



Módulo Implantación Tecnológica. Torch, Tensorflow y Keras - Actividad 1. Videoconferencia inicial

PROGRAMA EJECUTIVO DE IA Y DEEP
LEARNING

On-line -

2019

PROFESOR

Javier Abascal Carrasco



Esta publicación está bajo licencia Creative Commons Reconocimiento, Nocomercial, Compartirigual, (by-nc-sa). Usted puede usar, copiar y difundir este documento o parte del mismo siempre y cuando se mencione su origen, no se use de forma comercial y no se modifique su licencia. Más información: <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Índice

1. Objetivo de la semana de trabajo
2. Profesor y datos de contacto - Javier Abascal
3. Introducción a Torch, Tensorflow y Keras
4. Instalación
5. A programar!

1. Objetivo de la semana de trabajo

El objetivo principal es que el alumno entienda en que consiste el día a día de un trabajo que implique programar en Python para analizar datos y realizar modelos de Machine Learning y Deep Learning. Para ellos nos vamos a ayudar de unos ejercicios de ejemplo y unos ejercicios propuestos que tendrán que ser entregados

2. Profesor y datos de contacto - Javier Abascal

- Ingeniero de Telecomunicación - Universidad de Sevilla
- MBA - Thomas College, ME, USA
- Máster en Big Data - U-Tad, Madrid
- Ingeniero de Datos en Facebook, Londres (disculpad anglicismos)
- Ilusionista - www.javierabascal.com (hobby)


En resumen: Mezcla de consultor tradicional (por las posiciones anteriores) y programación focalizadas en analytics

Contacto:

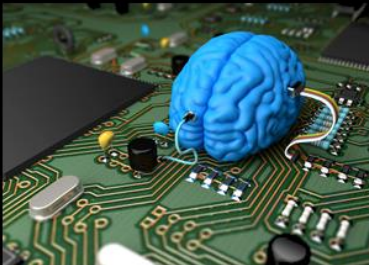
Jabascal@learning.eoi.es → Javier.Abascal@hotmail.com




Deep Learning




What society thinks I do




What my friends think I do



What other computer scientists think I do



What mathematicians think I do



What I think I do

```
In [1]:  
  
import keras  
  
Using TensorFlow backend.
```

What I actually do

3. Introducción a Torch, Tensorflow y Keras

El aprendizaje profundo (Deep Learning en inglés) con redes neuronales es actualmente una de las ramas de la inteligencia artificial más prometedora. Esta innovadora tecnología se usa comúnmente en aplicaciones como reconocimiento de imágenes, de voz, sistemas de traducción automática, entre otras.

Existen diversas opciones, librerías y frameworks de trabajo. Sin embargo, la más utilizada hoy en día es Tensorflow (de **Google**). No obstante, Torch (de **Facebook**), o en este caso PyTorch, una alternativa emergente que está ganando tracción rápidamente gracias a su facilidad de uso. Es la librería principal de Facebook para aplicaciones de aprendizaje profundo.

Ambas opciones son Open Source y su código está disponible en github. Si quieres saber las diferencias más notables se recomienda leer los siguientes enlaces (para nuestra clase serán muy similares)

- <https://towardsdatascience.com/pytorch-vs-tensorflow-spotting-the-difference-25c75777377b>
- <https://www.developereconomics.com/tensorflow-vs-pytorch>

No obstante, las podríamos enumerar en:

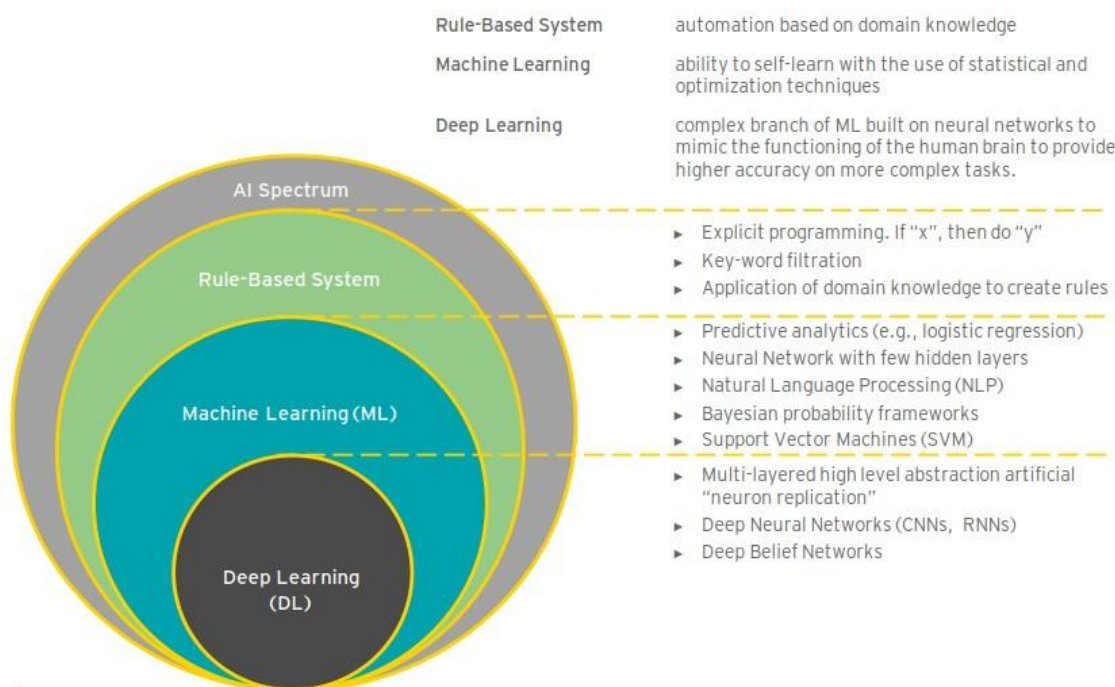
- Google vs Facebook
- Tensorflow define el grafo computacional de manera estática y PyTorch genera un gráfico dinámico. Esto es bastante útil cuando se quieren utilizar entradas variables en RNN
- PyTorch es más “pythonic” que Tensorflow, esto hace que Tensorflow tenga una curva de aprendizaje algo mayor
- Tensorflow es usado por un mayor número de usuarios 3v1 y la mayoría de cursos, tutoriales y MOOCs están basados en Tensorflow. PyTorch es más nuevo - apareció más tarde, lo que hace que el contenido disponible sea menor.
- Tensorflow posee TensorBoard que permite visualizar la red neuronal en el navegador. PyTorch utiliza otra herramienta (visdom) pero no está a la altura de TensorBoard
- Tensorflow es más maduro para código en producción y paralelizable.
- PyTorch soporta GPUs de forma nativa

Otra opción también madura sería CNTK (de **Microsoft**)

Respecto a Keras, está considerada una librería open source mantenida principalmente por François Chollet, un ingeniero de Google. En vez de ser un framework de Deep Learning, Keras fue diseñado para actuar como una interfaz simple y amigable para los desarrolladores, independientemente del framework de deep learning que se tuviera ejecutando. Actualmente permite correr Tensorflow, CNTK, Theano y PlaidML

