Fundamentos das redes neurais

Prof. Dr. Ricardo Primi

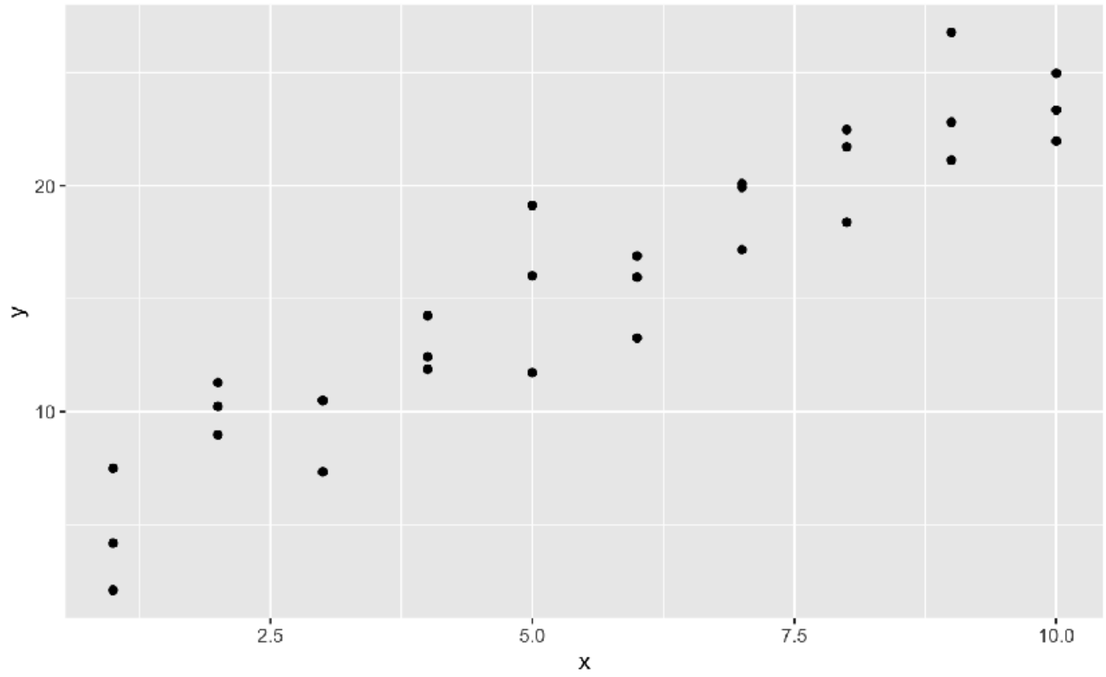
Tópicos

- PC e SVD
  - Aprendizagem não supervisionada
  - Encontrar padrões nos dados
- Fundamentos de processamento de textos
- Modelo preditivo (regressão)
  - Aprendizagem supervisionada
- Redes neurais

Modelagem com regressão

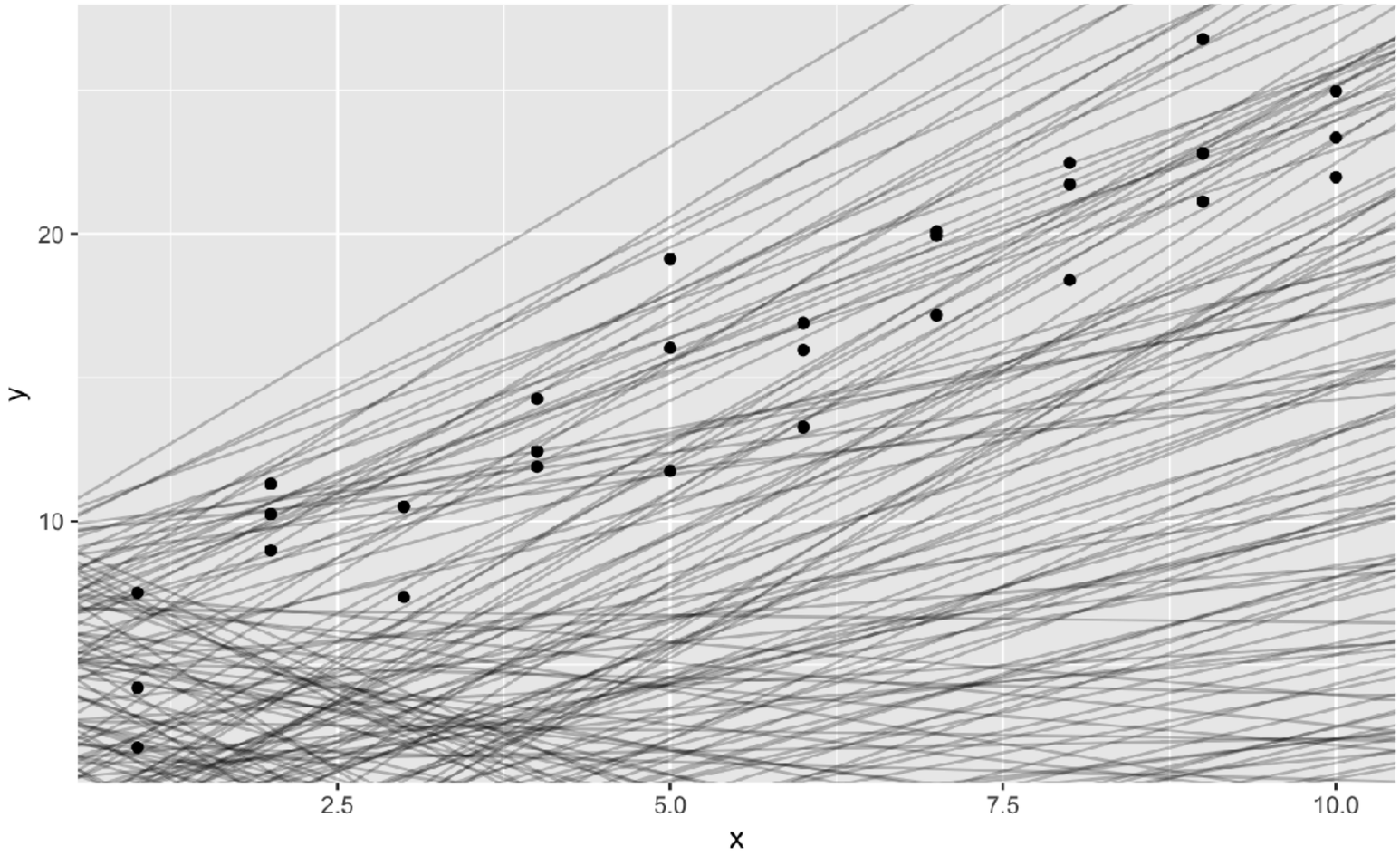
https://r4ds.had.co.nz/model-basics.html

