Sistemas Cognitivos Artificiales

Roberto Casado Vara

Tema 3: Frameworks de aprendizaje profundo

En este tema...

- Frameworks de aprendizaje profundo
- TensorFlow. Grafos de computación
- Otros frameworks
- Keras

Sistemas Cognitivos Artificiales

Roberto Casado Vara

Tema 3.1: Frameworks de aprendizaje profundo

Frameworks de deep learning

- Una consecuencia de la popularización del deep learning ha sido el surgimiento de una gran cantidad de paquetes de software para el entrenamiento de redes neuronales.

- Como sabemos, entrenar una red neuronal implica la realización de muchas operaciones, incluyendo el cálculo de gradientes sobre una gran cantidad de parámetros.

- La complejidad va en aumento con el uso de arquitecturas más avanzadas, por lo que se hace necesario contar con soluciones que faciliten el desarrollo.

- Un **framework** es una librería o conjunto de librerías de programación que facilita el desarrollo e implementación eficiente de redes neuronales o de algoritmos de machine learning en general.

Ventajas de los frameworks de deep learning

1. Simplifican el desarrollo de grafos de computación de gran tamaño

- La mayoría de frameworks entienden las redes neuronales como un grafo de computación, donde cada nodo es una operación.

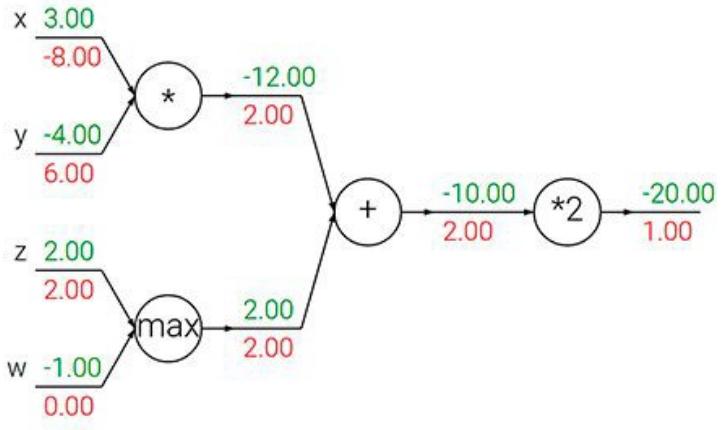
- De este modo, los frameworks nos dan una serie de abstracciones y operaciones ya disponibles que podemos conectar en forma de un grafo de computación (por ejemplo, activaciones sigmoid o convoluciones), facilitando el desarrollo.

Ventajas de los frameworks de deep learning

2.Calculan de manera automática los gradientes (auto-diferenciación)

- El cálculo de gradientes y el algoritmo de backpropagation es un proceso complejo de codificar donde podemos cometer fácilmente errores.

- A partir del **grafo de computación**, los frameworks modernos son capaces de realizar el *backward pass* de *backpropagation* de manera automática.



Ventajas de los frameworks de deep learning

3.Facilitan la ejecución y entrenamiento en GPUs y entornos distribuidos.

- Las tarjetas gráficas y los entornos distribuidos con varias máquinas se utilizan con frecuencia para agilizar el entrenamiento de redes neuronales.

- El uso de frameworks nos permite utilizar las GPUs y estos entornos de manera más sencilla, de manera que el programador pueda abstraerse en cierta medida de la complejidad que estos acarrean.

Ventajas de los frameworks de deep learning

4. **Optimizan el entrenamiento y ejecución de los algoritmos**

- Los frameworks emplean código altamente optimizado, utilizando librerías numéricas de alto rendimiento que permiten aprovechar las instrucciones especiales y capacidades multihilo de los procesadores.

- De nuevo, el programador no tiene que preocuparse de estos aspectos.

