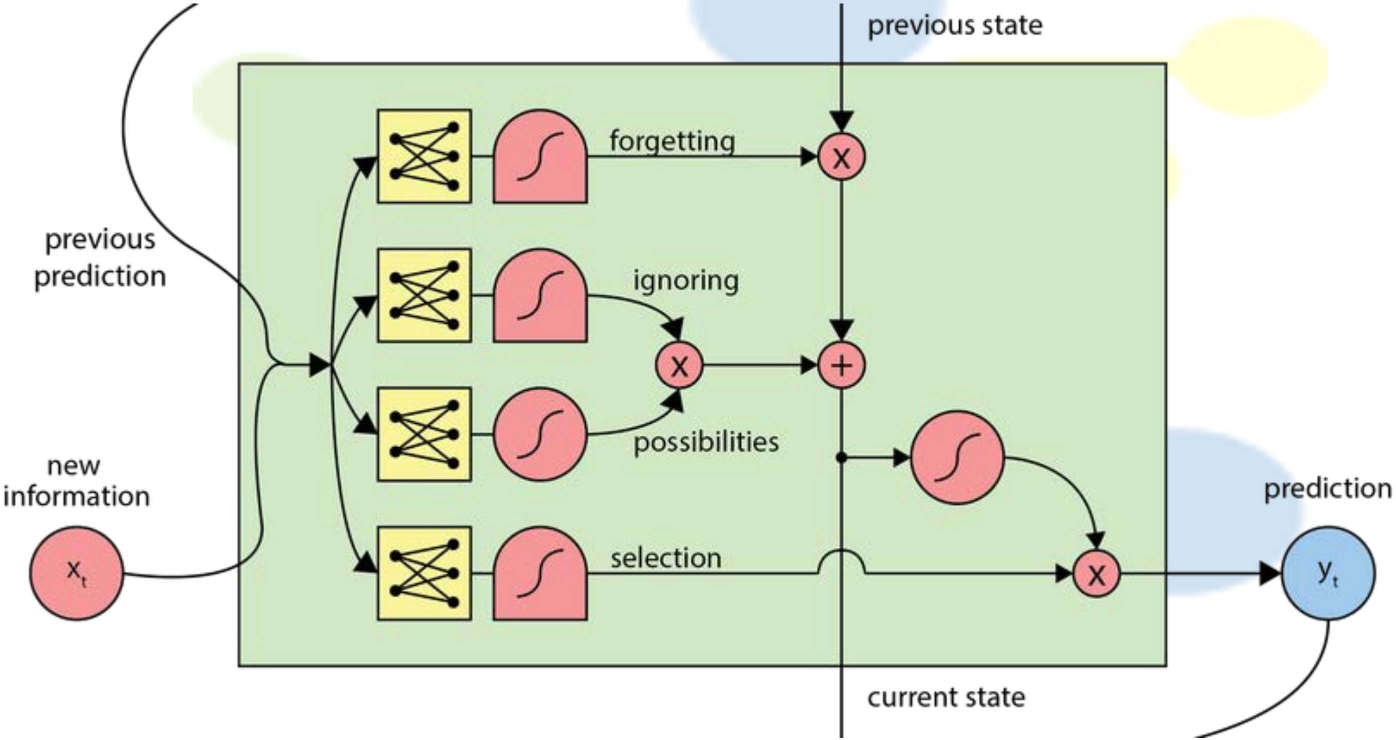
LSTM
Long Short-Term Memory


"positive"
Sigmoid
Sigmoid
Sigmoid
LSTM
LSTM
LSTM
Embed
Embed
Embed
best
movie
ever


Output
Probability Score
Sigmoid
Sigmoid
Sigmoid
Sigmoid
Fully Connected Layer
Last Timestep
LSTM
LSTM
LSTM
LSTM
Embedding
Embedding
Embedding
Embedding
78
454
3
564
Tokenize
this
movie
is
awesome



