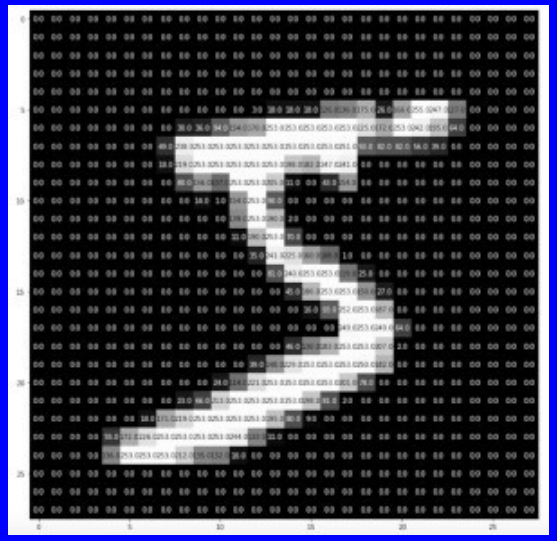
Machine Learning

Reconocimiento de Números



Rogelio Ferreira Escutia

Profesor / Investigador
Tecnológico Nacional de México
Campus Morelia

Formas de Trabajar

Formas de trabajar

- **Hay 2 formas de trabajar esta parte:**
  - **Local: descargando los datasets e instalación de todas las bibliotecas en una computadora local.**
  - **Nube: Utilizando alguna herramienta donde se pueda cargar los datasets y ejecutar el código (Google Colab).**

