Regressão Logística, o início das RNA

Camila Nori Dias Kubota

Definição:

Regressão Logística

- *“This type of statistical model (also known as logit model) is often used for classification and predictive analytics. Logistic regression estimates the probability of an event occurring, such as voted or didn’t vote, based on a given dataset of independent variables.”*

Site da IBM (Visitado em: 19/04/2023)[1]

Definição:

Ou seja:

- É um modelo voltado a problemas de classificação, definições binárias.

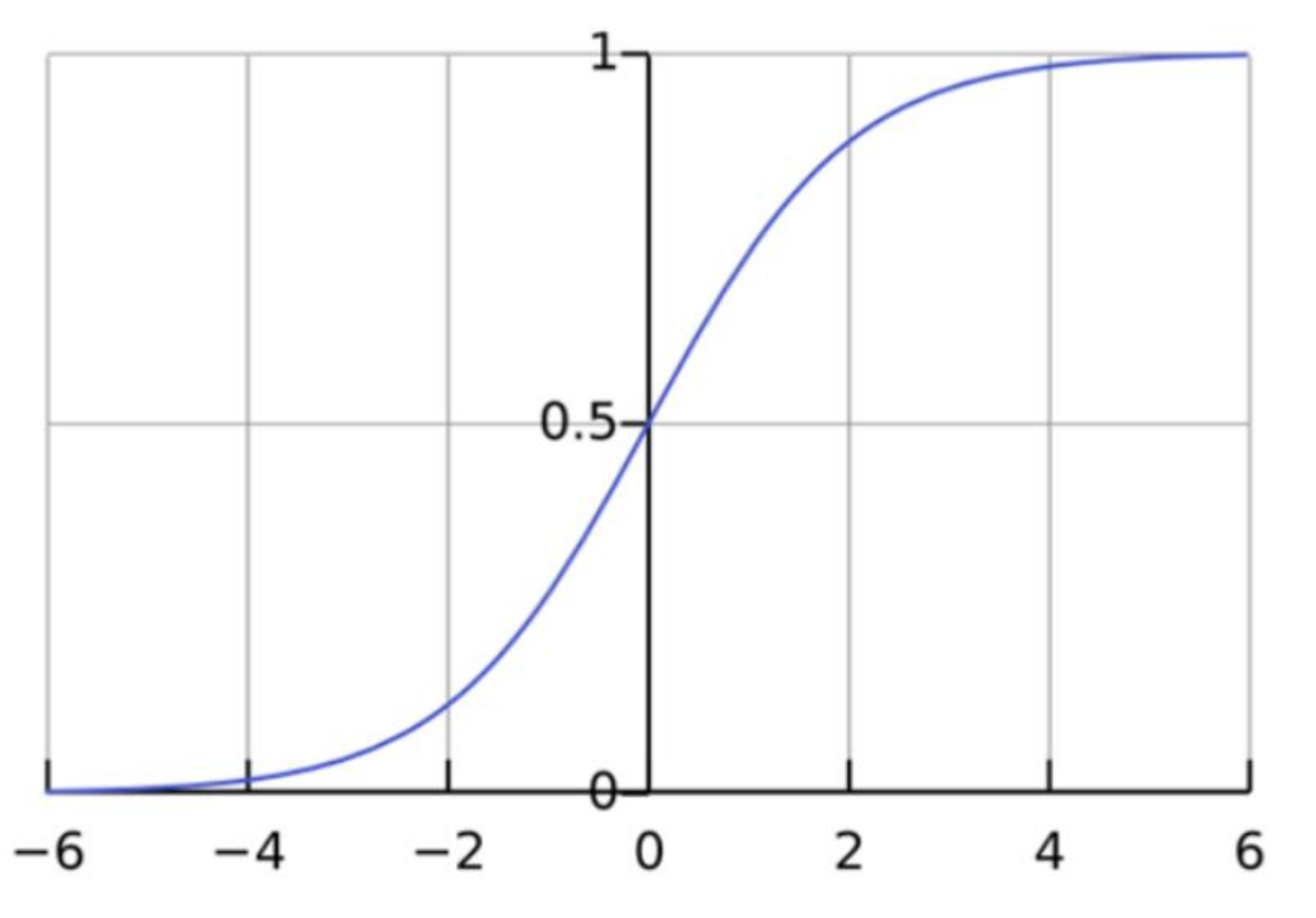
Ex: Prever se um cliente irá ou não comprar seu produto.

Principal ponto que diferencia na hora de decidir utilizar Regressão Linear ou Regressão Logística

Função:

$$f(x) = \frac{1}{1 + e^{-z}}$$

Função: Sigmoide



$$f(x) = \frac{1}{1 + e^{-z}}$$

Retirado do site AWS Amazon[2]

Tipos:

- Binária:

  Somente dois resultados possíveis.