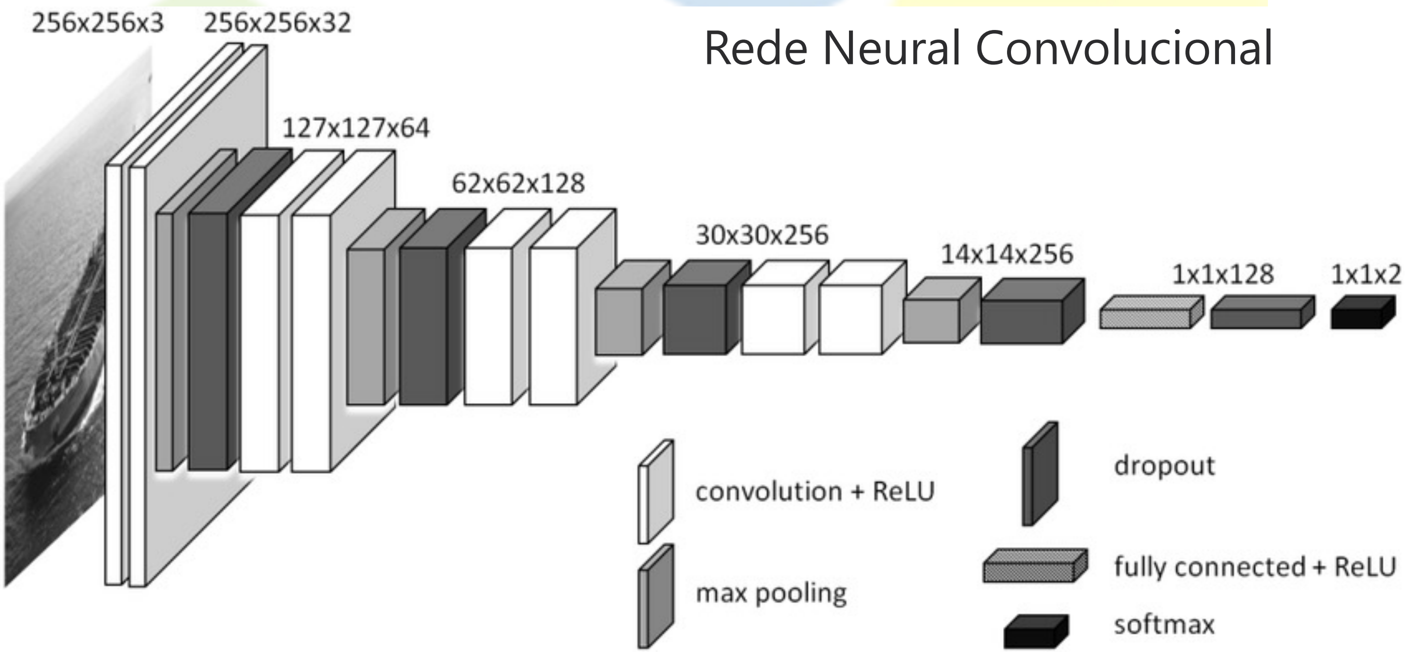
LSTM unit
o_{t-1}
o_t
o_{t+1}
c_{t-1}, h_{t-1}
c_{t+1}, h_{t+1}
x_{t-1}
x_t
x_{t+1}
c_{t-1}
c_t
h_t
h_{t-1}
F_t
I_t
O_t
σ
σ
tanh
σ
tanh
...
...


previous state
forgetting
X
previous prediction
ignoring
X
+
possibilities
new information
x_t
selection
X
prediction
y_t
current state

Data Science Academy

Por Que Usamos a Função Softmax?