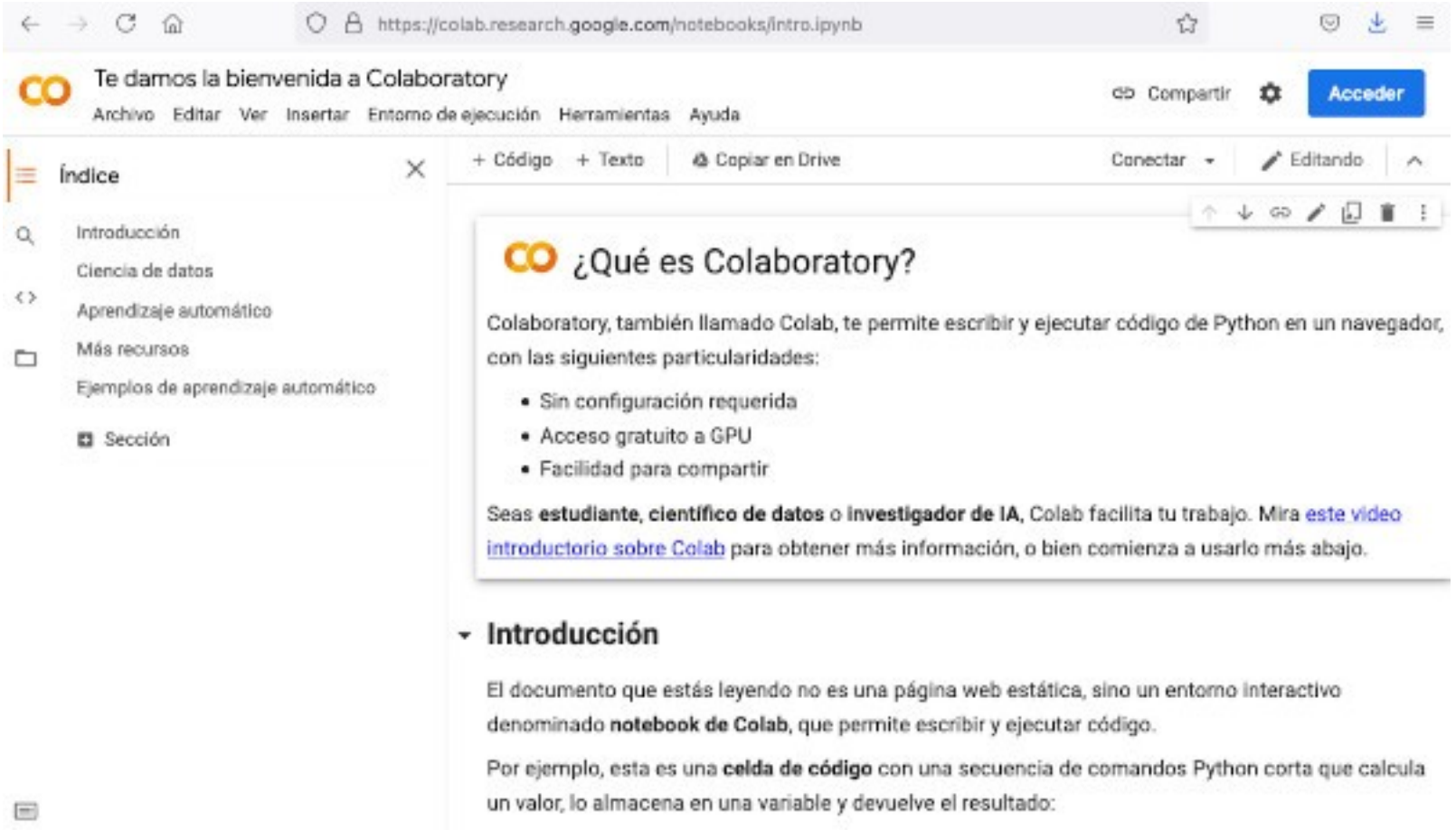
Con computadora local

- **Ventajas:**
  - **No depender de una conexión a Internet (después de descargar y configurar).**
- **Desventajas:**
  - **Hay que instalar todo lo necesario de manera local y dependemos del hardware de nuestra computadora (puede ser muy lento).**

Utilizando una aplicación en Nube

- **Ventajas:**
  - **Alta velocidad, no es necesario configurar nada.**
- **Desventajas:**
  - **Dependemos de las restricciones de la aplicación y de la conexión a Internet.**

Google Colaboratory



"Google Colaboratory", https://colab.research.google.com/notebooks/intro.ipynb, Junio 2021

Instalación

Instalación local

- **Se requiere instalar Numpy , TensorFlow y Keras:**
  - **> pip3 install numpy tensorflow keras**

Instalación en nube

- **No se requiere instalar nada, solo un navegador (se recomienda Chrome) y entrar a Google Colaboratory:**
  - **https://colab.research.google.com/notebooks/intro.ipynb**
- **Después hacer click en:**
  - **Acceder**
- **Y teclear login y password de una cuenta de Gmail.**
- **Luego seleccionamos:**
  - **Archivo > Bloc de notas nuevo**

Instalación en nube

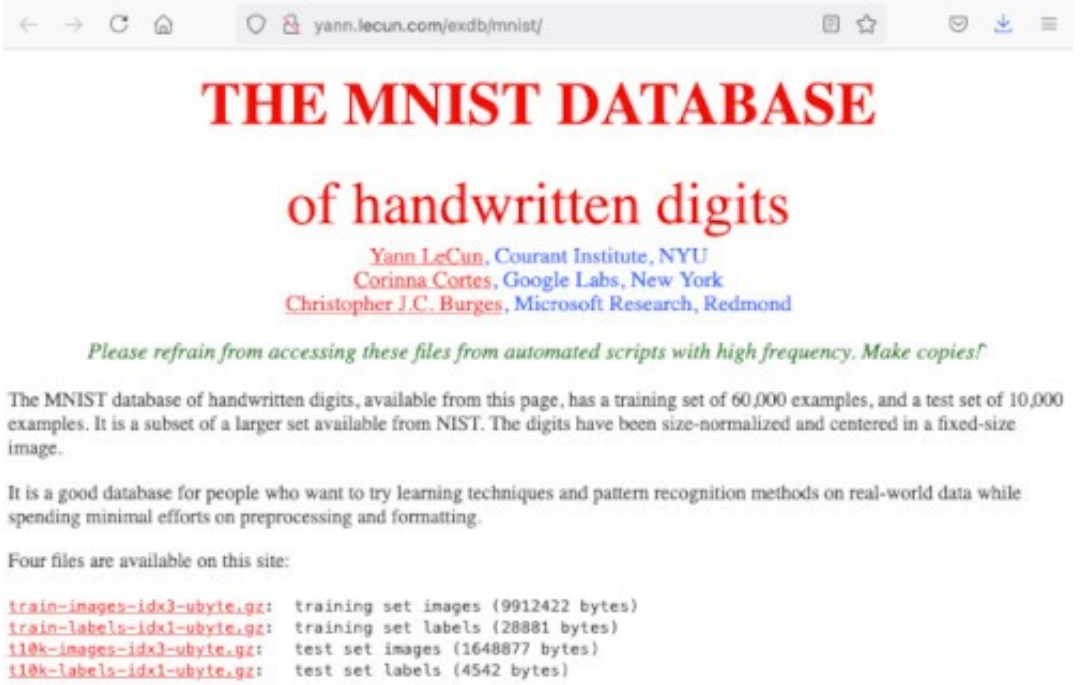
- **Entorno Google Colaboratory listo para trabajar:**