  Ex: 0 e 1

- Multinomial

  De 3 a mais resultados possíveis, porém não ordenados

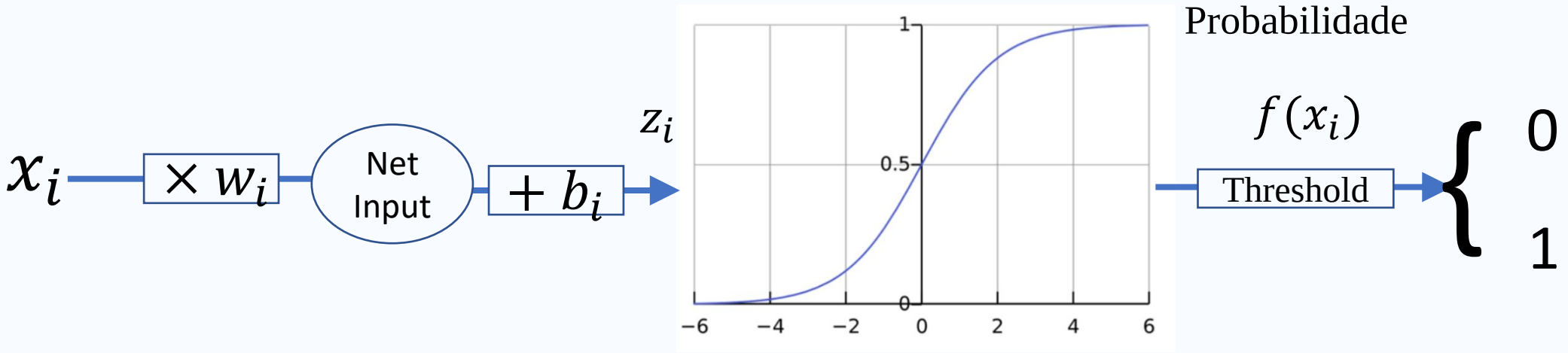
  Ex: Um estúdio deseja estudar a influência da idade, sexo e status de relacionamento em comparação ao gênero de filme favorito

- Ordinal:

  De 3 a mais resultados possíveis, mas ordenados

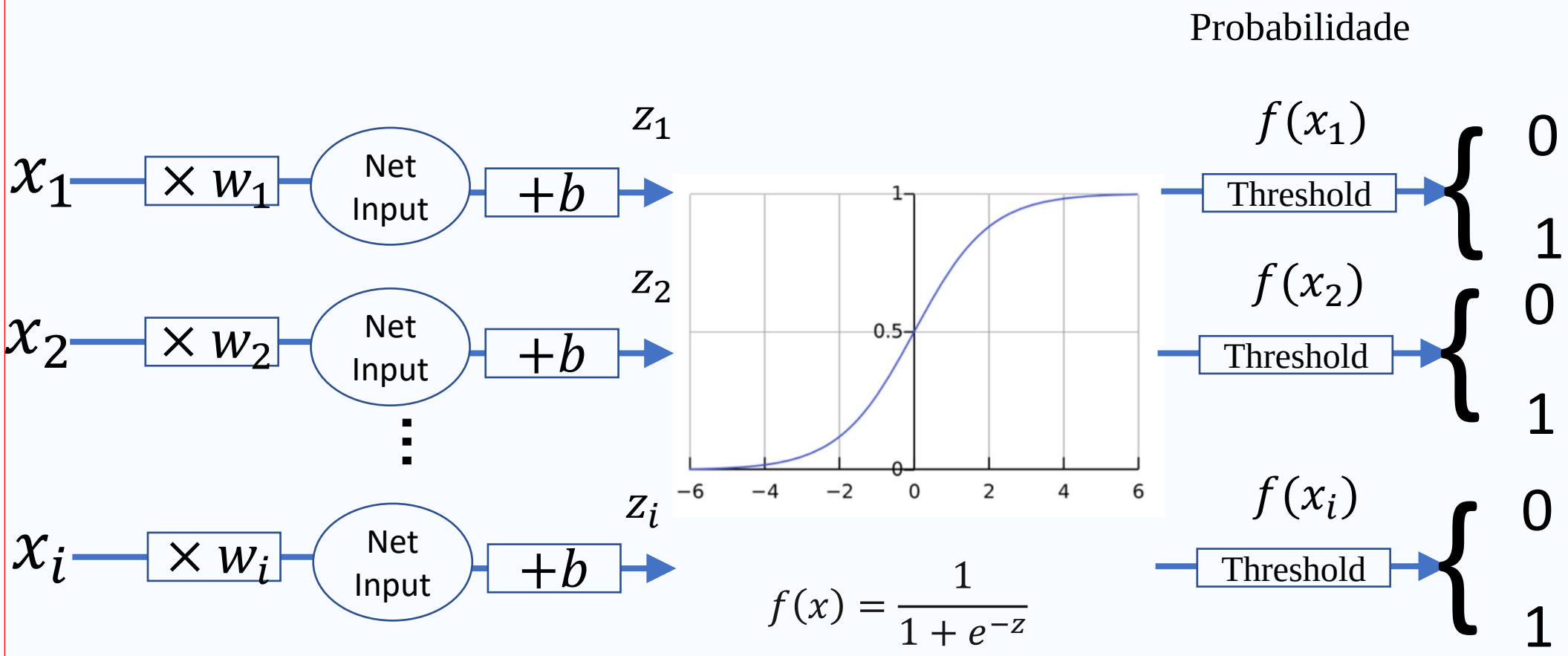
  Ex: Escala de 1 a 10 ou de A até F

Funcionamento:



$$f(x) = \frac{1}{1 + e^{-z}}$$

Funcionamento:



Funcionamento:

$$f(x) = \frac{1}{1 + e^{-z}}$$

$$z = \left(\sum_{i=1}^{n} w_i x_i \right) + b$$

Threshold:

É o “limiar”, valor definido em que:

- Caso a probabilidade esteja acima dele → Definido como o primeiro valor
- Caso esteja abaixo → Definido como o segundo valor

Exemplo:

$z \geq 0{,}75 \quad \rightarrow \quad = 1$

$z < 0{,}75 \quad \rightarrow \quad y = f(x) = 0$

Loss function:

Comumente utiliza-se como *loss function* a função denominada Cross Entropy.

De forma resumida:

$$loss = -p_{observado} \times \ln\left(p_{prevista}\right)$$

Ponto importante:

Os dados de input devem ser transformados para valores numéricos!

Ponto importante:

É uma boa prática normalizar os inputs

Deixar todas as variáveis latentes com o mesmo peso e manter seus valores entre 0 e 1, otimizando o gradiente descente.

Ponto importante:

Em certas situações otimizar e limpar seus dados pode ser mais eficiente na melhora do modelo do que alterar parâmetros da rede.

Um método de normalização:

Autoescalamento

$$x'_i = \frac{x_i - \bar{x}_i}{\sigma_i}$$

Um método de normalização:

Autoescalamento

$$\bar{x}_i = \frac{\left(\sum_{j=1}^{n} x_i^{(j)}\right)}{n}$$

$$\sigma_i = \sqrt{\frac{1}{n}\left(\sum_{i=1}^{n} x_i^{(j)} - \bar{x}_i\right)}$$

Bibliotecas

- Numpy
- Scikit-Learn
- Tensorflow (Keras)

Prática:

Banco de Dados

Foi utilizado o banco de dados do Titanic no Kaggle [3].

https://www.kaggle.com/competitions/titanic/data

Exemplos:

Numpy

```python
import numpy as np

def sigmoid(x):
   return (1/(1+np.exp(-x)))

class LogisticRegression():

    def __init__(self,lr=0.001,n_iters=1000):
        self.lr= lr
        self.n_iters=n_iters
        self.weights=None
        self.bias=None

    def fit(self, X, y):
        n_samples, n_features= X.shape
        self.weights=np.zeros(n_features)
        self.bias=0

        for _ in range(self.n_iters):
            linear_predictions=np.dot(X, self.weights)+self.bias
            predictions=sigmoid(linear_predictions)

            dw=(1/n_samples)*np.dot(X.T,(predictions-y))
            db=(1/n_samples)*np.sum(predictions-y)

            self.weights=self.weights-self.lr*dw
            self.bias=self.bias-self.lr*db
```

```python
    def predict(self,X):
        linear_predictions=np.dot(X, self.weights)+self.bias
        y_pred=sigmoid(linear_predictions)
        class_pred=[0 if y<=0.5 else 1  for y in y_pred]
        return class_pred


    def accuracy(y_pred,y_test):
        return np.sum((y_pred==y_test)/len(y_test))
```

Exemplos:

Scikit-Learn

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression


x_train,x_test,y_train,y_test=train_test_split(par,pred,test_size=0.25)


Previs_titanic=LogisticRegression()
Previs_titanic.fit(x_train,y_train)


testing=Previs_titanic.predict(x_test)
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test, testing))
from sklearn.metrics import classification_report
print(classification_report(y_test, testing))
```

Exemplos:

Tensorflow (Keras)

```
#Defining the model
model=keras.Sequential([keras.layers.Dense(1,input_shape=(len(labels),),
                                           activation='sigmoid')])
model.compile(optimizer='adam',loss='binary_crossentropy')


#Fitting the model
hist=model.fit(x_train,y_train,epochs=1000,validation_split=0.2)
```

Acurácia:

- Numpy

  0,74 (74%)

- Scikit-Learn

  0,76 (76%)

- Tensorflow (Keras)

  0,81 (81%)

Referências:

[1] https://www.ibm.com/topics/logistic-regression#:~:text=Resources-,What%20is%20logistic%20regression%3F,given%20dataset%20of%20independent%20variables.

[2] https://aws.amazon.com/pt/what-is/logistic-regression/#:~:text=A%20regress%C3%A3o%20log%C3%ADstica%20%C3%A9%20uma,resultados%2C%20como%20sim%20ou%20n%C3%A3o.

[4]https://edisciplinas.usp.br/pluginfile.php/3769787/mod_resource/content/1/09_RegressaoLogistica.pdf

[5] https://statquest.org/video-index/

[6] Jurafsky D., Martin J. H., "*Speech And Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, And Speech Recognition*" , 3ª edição,