Rede Neural Convolucional



Classificador Logístico

Treinamento

$$y = (x \cdot w) + b$$

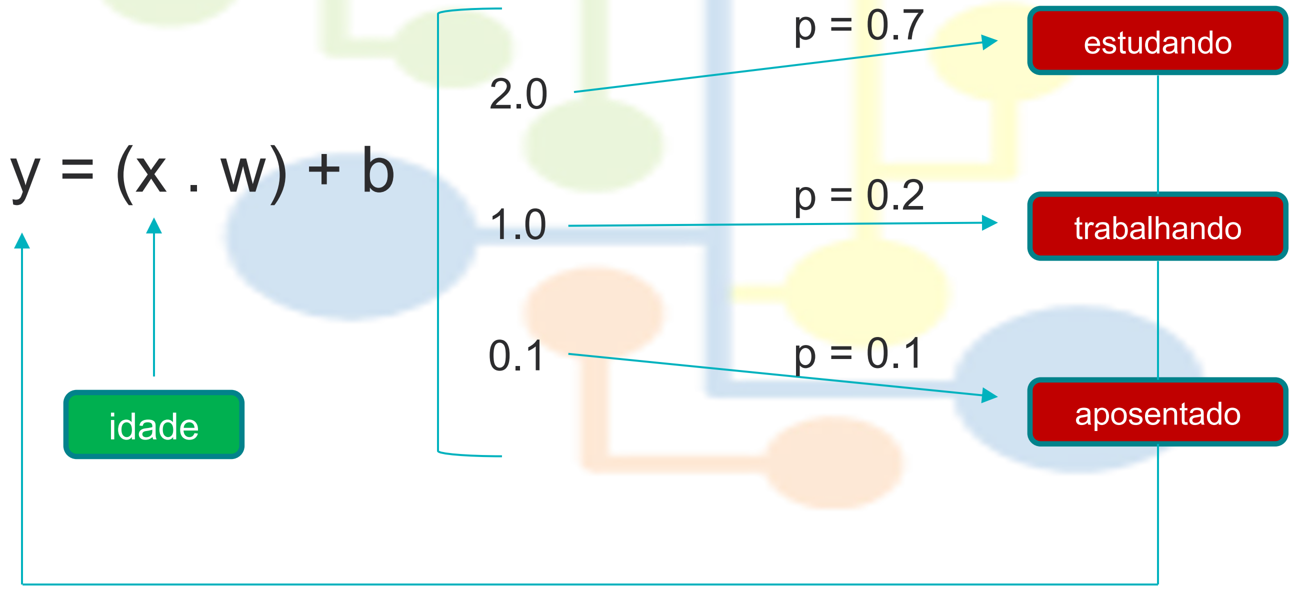
bias

peso

input

output
(previsão)

Classificador Logístico



$$y = (x \,.\, w) + b$$

$$\begin{bmatrix} 2.0 \longrightarrow p = 0.7 \\ 1.0 \longrightarrow p = 0.2 \\ 0.1 \longrightarrow p = 0.1 \end{bmatrix}$$

Função Softmax

$$y = (x \cdot w) + b$$

Scores	Probabilidades
2.0	p = 0.7
1.0	p = 0.2
0.1	p = 0.1

Scores → Probabilidades

Logits

Função Softmax

```
import numpy as np

scores = [3.0, 1.0, 0.2]

def softmax(x):
    return np.exp(x) / np.sum(np.exp(x), axis = 0)

print(softmax(scores))
```

```
[ 0.8360188   0.11314284  0.05083836]
```