$y = a1*x + a2$



Variando a1 e a2

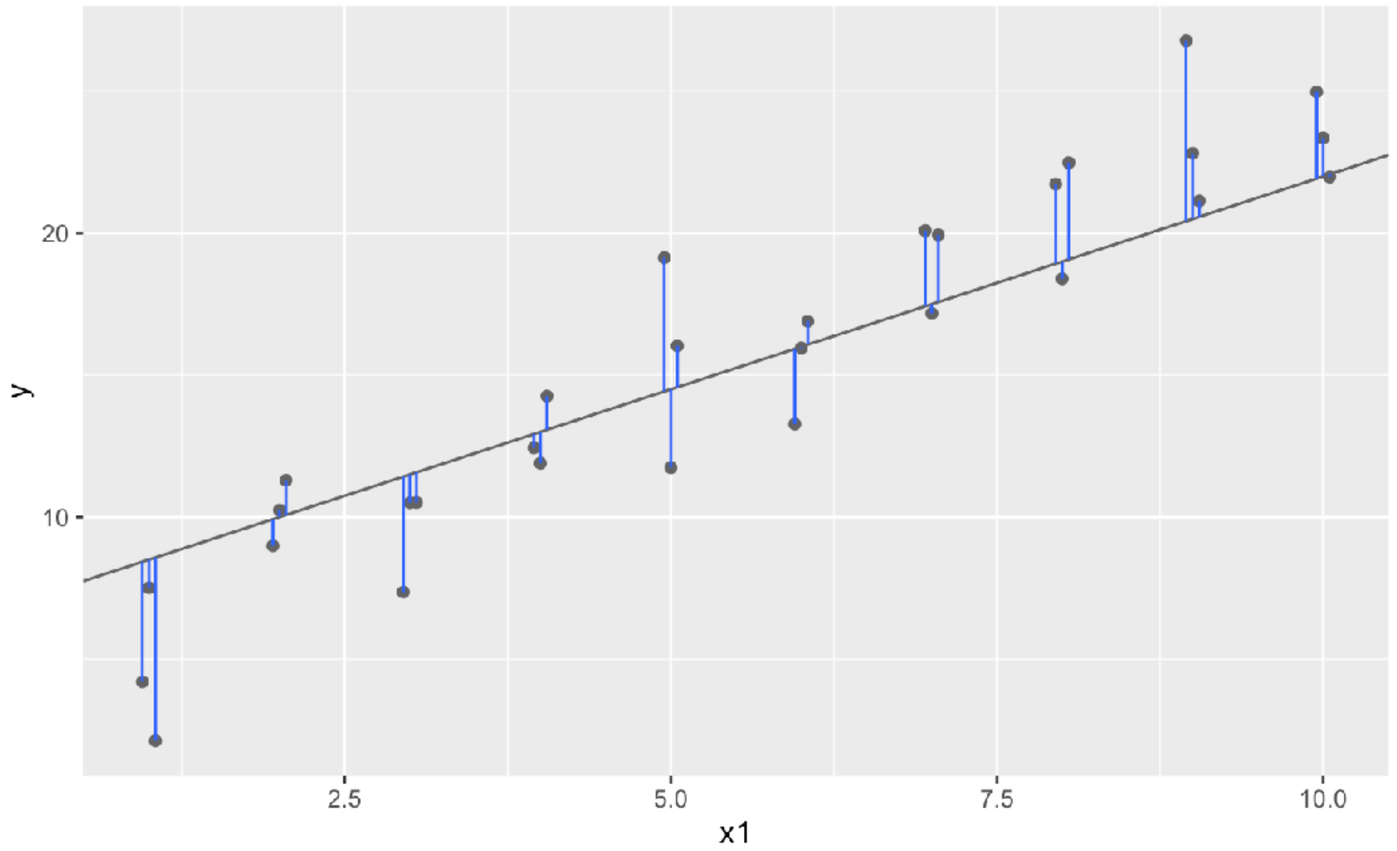
a1 = runif(250, -10, 10), a2 = runif(250, -3, 3)