Sistemas Cognitivos Artificiales

Roberto Casado Vara

Tema 3.2: Tensorflow. Grafos de computación

TensorFlow

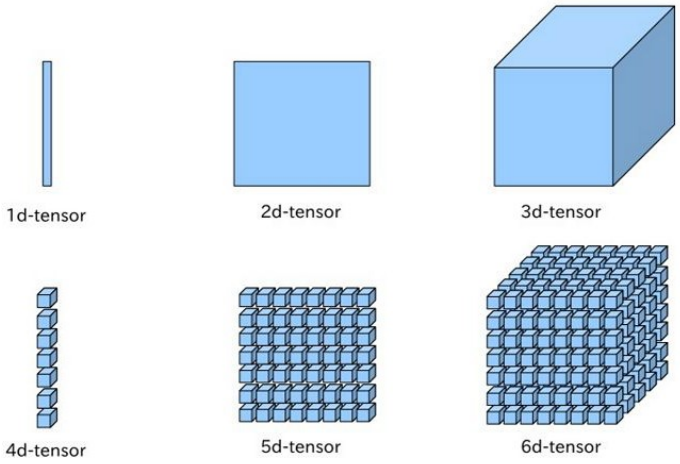
- **TensorFlow** es el framework más utilizado en la actualidad.
- Desarrollado por Google y presentado en 2015. Es *open source*.
- Utiliza un grafo de computación donde los datos, en forma de **tensores** (*Tensor)*, "fluyen" (*Flow*) entre distintas operaciones.
- Es un framework general de cálculo numérico, si bien su aplicación gira principalmente en torno al machine learning y, más en particular, al deep learning.

TensorFlow

- API principal en **Python**. Existen APIs para otros lenguajes como C++ y Java.
- Las operaciones internas (*kernels*) están implementados en C++ para una mayor velocidad de ejecución.

Tensores

- Un **tensor** es un array n-dimensional. Los tensores son la abstracción básica de TF.
    - Tensor de una dimensión: vector.
    - Tensor de dos dimensiones: matriz.
    - Tensor 0-dimensional: número escalar.

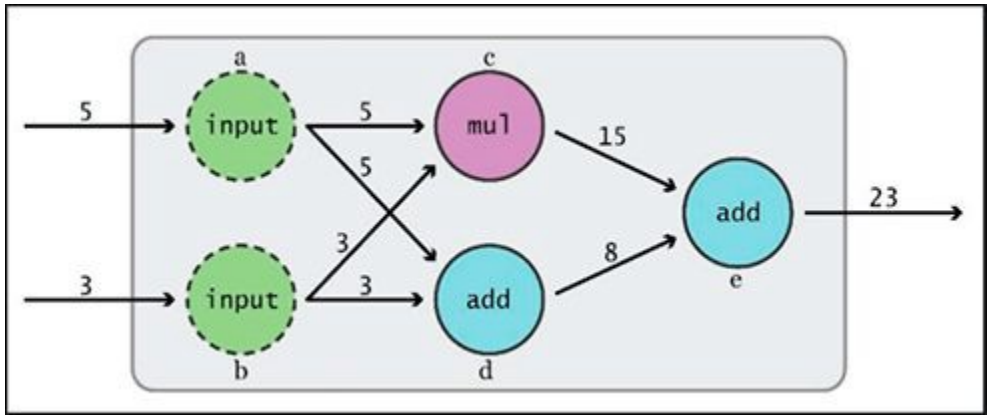


Fuente de la imagen: https://www.cc.gatech.edu/~san37/post/dlhc-start/

Grafos de computación

- Un **grafo de computación** es una representación de una serie de operaciones matemáticas. En particular, la representación es un **grafo dirigido** donde:
  - Los nodos son **operaciones**.
  - Las aristas son datos o **tensores**.

- Ejemplo: f(x, y) = xy + (x + y)