Função Softmax

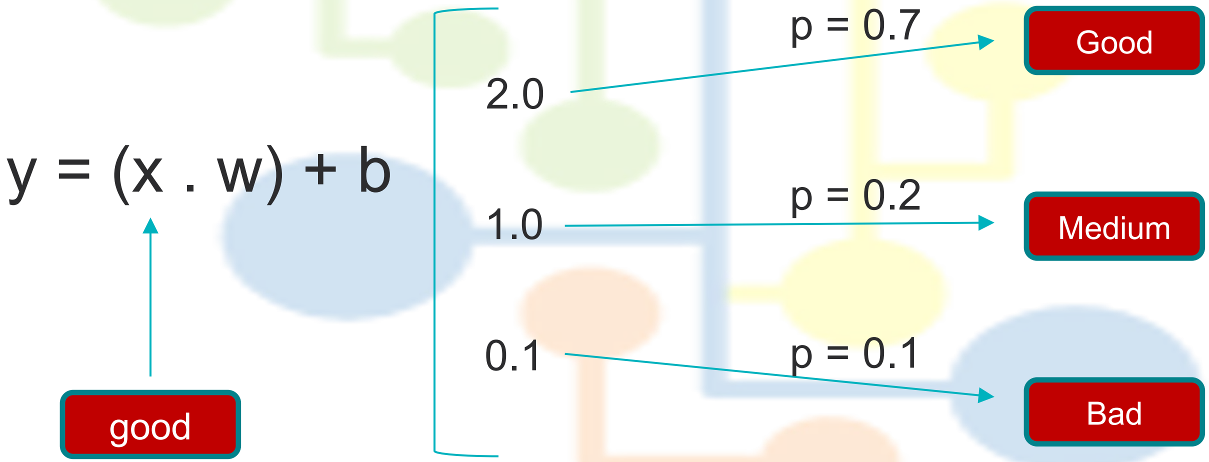
Softmax function

$$P(y=j \mid \Theta^{(i)}) = \frac{e^{\Theta^{(i)}}}{\sum_{j=0}^{k} e^{\Theta_k^{(i)}}}$$

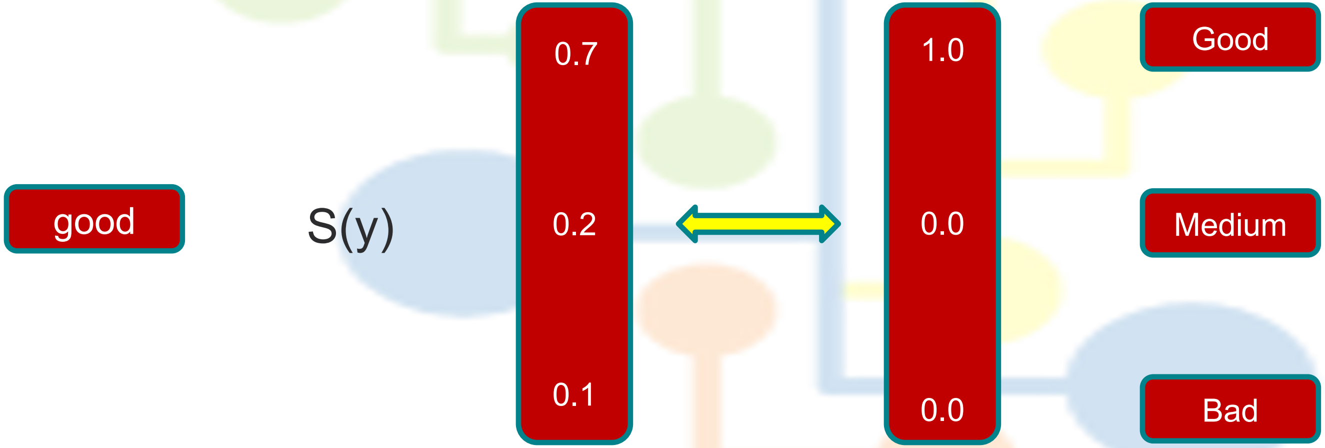
where $\Theta = w_0x_0 + w_1x_1 + \dots + w_kx_k = \sum_{i=0}^{k} w_ix_i = w^Tx$

Data Science Academy

Afinal, O Que é Hot Encoding?


y = (x . w) + b
good
2.0
1.0
0.1
p = 0.7
p = 0.2
p = 0.1
Good
Medium
Bad

Hot Encoding



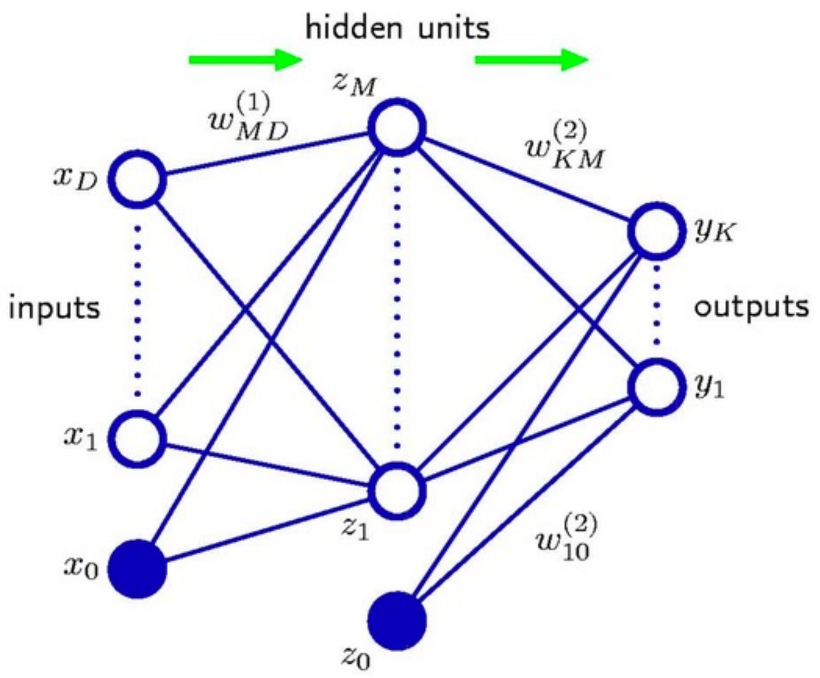
S(y) = Função Softmax que converte os scores para probabilidades