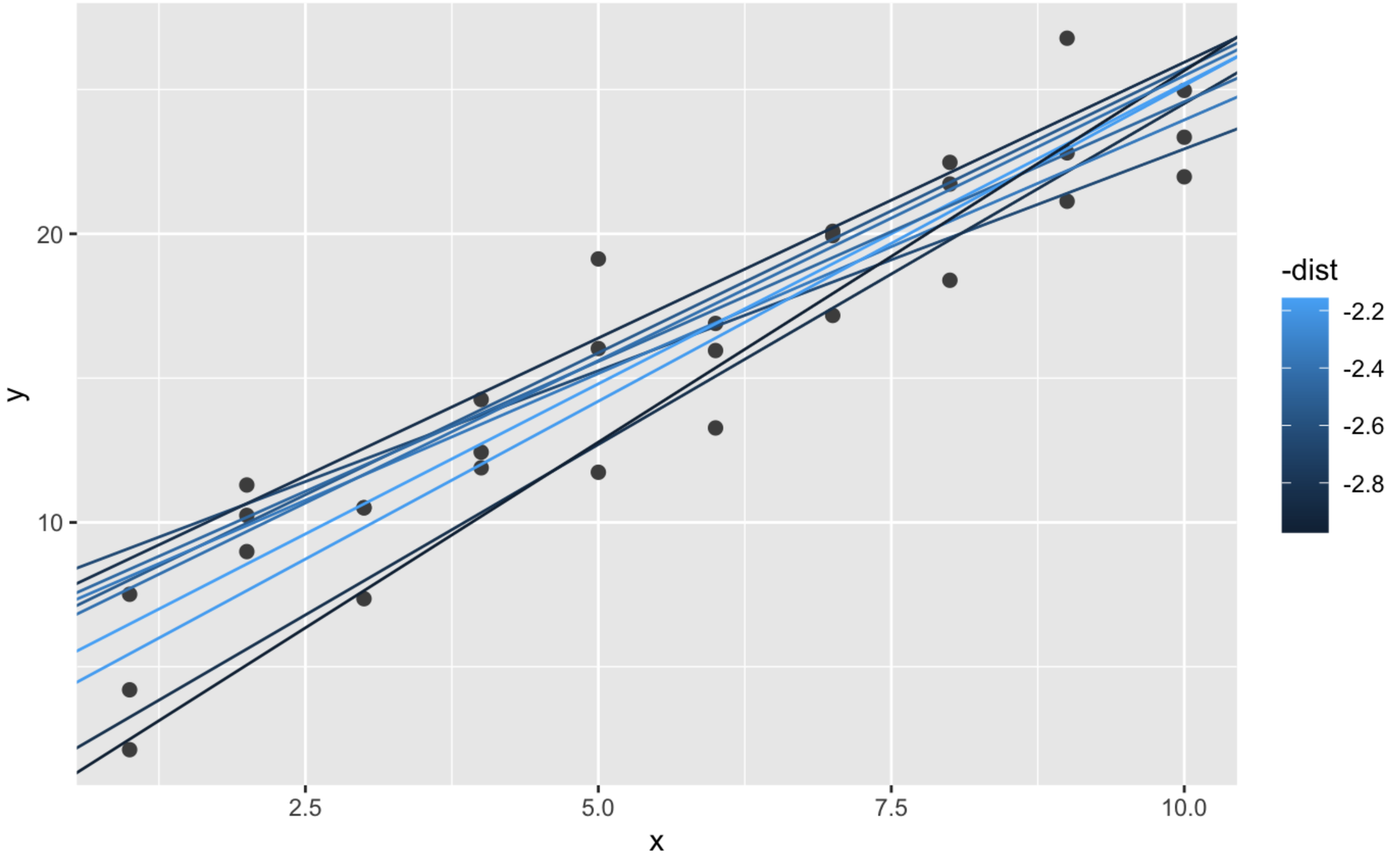
Entendendo o modelo linear

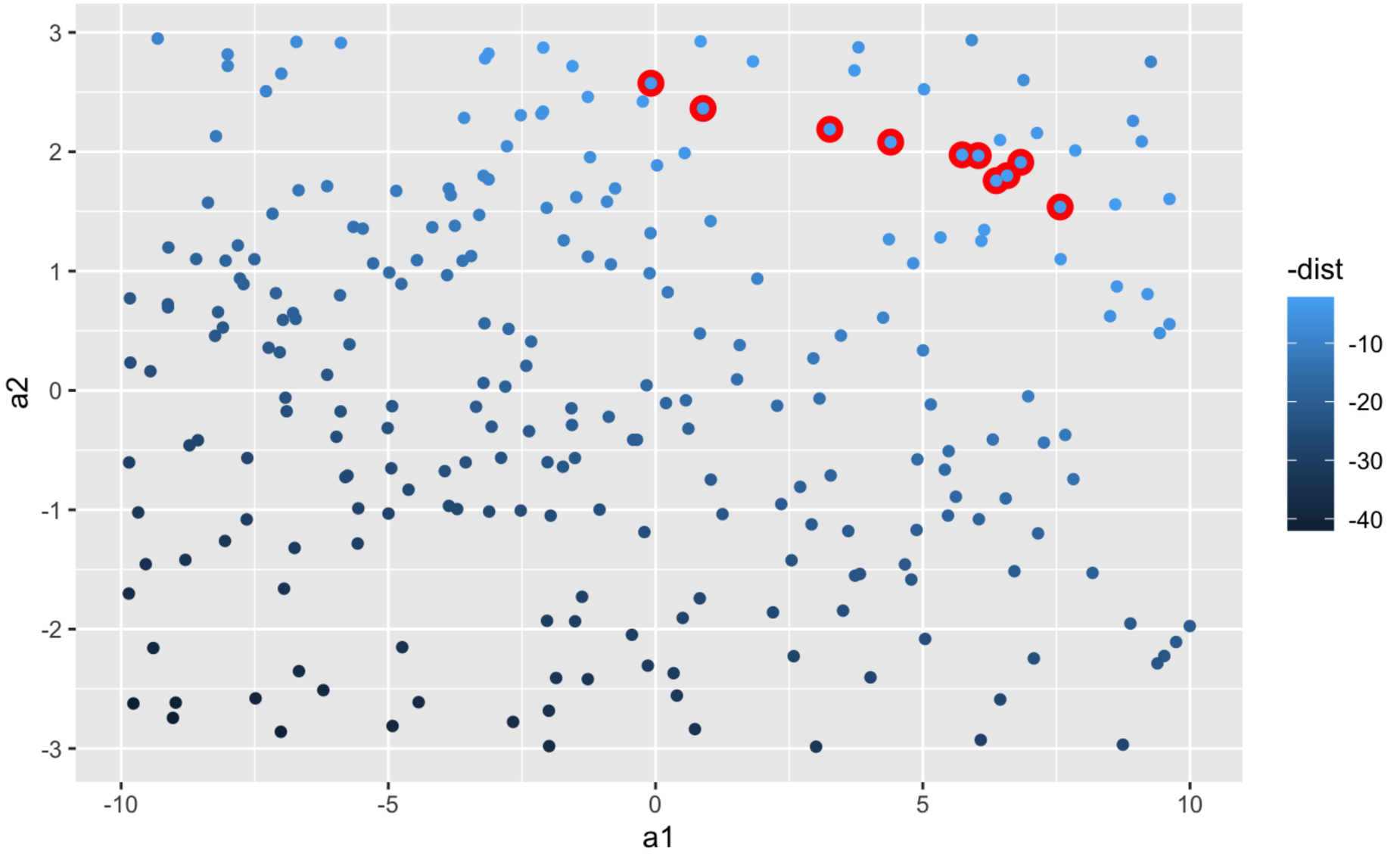
y = ax + b

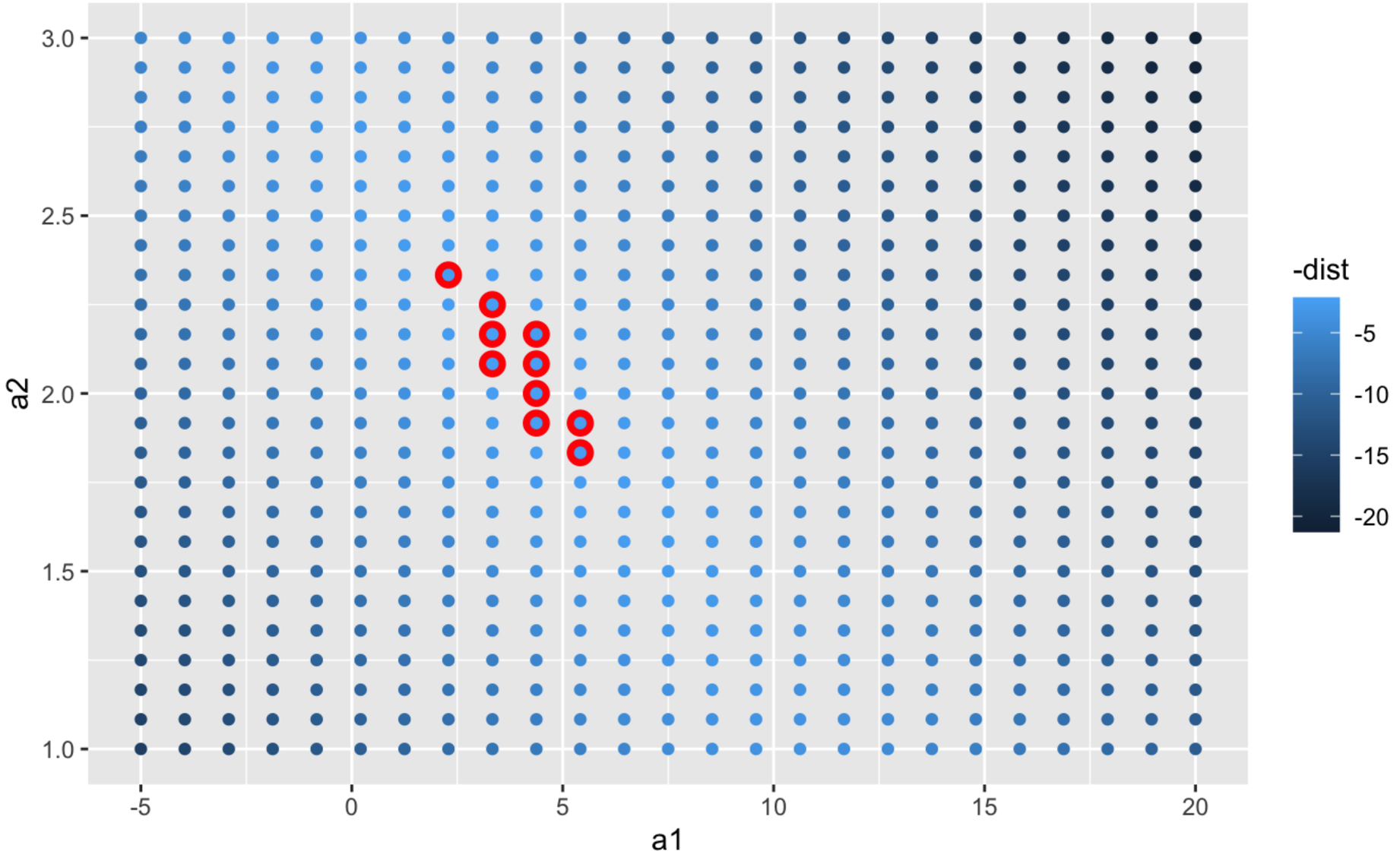
https://www.desmos.com



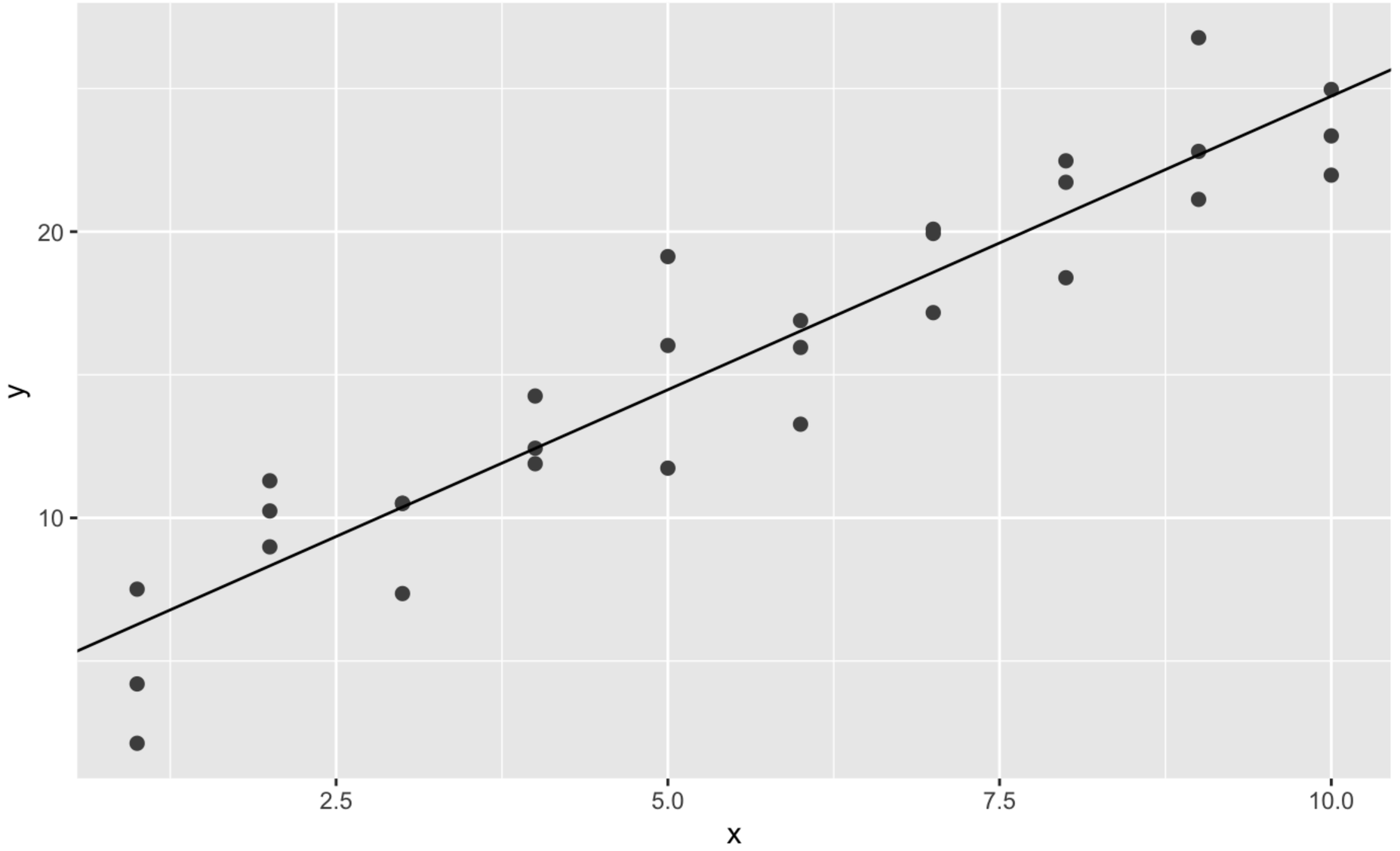
- Escolha um modelo (valores de a1 e a2)
- Calcule o valor predito usando a fórmula
- Calcule o erro entre o valor real e o valor predito para todos os valores