MINST

MNIST

- **Yann LeCun, Corinna Cortes y Christopher J.C. Burges crearon MNIST, una base de datos con imágenes de números escritos a mano.**

- **Contiene 60,000 imágenes de ejemplo y otras 10,000 imágenes para hacer pruebas.**



"MNIST", http://yann.lecun.com/exdb/mnist/, Junio 2021

Reconocimiento de números

Código

- **Cargamos bibliotecas necesarias:**

```
# Bibliotecas necesarias
import tensorflow as tf
from tensorflow import keras
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.utils import to_categorical
```

Código

- **Cargamos el dataset:**

```
print("Detección de números")
# Cargar el dataset con datos de entrenamiento y prueba del sitio de Google
# https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
mnist = tf.keras.datasets.mnist
(x_entrenamiento, y_entrenamiento), (x_prueba, y_prueba) = mnist.load_data()
```

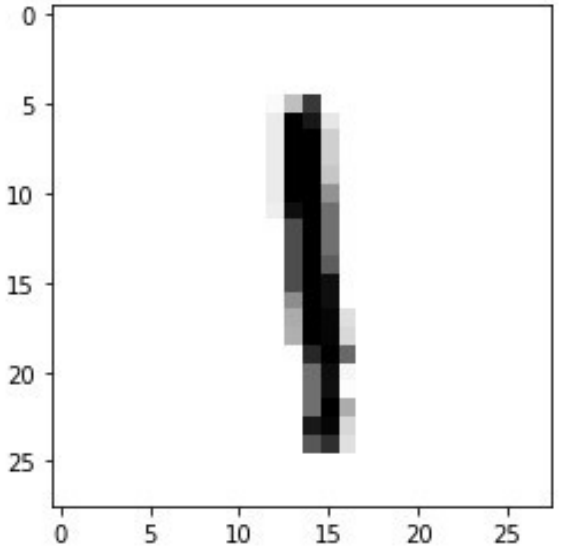
Código

- **Se imprime un número de muestra:**

```
# Para ver el dato de entrenamiento no. 8
plt.imshow(x_entrenamiento[8], cmap=plt.cm.binary)
```

- **En pantalla sale:**

```
Detección de números
<matplotlib.image.AxesImage at 0x7f2438d89110>
```



Código

- **Imprimir etiqueta, dimensión y contenido:**

```
# Para ver la etiqueta del dato de entrenamiento no. 8
print("Etiqueta: ", y_entrenamiento[8])

# Para ver el tamaño de nuestro tensor x_entrenamiento
print("Dimensiones de nuestro tensor x_entrenamiento: ", x_entrenamiento.ndim)

# Para ver el contenido de nuestro tensor x_entrenamiento
print("Contenido de nuestro tensor x_entrenamiento: ", x_entrenamiento.shape)
```

- **En pantalla sale:**

```
Etiqueta:  1
Dimensiones de nuestro tensor x_entrenamiento:  3
Contenido de nuestro tensor x_entrenamiento:  (60000, 28, 28)
```

Código

- **Convertir todos los datos a float32 (para evitar que haya datos con tipos diferentes:**

```
# Escalar los datos al tipo de formato float 32
x_entrenamiento = x_entrenamiento.astype('float32')
y_entrenamiento = y_entrenamiento.astype('float32')
```

Código

- **Escalar los datos a un rango solamente de 0 a 255:**

```
# Escalar los datos para que tengan valores entre 0 y 255
x_entrenamiento /= 255
y_entrenamiento /= 255
```