Hot Encoding

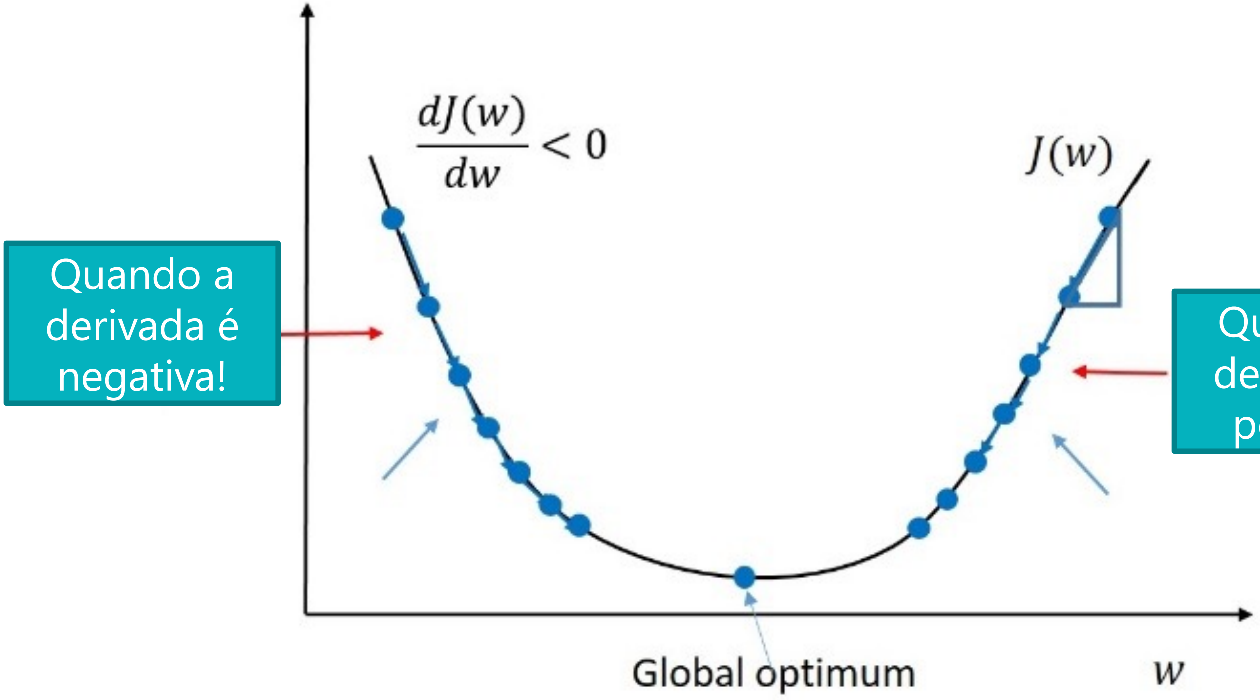
Color
Red
Red
Yellow
Green
Yellow

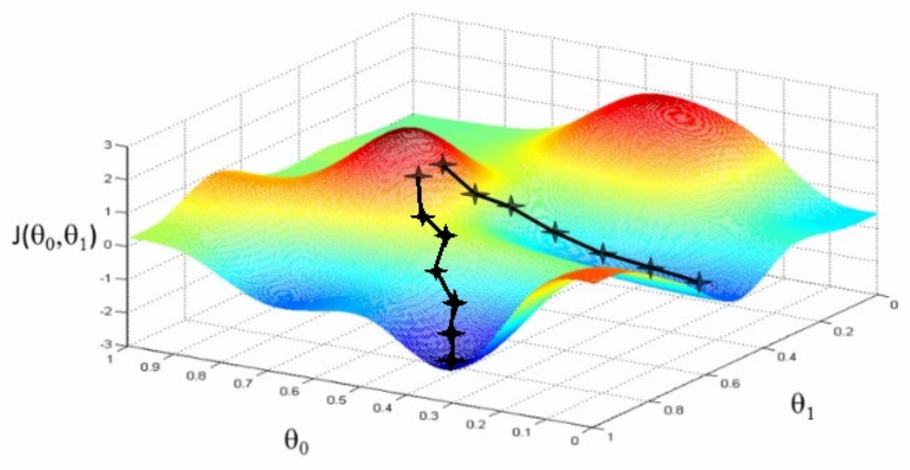
Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Otimização com Stochastic Gradient Descent



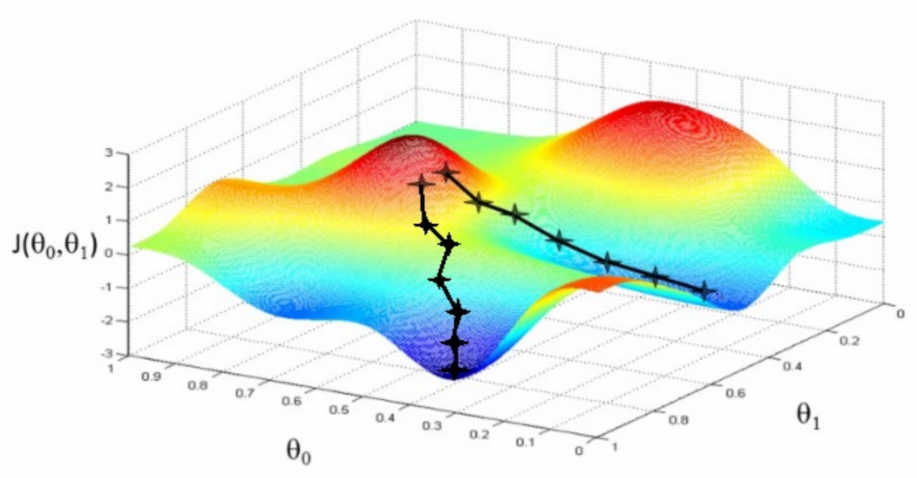
O treinamento de uma rede neural é convertido em um problema de otimização, cujo objetivo é minimizar o erro cometido pela rede, quando considerados todos os exemplos de treinamento.


$\frac{dJ(w)}{dw} < 0$
$J(w)$
Quando a derivada é negativa!
Quando a derivada é positiva!
Global optimum
w