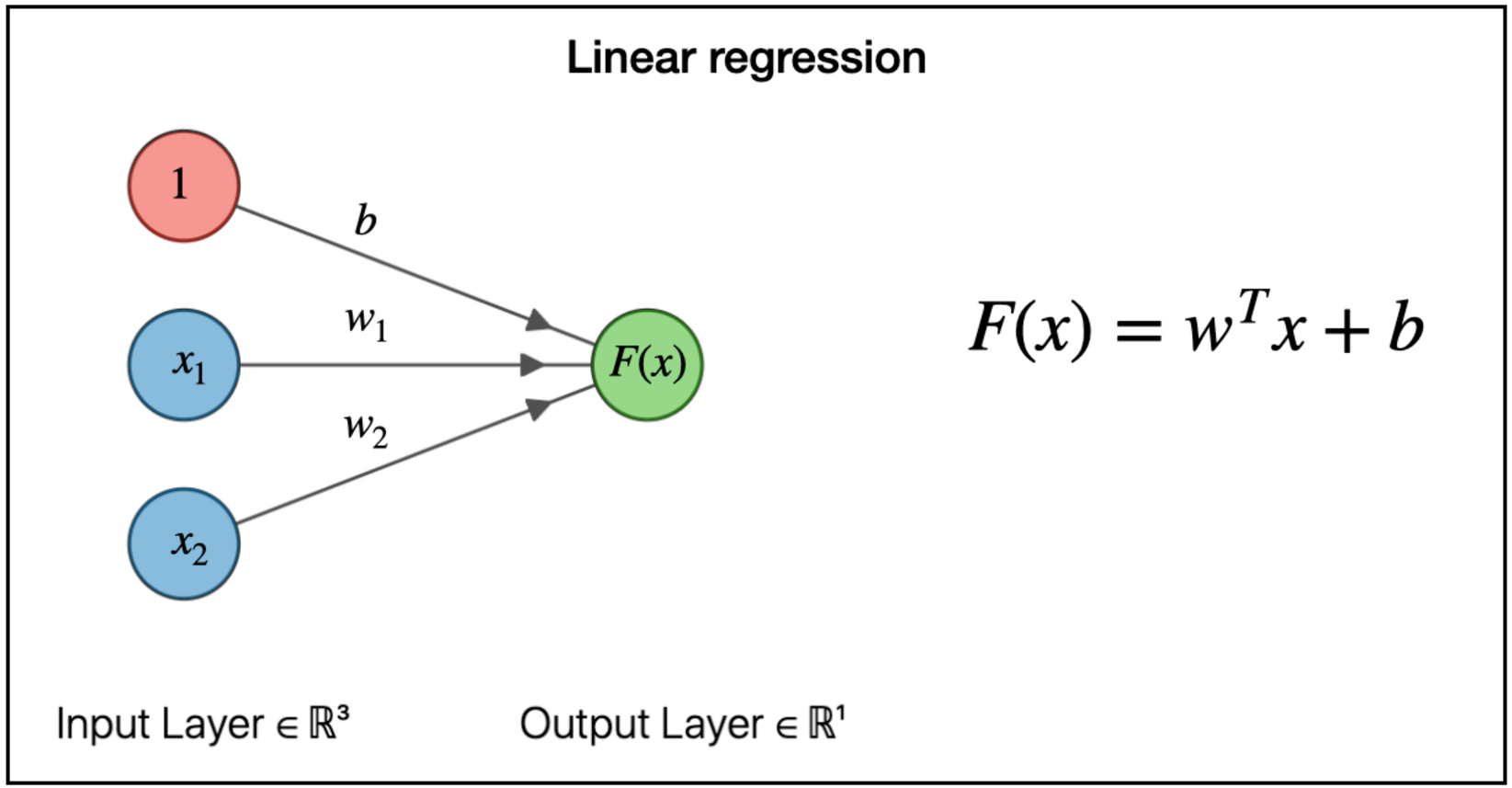

20
10
y
2.5
5.0
7.5
10.0
x
-dist
-2.2
-2.4
-2.6
-2.8


3
2
1
0
-1
-2
-3
a2
-10
-5
0
5
10
a1
-dist
-10
-20
-30
-40


3.0
2.5
2.0
1.5
1.0
a2
-5
0
5
10
15
20
a1
-dist
-5
-10
-15
-20

y = 2,05*x + 4.22





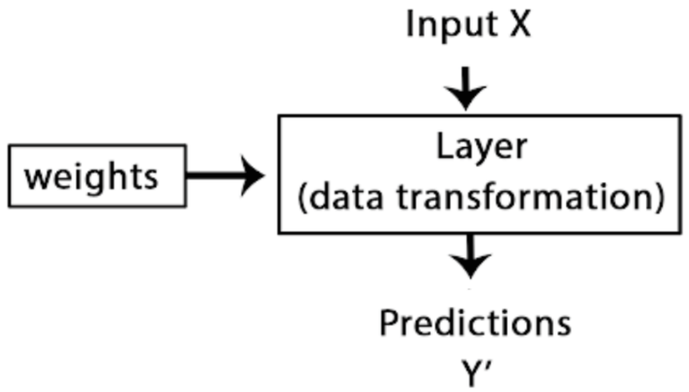
$$F(x) = w^T x + b$$

$$F(x) = w_1 \cdot x_1 + w_2 \cdot x_2 + 1 \cdot b$$

- Como aplicamos isso na nossa matriz de textos DFM ?

Abstraindo

Regressão Linear



Fonte: Figura adaptada do Deep Learning with R, Chollet, F. et al.

Objetivo: encontrar os melhores 'pesos' para essa Layer.

Definição do modelo

Definimos o modelo de regressão linear da seguinte forma:

$$\hat{y}_i = w \times x_i + b$$

- y_i: preço do imóvel i
- x_i: área do imóvel i

Poderíamos escrever

$$\hat{y}_i = f(x_i)$$

em que:

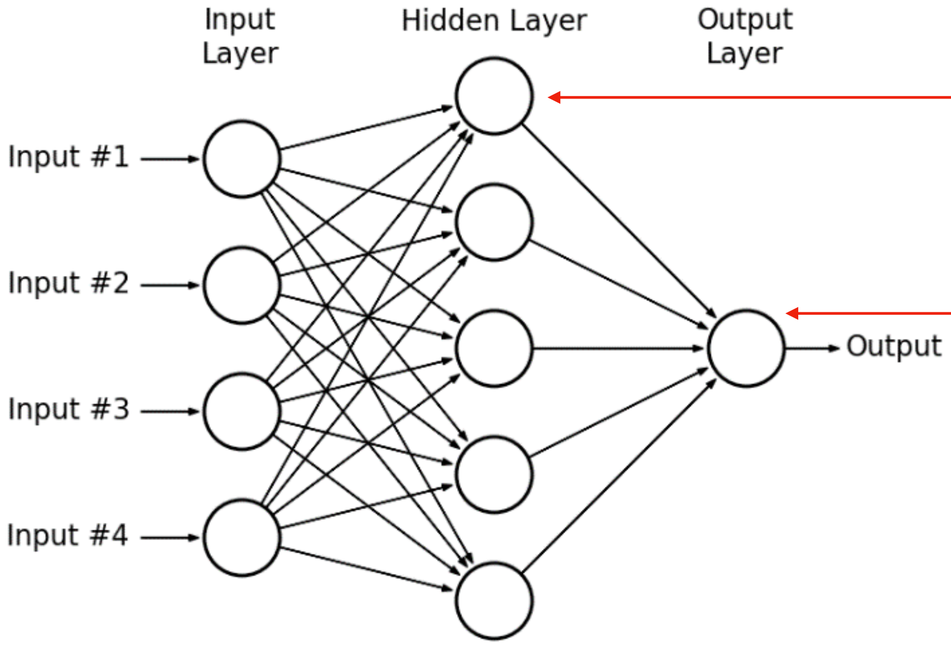
- $f(x) = w \times x + b$

Chamamos f de **'layer'** (camada) na linguagem do Deep learning.