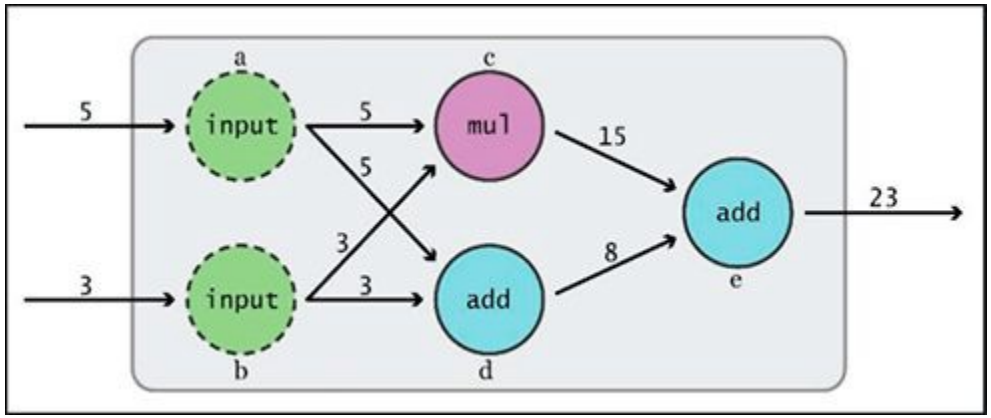


Fuente de la imagen: Abrahams, Hafner, Erwitt y Scarpinelli, 2016

Grafos de computación

- TensorFlow funciona mediante la construcción de un **grafo de computación** y el uso de una **sesión** que ejecuta las operaciones del grafo.

- Con la construcción del grafo, el framework obtiene la información necesaria sobre qué operaciones han de ejecutarse y en qué orden. Además, se asegura de que estas son compatibles entre ellas (por ejemplo, las dimensiones concuerdan).



Fuente de la imagen: Abrahams, Hafner, Erwitt y Scarpinelli, 2016

Grafos de computación

```
In [7]: import tensorflow as tf
   ...:
   ...: x = 2
   ...: y = 3
   ...: op1 = tf.add(x, y)
   ...: op2 = tf.multiply(x, y)
   ...: op3 = tf.pow(op2, op1)
   ...: with tf.Session() as sess:
   ...:     result = sess.run(op3)
   ...:
   ...: print(result)
   ...:
```

- El grafo se construye mediante la utilización de primitivas de TensorFlow.

- La sesión, instanciada mediante *tf.Session()*, es el entorno donde se ejecutan las operaciones definidas en el grafo computacional. La ejecución se hace mediante el método *run()*.

Grafos de computación

```
In [7]: import tensorflow as tf
   ...:
   ...: x = 2
   ...: y = 3
   ...: op1 = tf.add(x, y)
   ...: op2 = tf.multiply(x, y)
   ...: op3 = tf.pow(op2, op1)
   ...: with tf.Session() as sess:
   ...:     result = sess.run(op3)
   ...:
   ...: print(result)
   ...:
```

- Es importante comprender que **las operaciones no se están evaluando línea por línea**.
  - La variable *op1* no contiene 5.

- El código está construyendo el grafo de computación. La evaluación sólo ocurre cuando creamos una *session* y llamamos a *run()*.

- Al hacer *sess.run(op3)* estamos evaluando el grafo de computación que acaba en *op3*. TensorFlow obtiene en ese momento todas las operaciones que es necesario ejecutar (*op1, op2, op3*) y rellena los valores de los tensores de salida mediante una evaluación nodo a nodo. Si hubiéramos definido operaciones que no se usan para obtener *op3*, éstas serían ignoradas.

Ventajas de los grafos de computación

El uso de los grafos de computación tiene varias ventajas:

1. **Optimización de los cálculos a realizar**. Gracias al grafo, sabemos de antemano todas las operaciones a realizar, por lo que podemos optimizar estas operaciones e ignorar las que no sean necesarias.

2. **Facilita la auto-diferenciación**. Como sabemos, en *backpropagation* sólo necesitamos saber el valor de la derivada en el nodo y el gradiente que viene desde su salida. De este modo, TensorFlow recurre al grafo para calcular automáticamente las derivadas nodo a nodo.