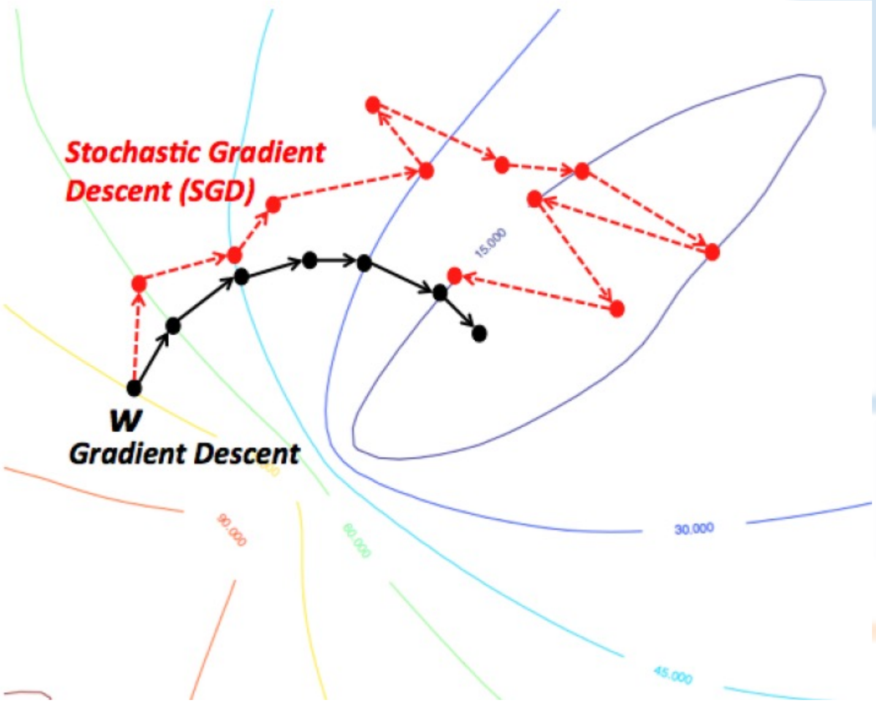


$J(f(x (t) ; \theta), y (t))$

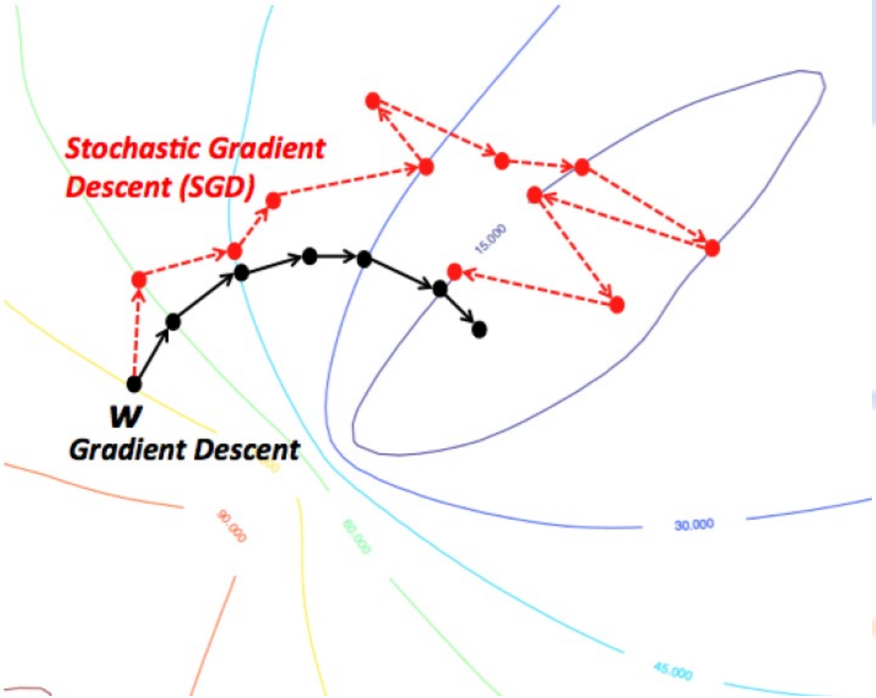
O gradiente de uma função f mede o quanto f varia uma vez que seus argumentos são alterados. Se f for uma função multivariada de n variáveis, então ∇ f é um vetor n-dimensional cujas