fonte: https://curso-r.github.io/202005-deep-learning/

Deep Learning

Várias regressões múltiplas em camadas
Adicionando-se uma transformação não linear na saída

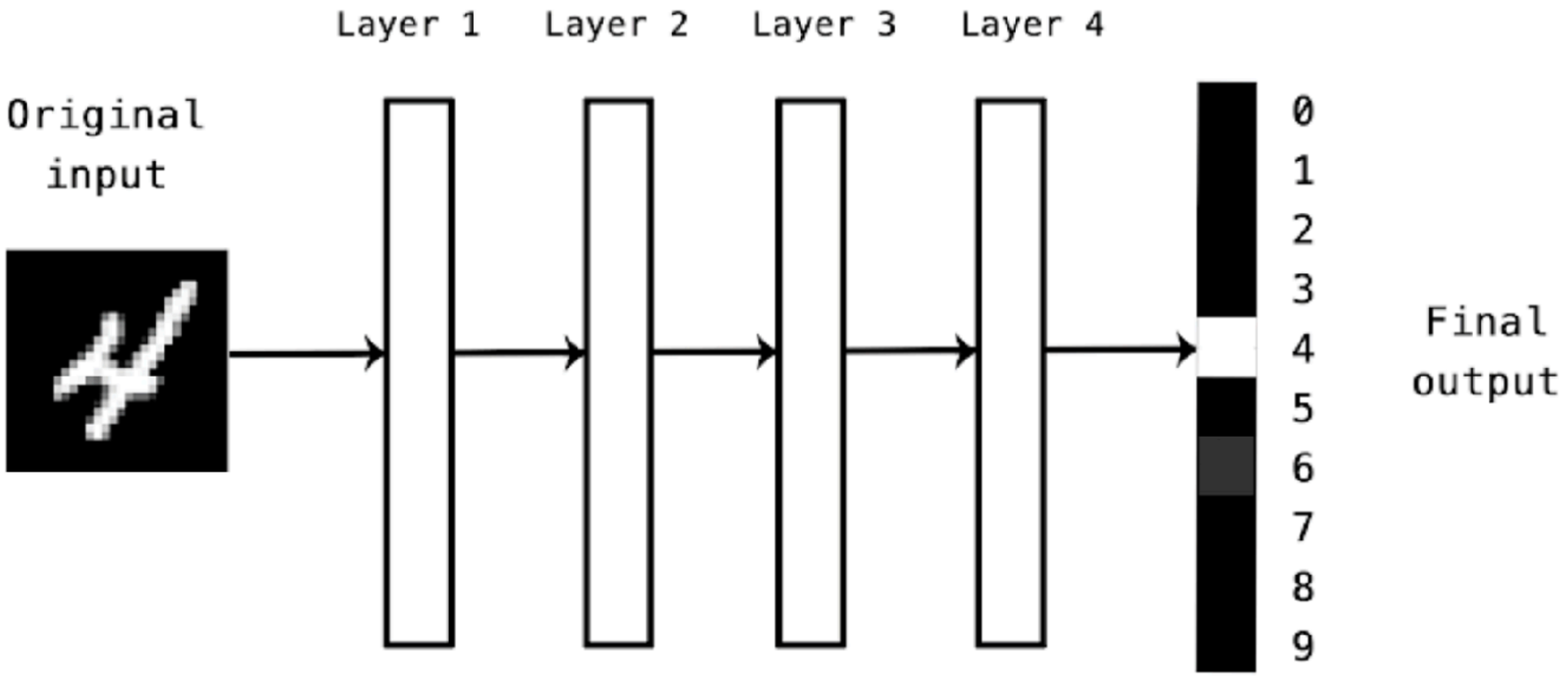


Camada 1 / Layer 1:
Uma regressão

Camada 2 / Layer 2:
Uma regressão tendo com entrada o resultado de várias regressões

Fonte: https://www.researchgate.net/figure/A-hypothetical-example-of-Multilayer-Perceptron-Network_fig4_303875065

Deep Learning



Abstraindo

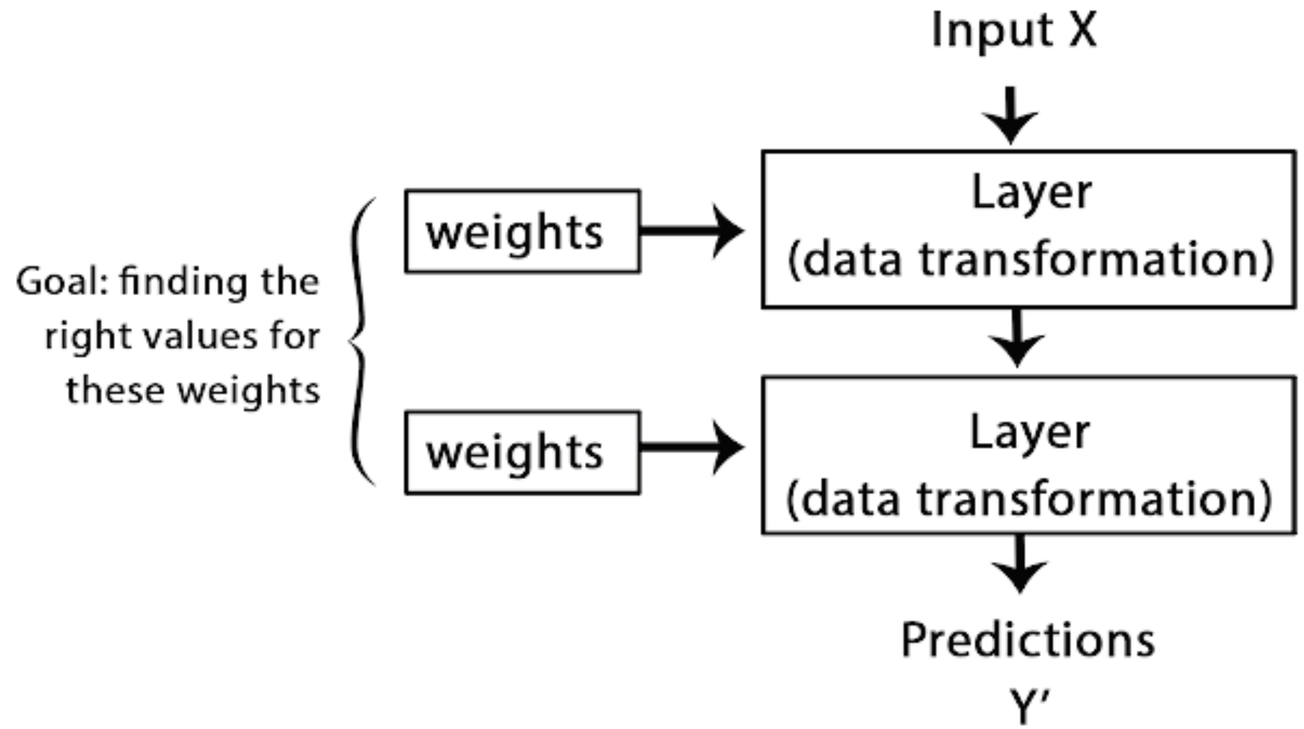


Figure 1.7 A neural network is parametrized by its weights.