Código

- **Convertir nuestra matríz (la de entrenamiento y la de prueba) a un vector, para poder alimentar de datos a nuestra red neuronal:**

```
# Convertimos la matríz de 28x28 a un vector de 784
x_entrenamiento = x_entrenamiento.reshape(60000, 784)
x_prueba = x_prueba.reshape(10000, 784)
```

Código

- **Convertir las etiquetas de entrenamiento (por ejemplo una imagen que tiene un 7, su etiqueta es “7”) a un formato de salida para nuestra red:**

```
# Convertir las etiquetas de datos, de un número a una salida de 10 números
# Ejemplo: La etiqueta "7" se convierte a [0,0,0,0,0,0,0,1,0,0]
# Esto es necesario para alimentar la red neuronal
y_entrenamiento = to_categorical(y_entrenamiento, num_classes=10)
y_prueba = to_categorical(y_prueba, num_classes=10)
```

Código

- **Seleccionar un modelo para la red neuronal:**

```
# Seleccionar el modelo de red neuronal a generar
modelo = tf.keras.Sequential()
```

Keras

- **Keras maneja los siguientes modelos de redes neuronales:**

Models API

There are three ways to create Keras models:

- The Sequential model, which is very straightforward (a simple list of layers), but is limited to single-input, single-output stacks of layers (as the name gives away).
- The Functional API, which is an easy-to-use, fully-featured API that supports arbitrary model architectures. For most people and most use cases, this is what you should be using. This is the Keras "industry strength" model.
- Model subclassing, where you implement everything from scratch on your own. Use this if you have complex, out-of-the-box research use cases.

"Keras – Models API", https://keras.io/api/models/, Junio 2021

Keras

- **Keras maneja los siguientes modelos de redes neuronales:**
  - **Sequential: Ideal para capas con una entrada y una salida por cada una de ellas.**
  - **Functional API: Para redes flexibles, con múltiples entradas y múltiples salidas.**
  - **Subclassing: Para desarrollar e investigar nuevas formas de conectar redes.**

Código

- **Creamos la “primer” capa de la red neuronal:**

```
# Creamos la primer capa de la red neuronal con activación "sigmoide"
modelo.add(tf.keras.layers.Dense(10, activation='sigmoid', input_shape=(784,)))
```

Funciones de Activación:

- **Las funciones que gobiernan el comportamiento de la neurona artificial se llaman funciones de activación.**
- **La transmisión de esa entrada se conoce como forward propagation.**
- **Las funciones de activación transforman la combinación de entradas, pesos y sesgos.**
- **Los productos de estas transformaciones se ingresan para la siguiente capa de nodo.**
- **Muchas (aunque no todas) las transformaciones no lineales usadas en redes neuronales transforman los datos a rango conveniente, por ejemplo [0,1] ó [-1,1].**
- **Cuando una neurona artificial pasa de un valor distinto de cero a otro, decimos que esa neurona se ha activado.**

"Las matemáticas del Machine Learning: Funciones de activación", https://empresas.blogthinkbig.com/las-matematicas-del-machine-learning-funciones-de-activacion/, junio 2021

Funciones de Activación:

- **Algunas de las funciones de activación mas usadas:**

Identity		$f(x) = x$
Binary step		$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ 1 & \text{for } x \geq 0 \end{cases}$
Logistic (a.k.a. Sigmoid or Soft step)		$f(x) = \sigma(x) = \frac{1}{1+e^{-x}}$ [1]
TanH		$f(x) = \tanh(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$

"Las matemáticas del Machine Learning: Funciones de activación", https://empresas.blogthinkbig.com/las-matematicas-del-machine-learning-funciones-de-activacion/, junio 2021

Keras

- **Keras maneja las siguientes funciones de activación:**

Activation layers

- ReLU layer
- Softmax layer
- LeakyReLU layer
- PReLU layer
- ELU layer
- ThresholdedReLU layer

"Activation Layers", https://keras.io/api/layers/activation_layers/, Junio 2021

Keras

- **Características de la función de activación "sigmoide" (sigmoid):**

`sigmoid` function

```
tf.keras.activations.sigmoid(x)
```

Sigmoid activation function, `sigmoid(x) = 1 / (1 + exp(-x))`.