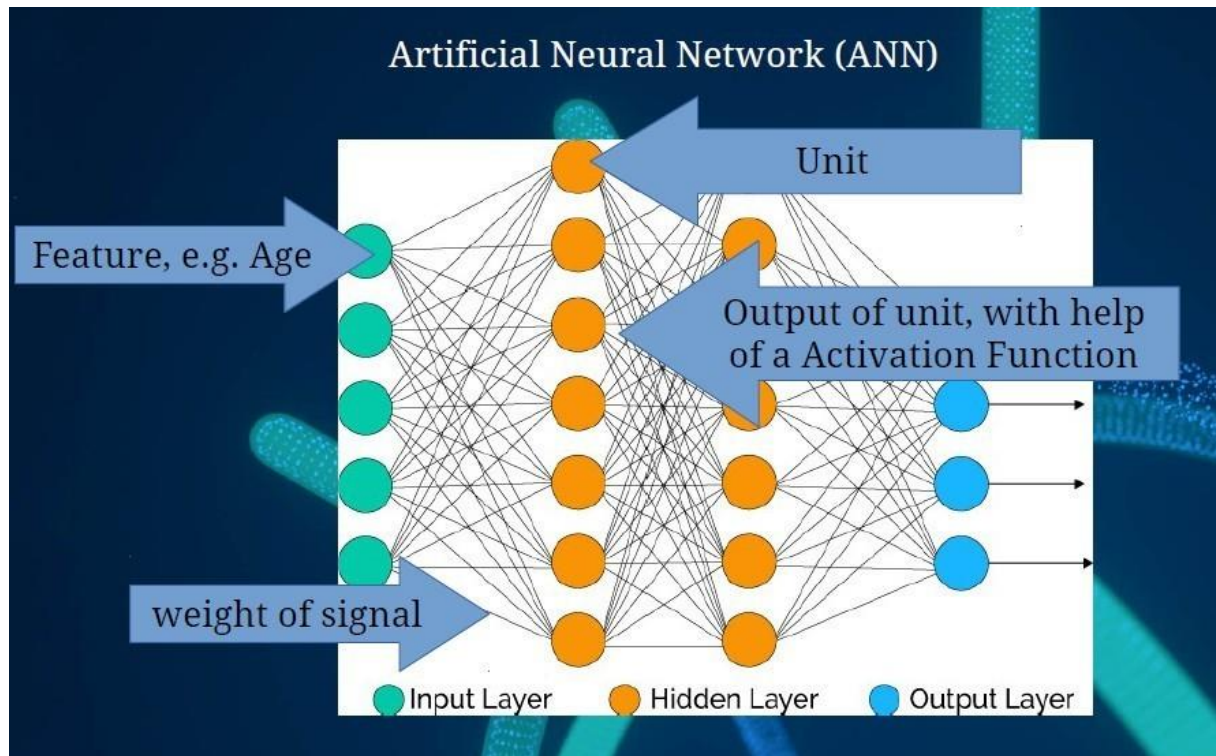
¿Qué es una red neuronal?

A estas alturas del curso, todos tenemos que tener bastante claro de qué se trata una red neuronal. No obstante, nunca viene mal realizar un repaso y entender donde se sitúa.



Una Red Neuronal Artificial (RNA) es un sistema de nodos interconectados de forma ordenada, distribuido en capas, a través del cual una señal de entrada se propaga para producir una salida. ~~Se conocen así porque pretenden emular de forma sencilla el funcionamiento de las redes neuronales biológicas que se encuentran en el cerebro animal.~~ Esta definición es errónea ya que el entendimiento del funcionamiento de las redes neuronales biológicas ha ido cambiando con el tiempo. Por lo tanto es mejor, dejar de explicar las redes neuronales con esta definición. Constan de una capa de entrada, una o varias capas ocultas y una capa de salida y se las puede entrenar para que "aprendan" a reconocer ciertos patrones. Esta característica es la que las incluye dentro del ecosistema de tecnologías conocidas como inteligencia artificial.

Las RNAs tienen varias décadas de historia, pero han cobrado gran relevancia recientemente debido a la disponibilidad de las grandes cantidades de datos y de la potencia de computación necesaria para su utilización en problemas complejos. Han supuesto un hito histórico en aplicaciones tradicionalmente esquivas a la programación clásica, basada en reglas como el reconocimiento de imágenes o de voz. Sus elementos fundamentales son los tensores, que se pueden equiparar con vectores de una o varias dimensiones.



Una buena forma de entender una red neuronal es la siguiente (contenido creado por IBM). Abrir la presentación Power Point de [Explicación red neuronal.pptx](#).