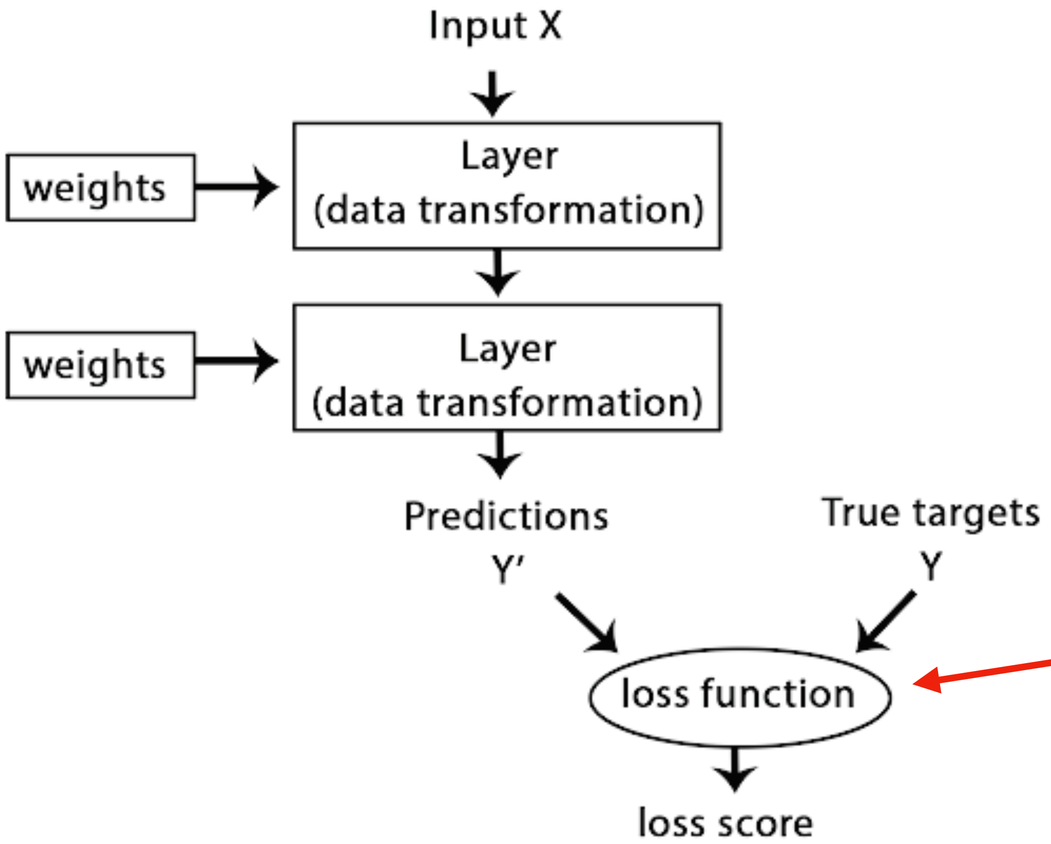


Figure 1.8 A loss function measures the quality of the network's output.

Função de erro

Lost function, função de perda: quando vc perde em cada aposta em uma combinação de parâmetros

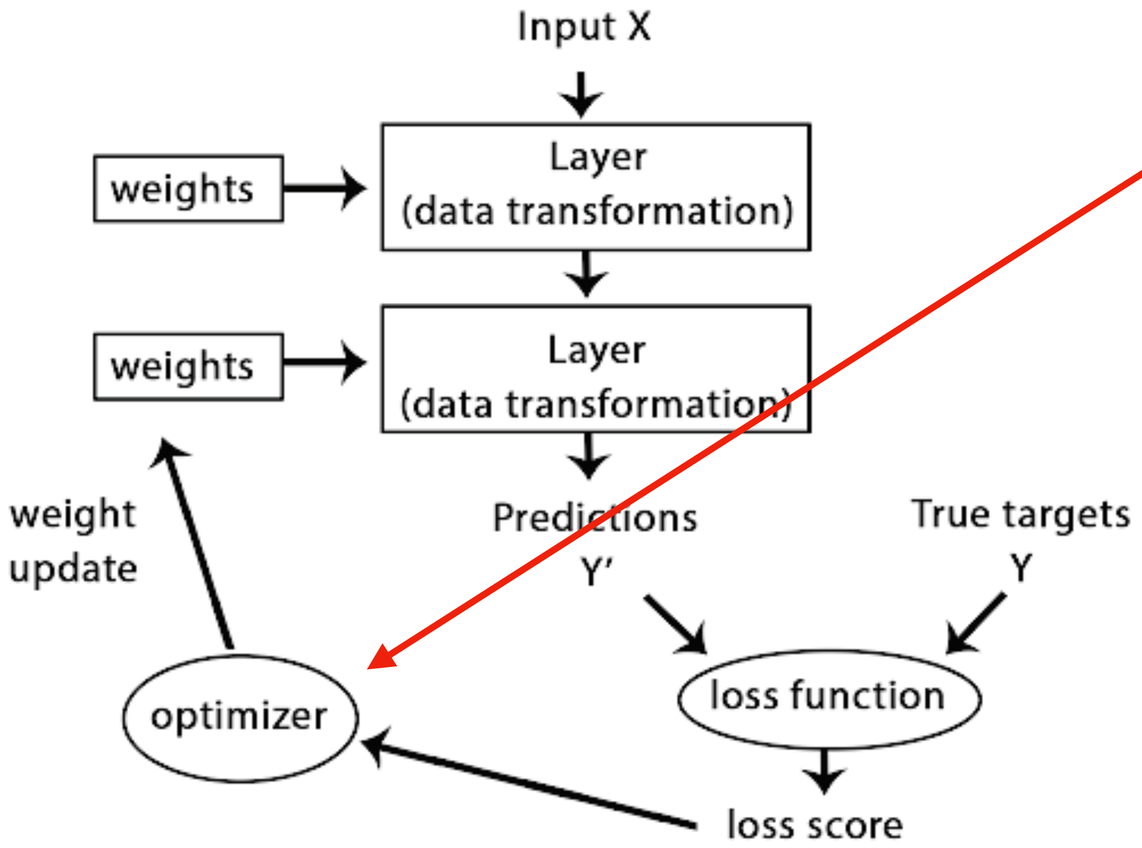
Mede quanto o modelo está perto do que queremos que ele fique.

Uma função de perda bastante usada é o Mean Squared Error ou Erro quadrático médio:

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

Substituindo o valor predito:

$$L(w,b) = \frac{1}{n}\sum_{i=1}^{n}(y_i - w \times x_i - b)^2$$



Algoritmo que procura valores de *w* que minimizam o erro (estão associados ao menor valor da função de perda)

- Os algoritmos são variações do “Gradient descent” (média das mudanças em w)

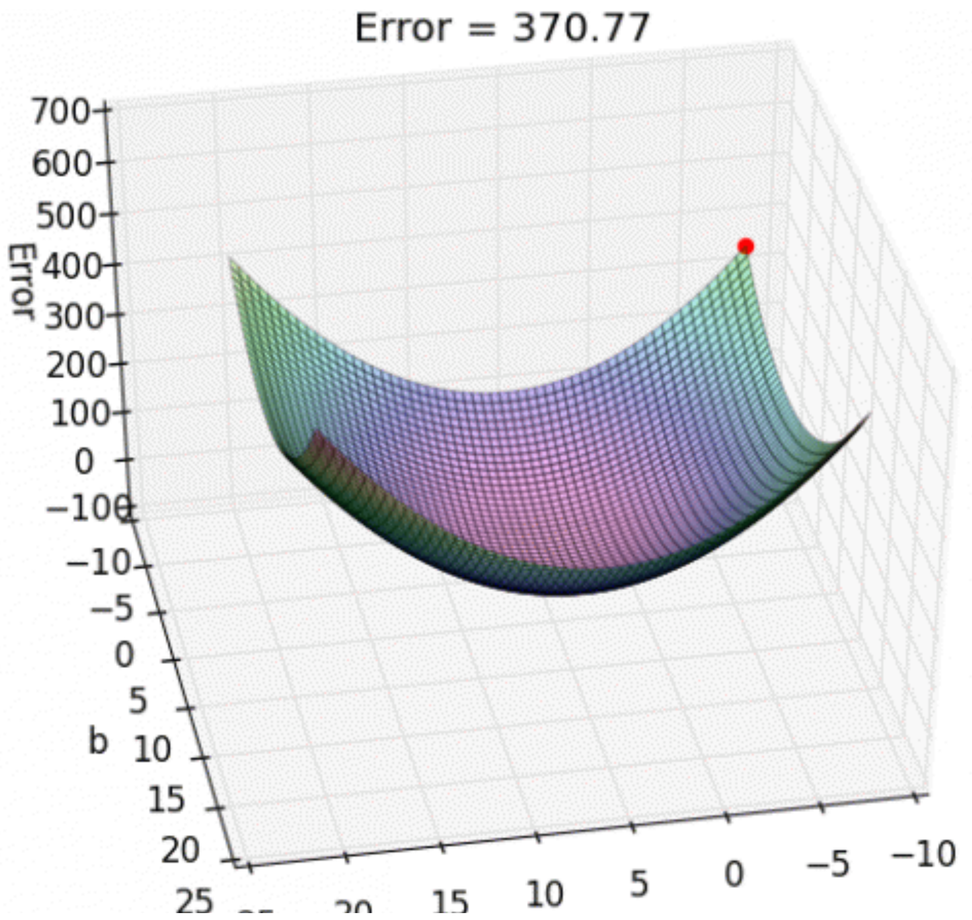
- SGD: "Stocastic Gradient Descent" (mudança em um w)