3. **Facilita la ejecución en entornos distribuidos y GPUs**. Podemos dividir el grafo en varios subgrafos y hacer que distintos componentes de nuestro sistema se encarguen de diferentes partes.

```
import tensorflow as tf

import utils

DATA_FILE = "data/birth_life_2010.txt"

# Step 1: read in data from the .txt file
# data is a numpy array of shape (190, 2), each row is a datapoint
data, n_samples = utils.read_birth_life_data(DATA_FILE)

# Step 2: create placeholders for X (birth rate) and Y (life expectancy)
X = tf.placeholder(tf.float32, name='X')
Y = tf.placeholder(tf.float32, name='Y')

# Step 3: create weight and bias, initialized to 0
w = tf.get_variable('weights', initializer=tf.constant(0.0))
b = tf.get_variable('bias', initializer=tf.constant(0.0))

# Step 4: construct model to predict Y (life expectancy from birth rate)
Y_predicted = w * X + b

# Step 5: use the square error as the loss function
loss = tf.square(Y - Y_predicted, name='loss')

# Step 6: using gradient descent with learning rate of 0.01 to minimize loss
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.001).minimize(loss)

with tf.Session() as sess:
        # Step 7: initialize the necessary variables, in this case, w and b
        sess.run(tf.global_variables_initializer())

        # Step 8: train the model
        for i in range(100): # run 100 epochs
                for x, y in data:
                        # Session runs train_op to minimize loss
                        sess.run(optimizer, feed_dict={X: x, Y:y})

        # Step 9: output the values of w and b
        w_out, b_out = sess.run([w, b])
```

Fuente del código: Stanford CS 20: TensorFlow for Deep Learning Research

Sistemas Cognitivos Artificiales

Roberto Casado Vara

Tema 3.3: Otros frameworks

Theano

- Librería en Python para computación numérica. Creado en 2007 por Yoshua Bengio, puede considerarse el primer framework que empezó a utilizarse en el mundo del deep learning.
- Introduce la idea de ejecutar las operaciones en forma de grafo de computación.
- Incluye soporte para utilizar GPUs.

Torch / PyTorch

- **Torch** es otro de los frameworks “clásicos” junto a Theano. Está escrito en Lua.
- **PyTorch** es una evolución de Torch en Python. Incluye auto-diferenciación y está desarrollado por Facebook.
- PyTorch utiliza **grafos de computación dinámicos**. El grafo es calculado a la vez que el código se ejecuta, a diferencia de los grafos de TensorFlow (estáticos), donde el grafo se crea antes de ser ejecutado.

Torch / PyTorch

- Los grafos de computación dinámica tienen varias ventajas:
  - El código es más sencillo y más parecido a la programación “de toda la vida”.
  - El grafo puede cambiar de manera dinámica durante el entrenamiento, lo que facilita la creación de arquitecturas más complejas.

- TensorFlow también dispone de una variante de ejecución de grafos de computación dinámica llamada ***Eager Execution***.
  - **TensorFlow 2.0**: Eager Execution por defecto.

Caffe / Caffe2

- **Caffe** es otro de los frameworks de deep learning “clásicos”. Está escrito en C++ y su énfasis es en los sistemas en producción.
- Los modelos no se definen mediante código, sino con ficheros de configuración (.prototxt)
- El uso de Caffe ha descendido mucho en la actualidad dadas sus dificultades para definir modelos complejos, que requieren de larguísimos ficheros de configuración.
- **Caffe2**, desarrollado por Facebook, es la evolución de Caffe y también funciona mediante grafos estáticos de computación, con una interfaz en Python sobre operaciones escritas en C++.

Sistemas Cognitivos Artificiales

Roberto Casado Vara

Tema 3.4: Keras

Keras

- **Keras** es la librería de alto nivel para definir redes neuronales más utilizada en la actualidad.

- A diferencia de los frameworks vistos ya, Keras no define elementos de bajo nivel como operaciones y grafos de computación, sino que define una API de alto nivel sobre la que diseñar redes neuronales de manera sencilla y efectiva.

- De hecho, Keras utiliza de manera interna los frameworks vistos anteriormente para funcionar. En la actualidad, puede operar sobre TensorFlow, Theano y CNTK.