componentes são as derivadas parciais de f.



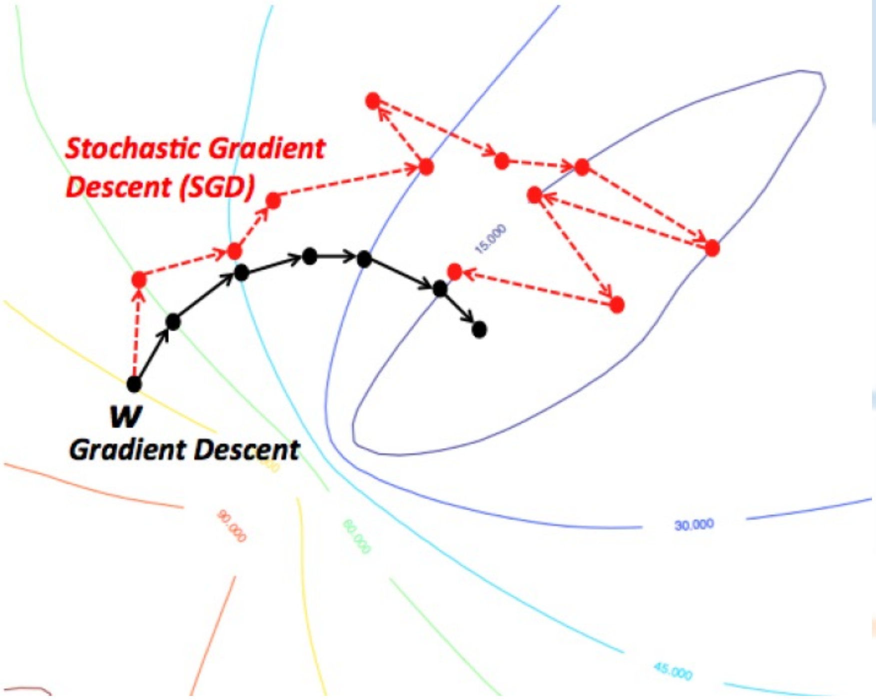
Além de ser computacionalmente intensivo, com Gradient Descent você precisa calcular o gradiente de cada elemento do seu conjunto de treinamento, o que pode levar muito tempo em grandes conjuntos de dados.