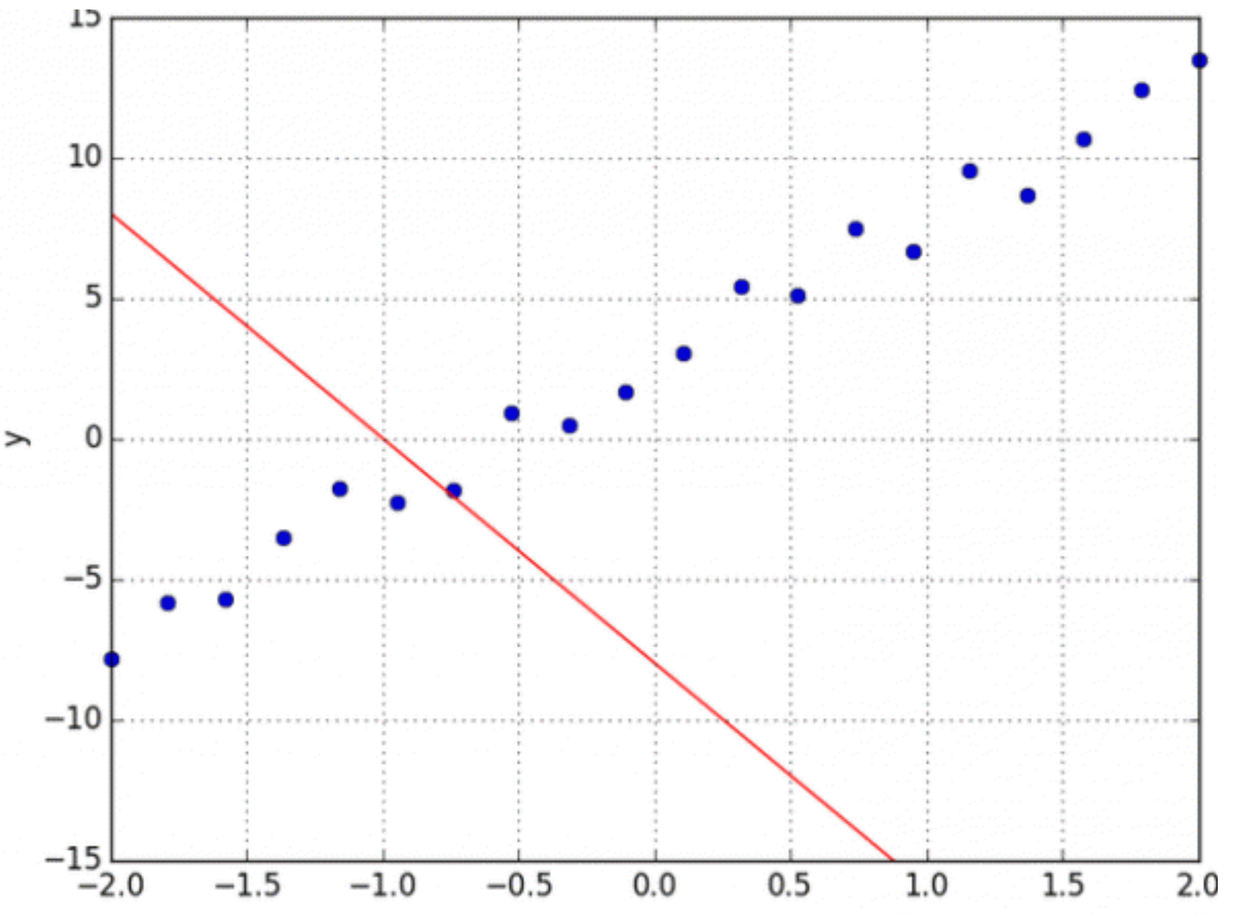
- α: learning rate (magnitude das mudanças de *w* a cada passo)

- Mini Batch SGD

- Epoch

Figure 1.9 The loss score is used as a feedback signal to adjust the weights.





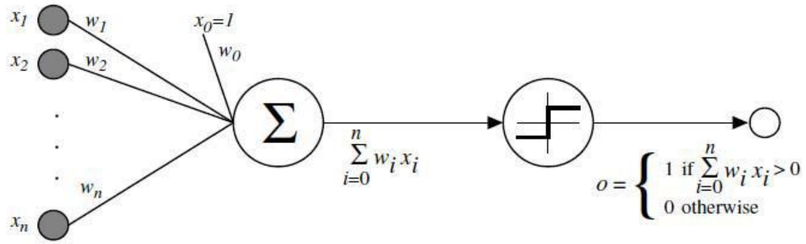
https://alykhantejani.github.io/images/gradient_descent_line_graph.gif

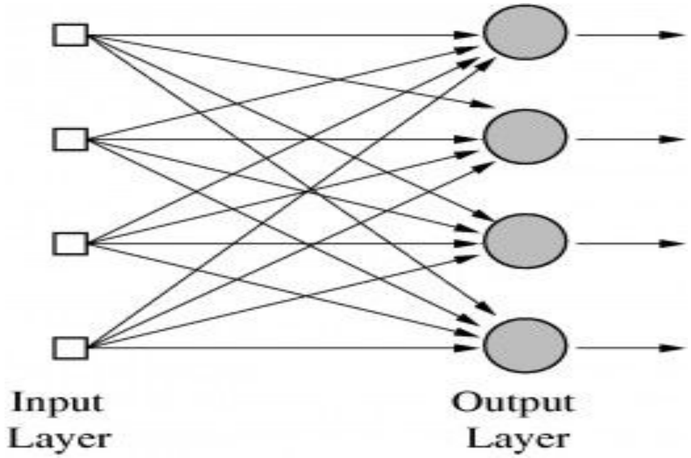
Exercício

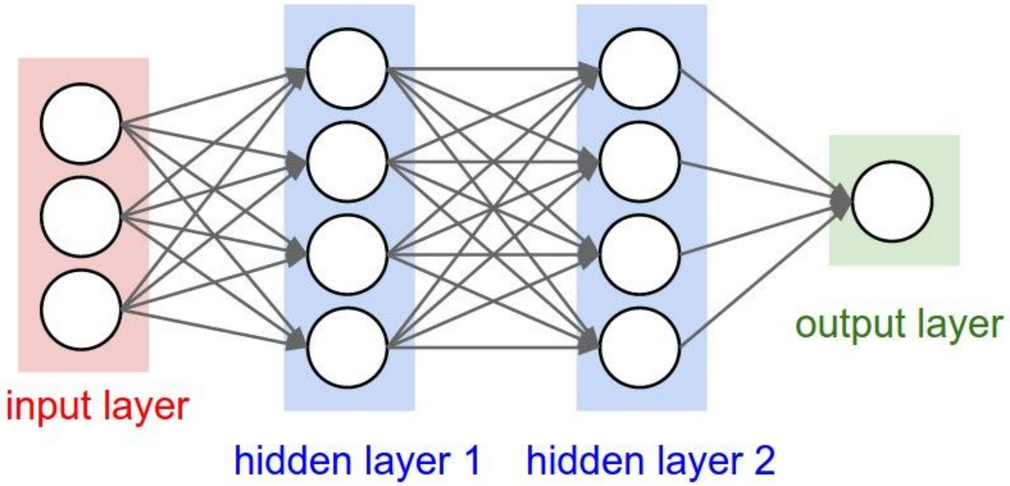
Qual a diferença entre a função linear de regressão e a função de perda ?

Quais inputs e quais outputs dessa função ?