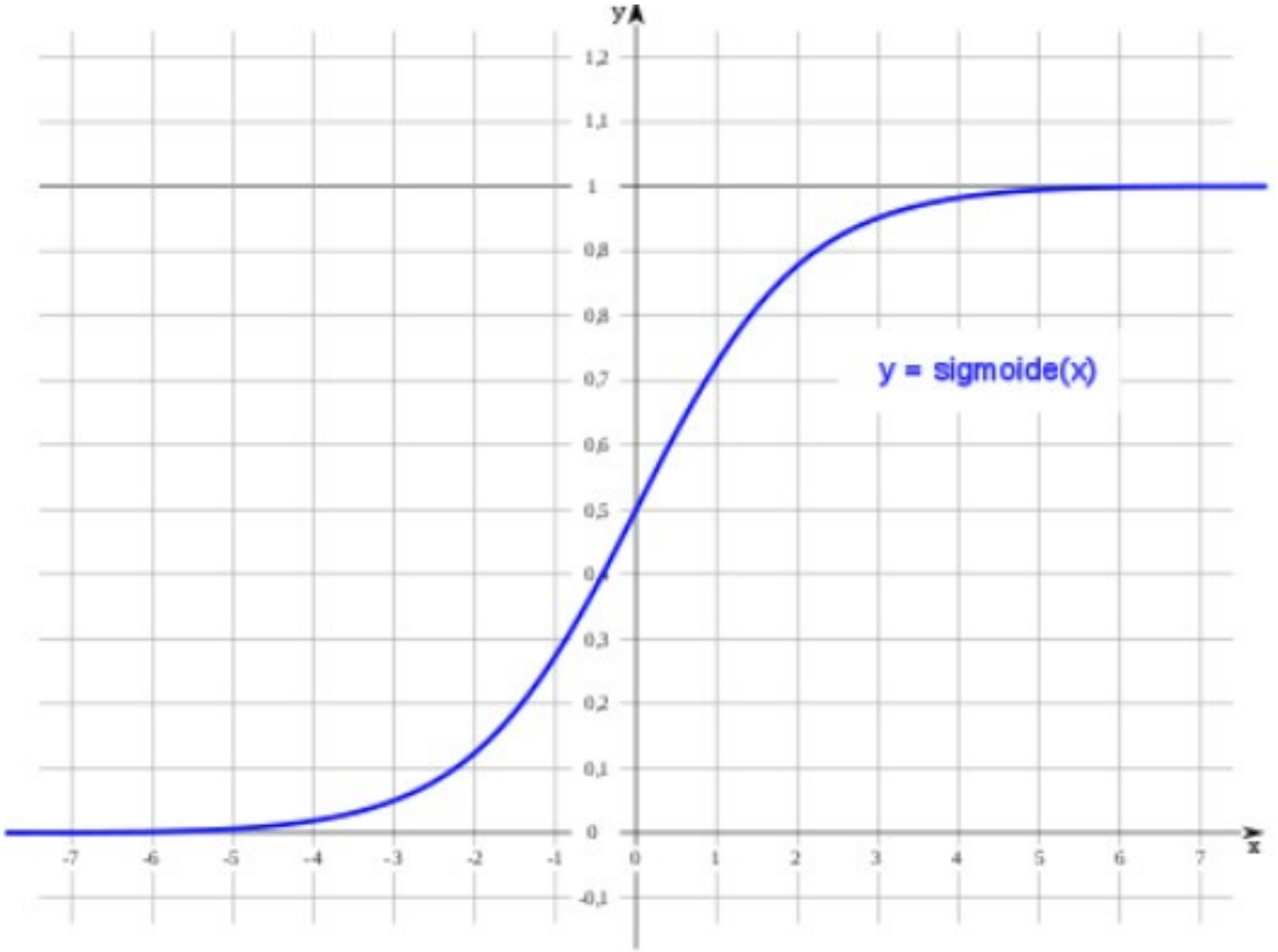
Applies the sigmoid activation function. For small values (<-5), `sigmoid` returns a value close to zero, and for large values (>5) the result of the function gets close to 1.

Sigmoid is equivalent to a 2-element Softmax, where the second element is assumed to be zero. The sigmoid function always returns a value between 0 and 1.