Cómo ya habéis visto, las arquitecturas más destacadas de Deep Learning son:

- Redes convolucionales (CNN)
- Redes recurrentes (RNN)
- Redes contrarias generativas (GAN - Generative Adversarial Networks)
- Redes de aprendizaje por refuerzo (Reinforcement Learning)

En las lecturas opcionales (Actividad 3), se comparten materiales para entender cada una de las arquitecturas anteriores, ya que poseen unas características concretas así como casos de uso particulares. No obstante, en los ejercicios prácticos se utilizan diversas de estas arquitecturas.

4. Instalación

Todo este módulo va a ser ejecutado a través de Python. Como ya se ha utilizado en otros módulos, los jupyter notebooks son una herramienta muy útil. Para trabajar de manera sencilla se recomienda la creación de “[virtual environments](#)” para Tensorflow y PyTorch (Keras se incluye dentro de la de Tensorflow). También se recomienda la ejecución de Python por la distribución de [Anaconda](#). Por simplicidad en la instalación y en los ejercicios que vamos a realizar vamos únicamente a instalar las distribuciones de CPU. En caso de que se posea una tarjeta gráfica y CUDA, se podrá ejecutar Tensorflow y PyTorch con ella siguiendo los siguientes tutoriales de instalación:

- <https://www.quantinsti.com/blog/install-tensorflow-gpu>
- `$ conda install pytorch torchvision cudatoolkit=9.0 -c pytorch`

Un buen setup podría ser tener un virtual environment para cada tipo de instalación evitando así cualquier problema de compatibilidades. Por ejemplo (los comandos están para Windows. Varían un poco para Linux o MAC pero son similares):

```
# upgrade conda
$ conda upgrade conda

# create a new environment for Tensorflow (CPU)
$ conda create -n tf python=3.6

# create a new environment for Pytorch (CPU)
$ conda create -n pytorch python=3.6

# List of current environments
$ conda info --envs

# activate the environment tf
$ activate tf

# install pip
$ conda install pip

# install latest version of Tensorflow
$ pip install --upgrade tensorflow

# install KERAS and matplotlib
$ conda install keras, matplotlib, h5py

# installing ipykernel
$ pip install ipykernel
```

```
$ ipython kernel install --user --name=tf

# deactivate tf environment and activate pytorch
$ deactivate
$ activate pytorch

# First, install numpy and matplotlib
$ conda install numpy, matplotlib, h5py

# First, install numpy
$ conda install pytorch-cpu torchvision-cpu -c pytorch

# installing ipykernel
$ pip install ipykernel
$ ipython kernel install --user --name=pytorch
```

5. A programar

Los ejercicios están separados por tecnologías. Tensorflow, Keras y Pytorch. En esas carpetas, el objetivo fundamental es entender cómo funciona el framework de programación. Para ello, se toma como ejemplo base, realizar un ejercicio de clasificación, ya sea con el dataset MNIST de números escritos a manos o el CIFAR10 para clasificar objetos. **Este es el objetivo fundamental del módulo y lo más importante es entender y realizar esos ejercicios.**

Adicionalmente, se proporciona una última carpeta con problemas extras que intentan enseñar otras topologías de redes neuronales y usos. Para entender mejor que se realiza en esos ejercicios es importante leer las lecturas opcionales. Aunque cada alumno es libre de realizar los ejercicios de la manera que vea más conveniente, lo que se recomienda es:

- Empezar con las lecturas opcionales de que es una red convolucional. Esto es importante, porque es la arquitectura de red que se utiliza para los problemas de clasificación de los frameworks. Simplificará la tarea de entendimiento.
- Una vez entendido, realizar los ejercicios resueltos de los 3 frameworks, en el siguiente orden PyTorch, Tensorflow y Keras
- Realizar los ejercicios a entregar sobre PyTorch, Tensorflow y Keras
- Continuar con el resto de lecturas opcionales sobre GAN, Reinforcement Learning y Redes Recurrentes (RNN)
- Realizar los ejercicios resueltos extras para ver otros casos de uso