A solução encontrada para esse problema, foi o Stochastic Gradient Descent (SGD) que é uma versão do Gradient Descent, em que trabalhamos com amostras aleatórias.

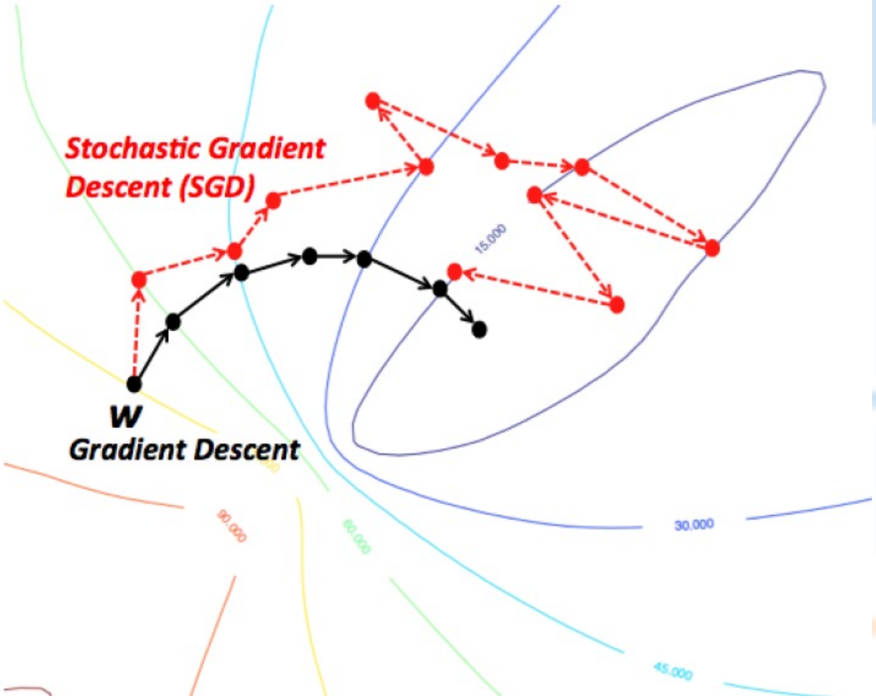


SGD é uma aproximação de Gradient Descent e quanto mais lotes processados pela rede neural (ou seja, mais amostra aleatórias), melhor a aproximação.



A implementação do SGD compreende:

1. Amostragem aleatória de um lote de dados do conjunto de dados total.
2. Executar a rede para frente e para trás para calcular o gradiente (com dados gerados no item 1).
3. Aplicar a atualização de descida de gradiente.
4. Repetir os passos 1 a 3 até que a convergência ou o ciclo seja interrompido por outro mecanismo, ou seja, o número de épocas (epochs).



A técnica de Stochastic Gradient Descent está no cerne do Deep Learning.

Isso ocorre porque o SGD se equilibra bem com os dados e o tamanho do modelo, e queremos trabalhar com Big Data e modelos com muitas camadas ocultas.

Continue Trilhando uma Excelente Jornada de Aprendizagem!