"Sigmoid", https://keras.io/api/layers/activations/, Junio 2021

Función Sigmoide

- **Gráfica de la función Sigmoide:**



"Función Sigmoide", https://es.wikipedia.org/wiki/Funci%C3%B3n_sigmoide, Junio 2021

Código

- **Creamos la “segunda” capa de la red neuronal:**

```
# Creamos la segunda capa de la red neuronal con activación "softmax"
modelo.add(tf.keras.layers.Dense(10, activation='softmax'))
```

Código

- **Para imprimir la características de nuestro modelo:**

```
#Para imprimir un resumen de nuestra red neuronal
print(modelo.summary())
```

- **En pantalla sale:**

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 10)                7850
_________________________________________________________________
dense_4 (Dense)              (None, 10)                110
=================================================================
Total params: 7,960
Trainable params: 7,960
Non-trainable params: 0
_________________________________________________________________
```

Código

- **Especificar algunas características de nuestro modelo:**
  - **Función de coste.**
  - **Optimizador.**
  - **Métrica.**

```
# Especificaciones del modelo
#   Función de coste:   "categorical_crossentropy"
#   Optimizador:        "sgf (Stocastic Gradient Descent)
#   Métrica:           "acuracy"
modelo.compile(loss="categorical_crossentropy", optimizer="sgd", metrics = ['accuracy'])
```

Funciones de Coste

- **La función de coste trata de determinar el error entre el valor estimado y el valor real, con el fin de optimizar los parámetros de la red neuronal.**

- **Las funciones de coste mas utilizadas son:**

  - **Raíz cuadrada media – RMSE**
  - **Error absoluto medio – MAE**
  - **Error absoluto medio escalado – MASE**
  - **Entropía cruzada categórica – Categorical Cross-Entropy**
  - **Entropía cruzada binaria – Binary Cross-Entropy**

"Función de coste – Redes neuronales", https://www.diegocalvo.es/funcion-de-coste-redes-neuronales/, junio 2021

Keras

- **Funciones de coste disponibles en Keras.**