Keras

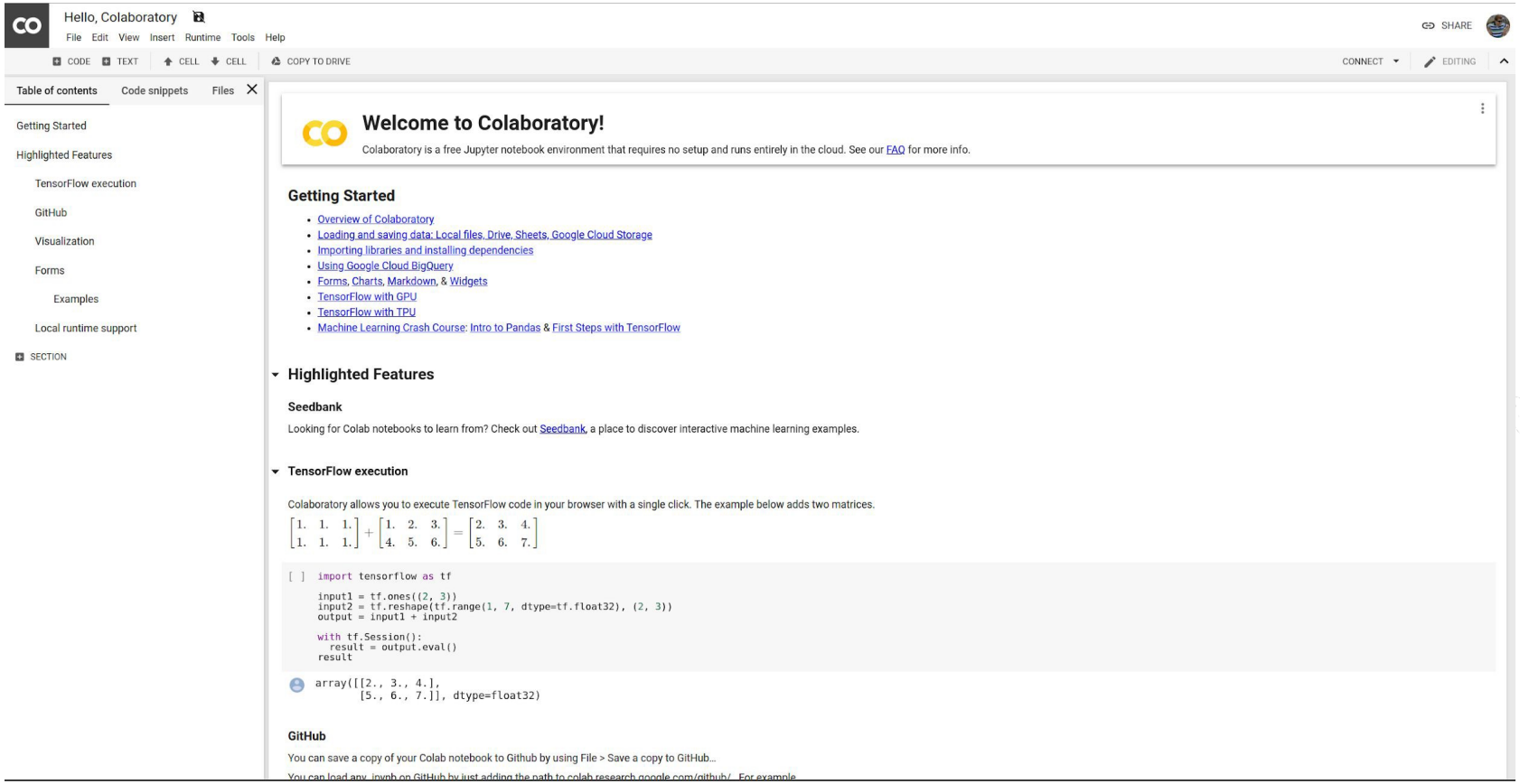
```
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import SGD

# Generate dummy data
import numpy as np
x_train = np.random.random((1000, 20))
y_train = keras.utils.to_categorical(np.random.randint(10, size=(1000, 1)), num_classes=10)
x_test = np.random.random((100, 20))
y_test = keras.utils.to_categorical(np.random.randint(10, size=(100, 1)), num_classes=10)

model = Sequential()
# Dense(64) is a fully-connected layer with 64 hidden units.
# in the first layer, you must specify the expected input data shape:
# here, 20-dimensional vectors.
model.add(Dense(64, activation='relu', input_dim=20))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
model.compile(loss='categorical_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])

model.fit(x_train, y_train,
          epochs=20,
          batch_size=128)
score = model.evaluate(x_test, y_test, batch_size=128)
```

Keras

- TensorFlow incluye Keras como una interfaz propia de alto nivel. Por ello, podemos utilizar directamente Keras desde TF (tf.keras).
  - **TensorFlow 2.0**: tf.keras pasa a ser la interfaz estándar de uso de TensorFlow.