Perceptron (1957) e Rede Neural Artificial

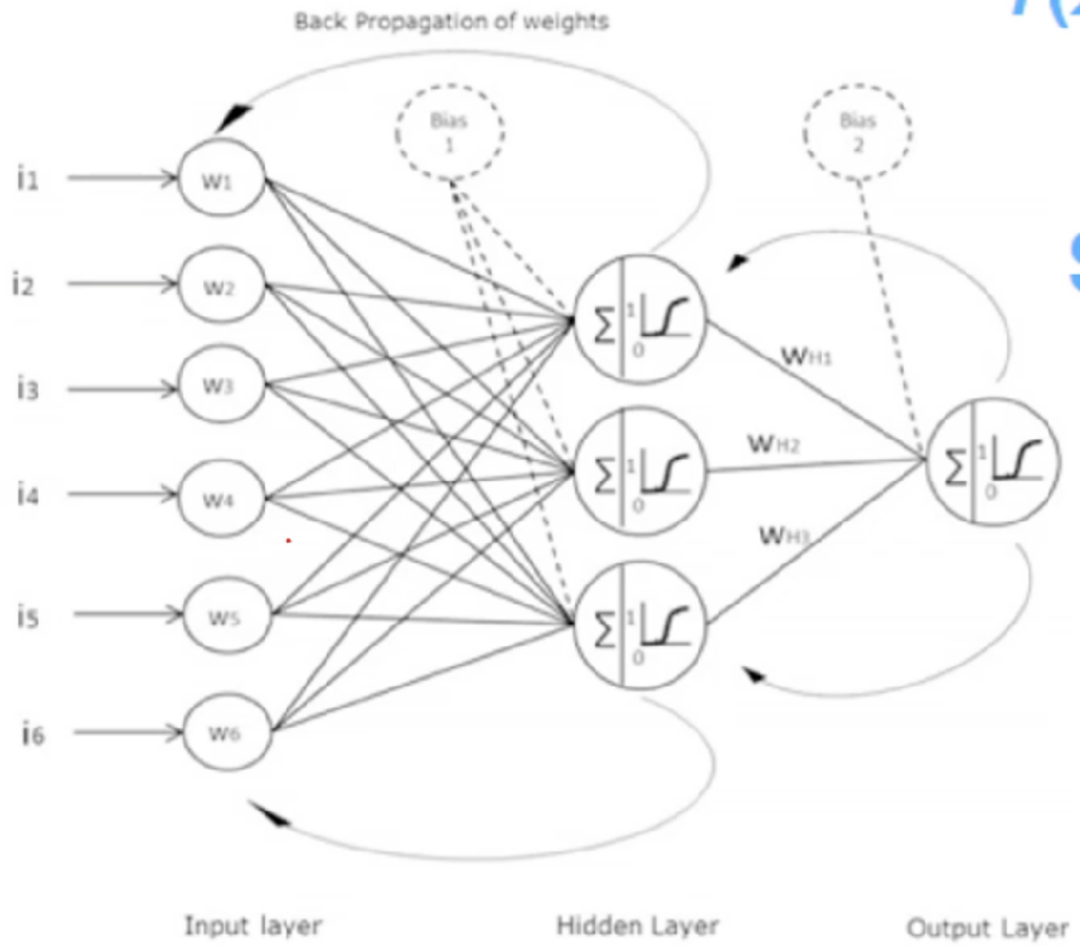






NeuralMind TECHNOLOGIES

Neural Network

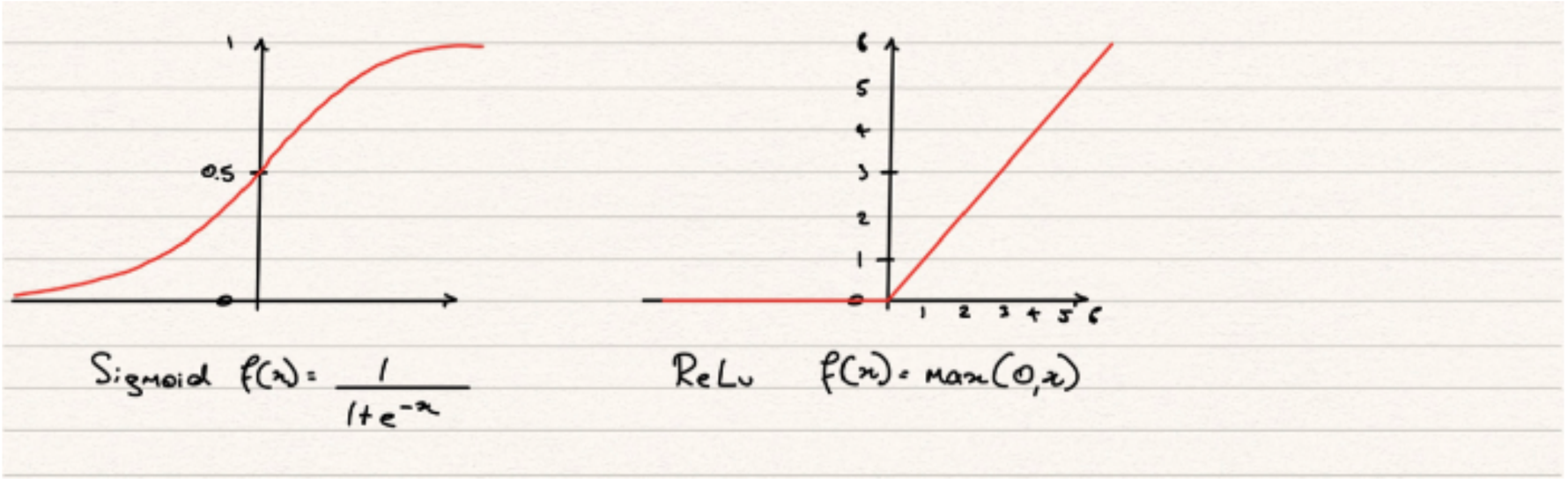


f (X) = nonlinear function composed of

Σ, Π, S

S(x) = 1 / (1 - e^x)

Não-linearidade



Fonte: https://towardsdatascience.com/the-components-of-a-neural-network-af6244493b5b

Output layer e a função de perda

- Ao final é preciso conectar o output a variável que se quer predizer.
- Dependendo do tipo de variável temos diferentes funções de perda

Table 4.1 Choosing the right last-layer activation and loss function for your model

Problem type	Last-layer activation	Loss function
Binary classification	`sigmoid`	`binary_crossentropy`
Multiclass, single-label classification	`softmax`	`categorical_crossentropy`
Multiclass, multilabel classification	`sigmoid`	`binary_crossentropy`
Regression to arbitrary values	`None`	`mse`
Regression to values between 0 and 1	`sigmoid`	`mse` or `binary_crossentropy`

TensorFlow

- Biblioteca open-source para cálculos numéricos.
- Desenvolvida incialmente pela Google.
- Foco em Machine Learning e principalmente Deep Learning.
- Muito rápido - implementado para diversos hardwares como GPU's e até TPU's.
- Feature: **Automatic Differentiation**!
- Um grande ecossistema de addosn e extensões.
- Biblioteca mais utilizada para fazer Deep Learning atualmente.

Keras

- É uma biblioteca open-source criada para especificar modelos de Deep Learning
- Foi criada antes do TensorFlow existir
- Funciona com múltiplos 'backends' - exemplo Theano, CNTK e PlaidML
- Foi **incorporada pelo TensorFlow** e à partir do 2.0 é a forma recomendada de especificar modelos no TensorFlow

fonte: https://curso-r.github.io/202005-deep-learning/

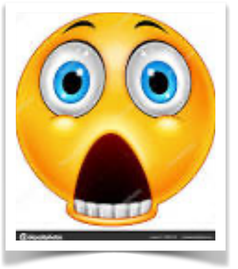
Algebra linear

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3\times 2} \times \begin{bmatrix} 1 & 1 & 1 \\ 2 & 2 & 2 \end{bmatrix}_{2\times 3} =$$

$$\begin{bmatrix} 1\times 1 + 2\times 2 & 1\times 1 + 2\times 2 & 1\times 1 + 2\times 2 \\ 3\times 1 + 4\times 2 & 3\times 1 + 4\times 2 & 3\times 1 + 4\times 2 \\ 5\times 1 + 6\times 2 & 5\times 1 + 6\times 2 & 5\times 1 + 6\times 2 \end{bmatrix}_{3\times 3} =$$

$$\begin{bmatrix} 5 & 5 & 5 \\ 11 & 11 & 11 \\ 17 & 17 & 17 \end{bmatrix}$$

Algebra linear



$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}_{3\times 2} \Rightarrow \grave{A} \text{ ou } A^T = \begin{bmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix} \Rightarrow \text{DIAG}(B) = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}$$

- https://www.youtube.com/watch?v=bxe2T-V8XRs&t=84s

- https://www.3blue1brown.com/topics/neural-networks