  - BinaryCrossentropy class
  - CategoricalCrossentropy class
  - SparseCategoricalCrossentropy class
  - Poisson class
  - binary_crossentropy function
  - categorical_crossentropy function
  - sparse_categorical_crossentropy function
  - poisson function
  - KLDivergence class
  - kl_divergence function

"Keras Losses", https://keras.io/api/losses/, Junio 2021

Optimizadores

- **Un optimizador lo que hace es obtener mejores valores de los parámetros, para reducir el error cometido por la red neuronal.**

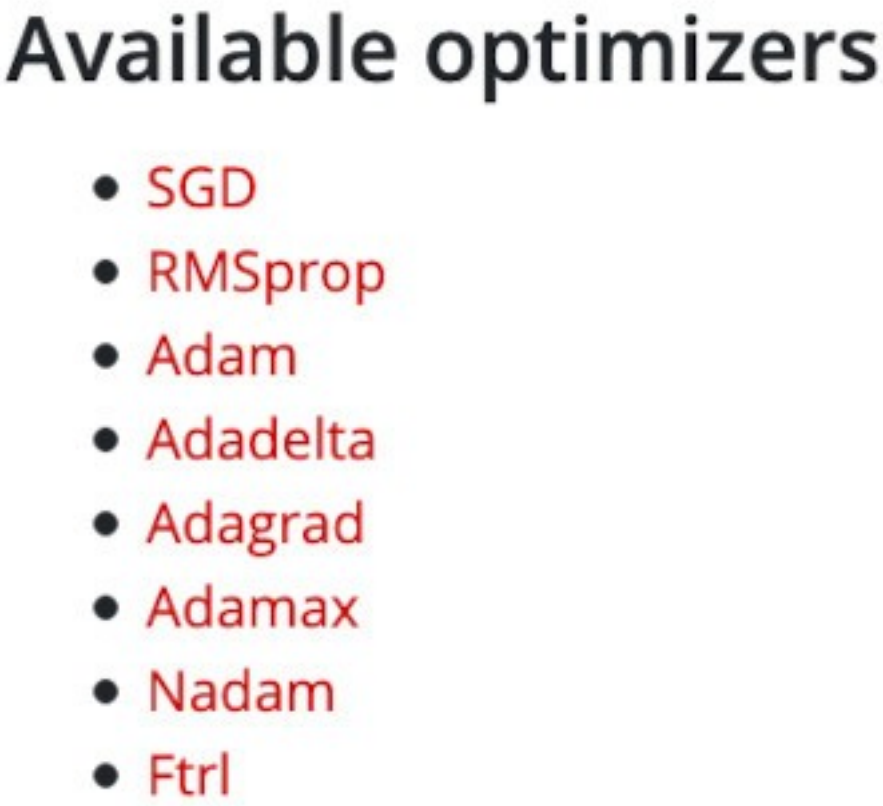
- **El proceso mediante el cual se hace esto se conoce como “backpropagation”.**

- **Algunos de los optimizadores mas usados son:**
  - **Stochastic Gradient Descent (SGD)**
  - **Adagrad**
  - **Adadelta**
  - **RMSprop**
  - **Adam**
  - **Nadam**

"¿Qué es un Optimizador y Para Qué Se Usa en Deep Learning?", https://datasmarts.net/es/que-es-un-optimizador-y-para-que-se-usa-en-deep-learning/, junio 2021

Keras

- **Los optimizadores disponibles en Keras son los siguientes:**



"Keras Optimizers", https://keras.io/api/optimizers/, junio 2021

Métricas

- **Las métricas son las funciones que se usan para evaluar el desempeño del modelo.**

- **Hay varios tipos de métricas y se clasifican de la siguiente manera:**

  - **Exactitud**
  - **Probabilísticas**
  - **Regresión**
  - **Clasificación**

"Keras Metrics", https://keras.io/api/metrics/, junio 2021

Keras

- **Las métricas de Exactitud disponibles en Keras son:**

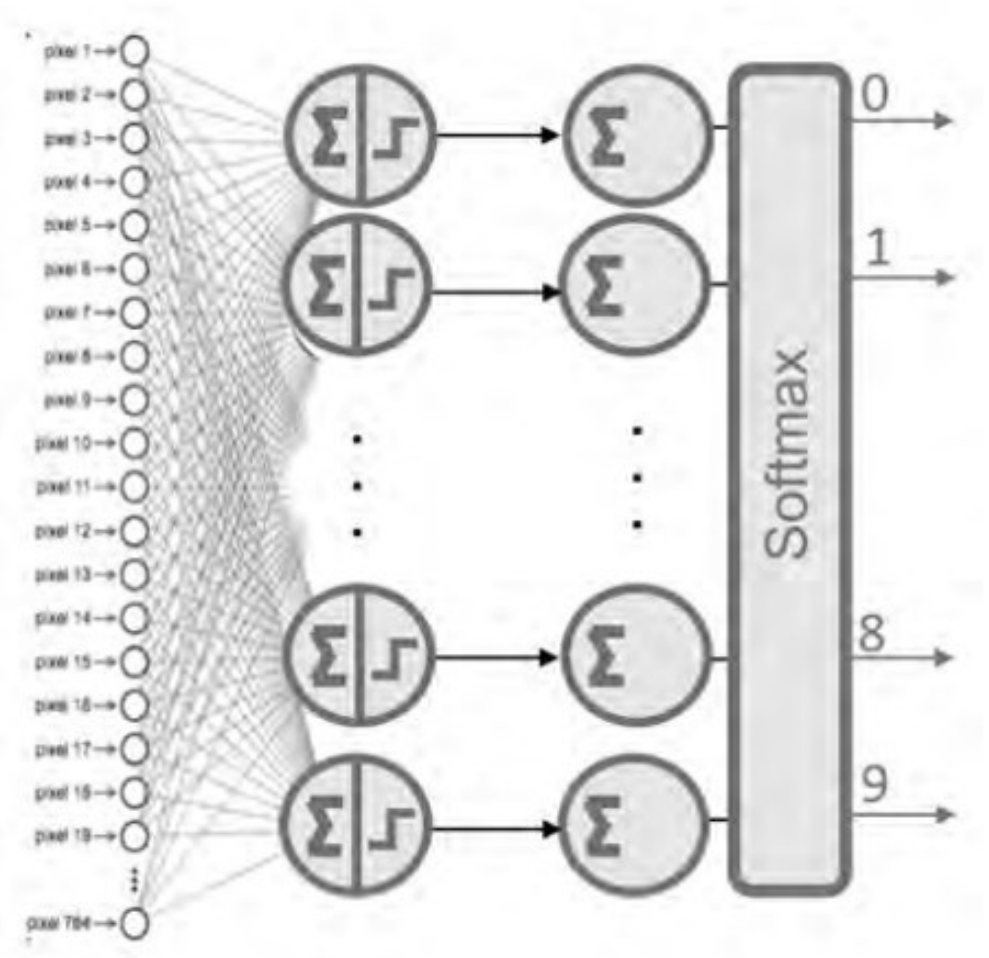
Accuracy metrics

- Accuracy class
- BinaryAccuracy class
- CategoricalAccuracy class
- TopKCategoricalAccuracy class
- SparseTopKCategoricalAccuracy class

"Keras Metrics", https://keras.io/api/metrics/, junio 2021

Diagrama

- **Este sería el diagrama del modelo propuesto:**



"Python Deep Learning", https://torres.ai/wp-content/uploads/2020/02/Empieza_a_leer_Python_Deep_Learning.pdf, junio 2021

Código

- **Se entrena el modelo para 5 épocas:**