- Página web: https://keras.io/. Documentación de la librería, ejemplos, etc.

Colaboratory

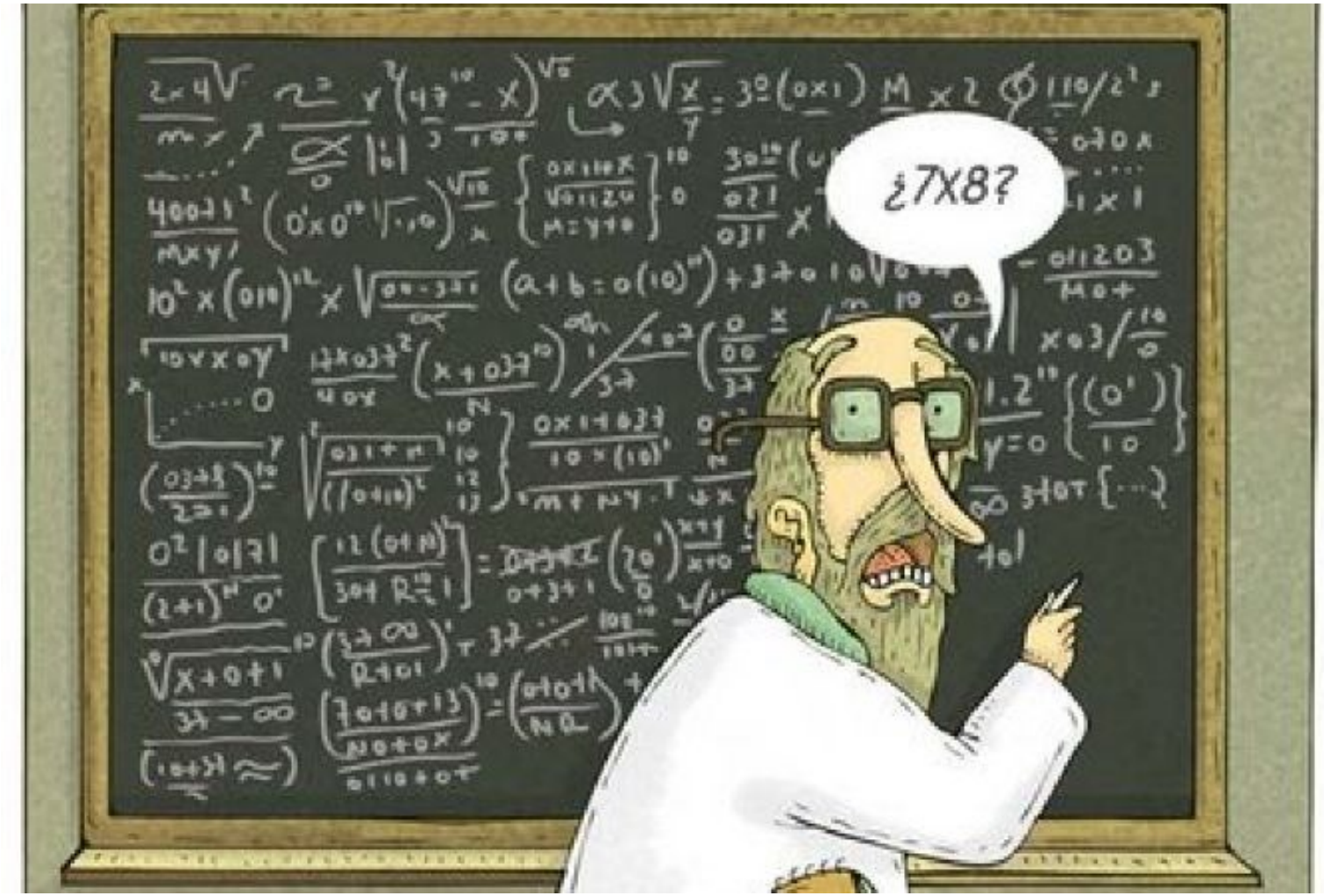


https://colab.research.google.com/

Fuentes de los datos para el Deep Learning

- Vamos a ver como importar datos y guardarlos en ciertas rutas, consultar el código de clase.

¿Dudas?



www.unir.net