```
# Entrenamiento del Modelo
modelo.fit(x_entrenamiento, y_entrenamiento, epochs=5)
```

- **En pantalla sale:**

```
Epoch 1/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0683 - accuracy: 0.9950
Epoch 2/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0048 - accuracy: 1.0000
Epoch 3/5
1875/1875 [==============================] - 3s 2ms/step - loss: 0.0027 - accuracy: 1.0000
Epoch 4/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0019 - accuracy: 1.0000
Epoch 5/5
1875/1875 [==============================] - 3s 1ms/step - loss: 0.0015 - accuracy: 1.0000
```

Código

- **Evaluar el modelo propuesto ahora con los datos de prueba:**

```
# Probar modelo con datos de prueba
prueba_perdida, prueba_precision = modelo.evaluate(x_prueba, y_prueba)

# Imprimir precisión
print("Precisión: ", prueba_precision)
```

- **En pantalla sale:**

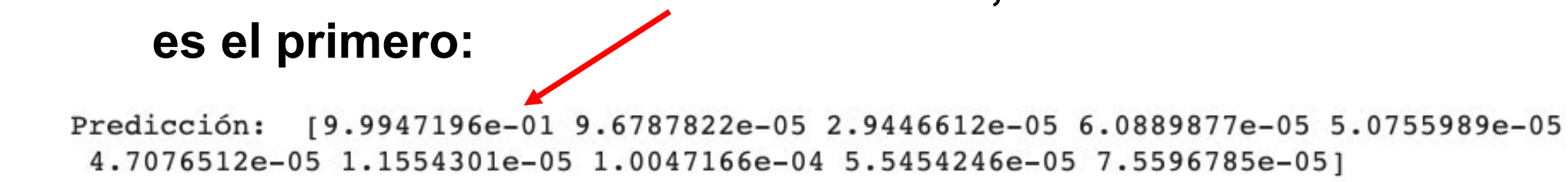
```
Precisión:  0.09799999743700027
```

Código – Parte Final - Predicción

- **Por último, le asignamos a nuestra red neuronal (ya entrenada) una imagen, y que haga la predicción del número que le presentamos (es la imagen número "11", nó el numero "11":**

```
# Predecir el valor de la imagen del número "11"
prediccion = modelo.predict(x_prueba)
np.argmax(prediccion[11])
print("Predicción: ", prediccion[11])
```

- **El modelo predice que es el número "0", ya que, en los valores de salida de la red neuronal, el número más alto es el primero:**



Para mejorar el resultado

- **Si el resultado no es bueno, se pueden hacer lo siguientes ajustes:**

- **Entrenar más veces nuestra red neuronal (asignar más "épocas") para mejorar los valores de predicción.**

- **Cambiar alguno de estos parámetros:**
  - **"función de coste"**
  - **"optimizador"**
  - **"métrica".**

- **Agregar más imágenes al entrenamiento.**

Rogelio Ferreira Escutia

Profesor / Investigador
Tecnológico Nacional de México
Campus Morelia

rogelio.fe@morelia.tecnm.mx

rogeplus@gmail.com

xumarhu.net

@rogeplus

https://www.youtube.com/channel/UC0on88n3LwTKxJb8T09sGjg

rogelioferreiraescutia