Escuela de organización industrial

Taller práctico de Deep Learning (DL)

Javier Abascal Carrasco

Índice

01

Objetivos de la sesión

Introducción

El objetivo principal es que el alumno entienda que son exactamente las redes neuronales y cuáles son los mejores *frameworks* existentes en el mercado para trabajar con las mismas

Para ellos vamos a utilizar la metodología de “aprender haciendo”

Nos vamos a ayudar de unos ejercicios de ejemplo y unos ejercicios propuestos que tendrán que ser realizados para ser entregados

02

Profesor
– Datos de contacto

Introducción

Estudios
- Ingeniero de Telecomunicación – Universidad de Sevilla
- MBA – Thomas College, ME, USA
- Máster en Big Data – U-Tad, Madrid

Trabajo Pasado
- Consultor Big Data en Deloitte y Ernst & Young
- Ingeniero de datos en Gennion (start-up) y Privitar (start-up)

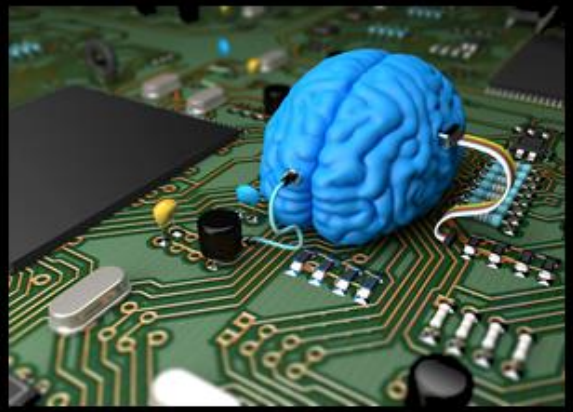
Trabajo Actual
- Ingeniero de Datos en Facebook – equipo de WhatsApp, Londres (disculpad anglicismos)
- Ilusionista – (hobby)

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

Contacto:
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.
```

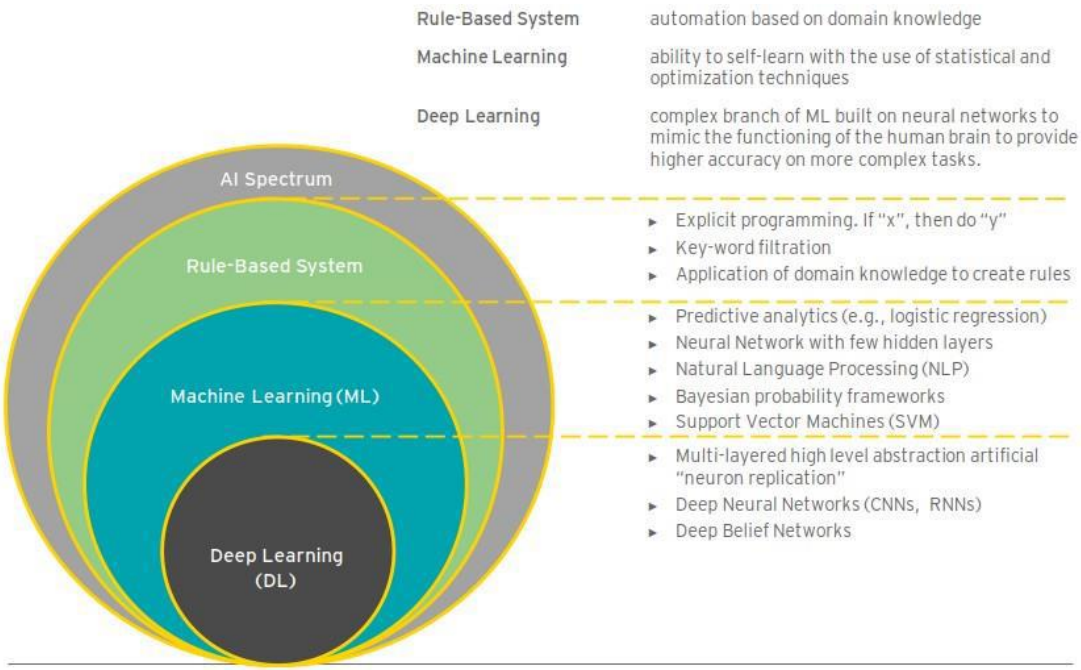
What I actually do

03

Historia del Deep Learning

Deep Learning

- 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.
- Siempre es importante entender en el contexto donde nos movemos, por ello, la imagen de la derecha ayuda a entender que Deep Learning no es más que un subconjunto de machine Learning dentro del ecosistema de inteligencia artificial
- NOTA: No hay un acuerdo ”oficial” sobre estas categorizaciones, esto es simplemente una de las perspectiva existentes

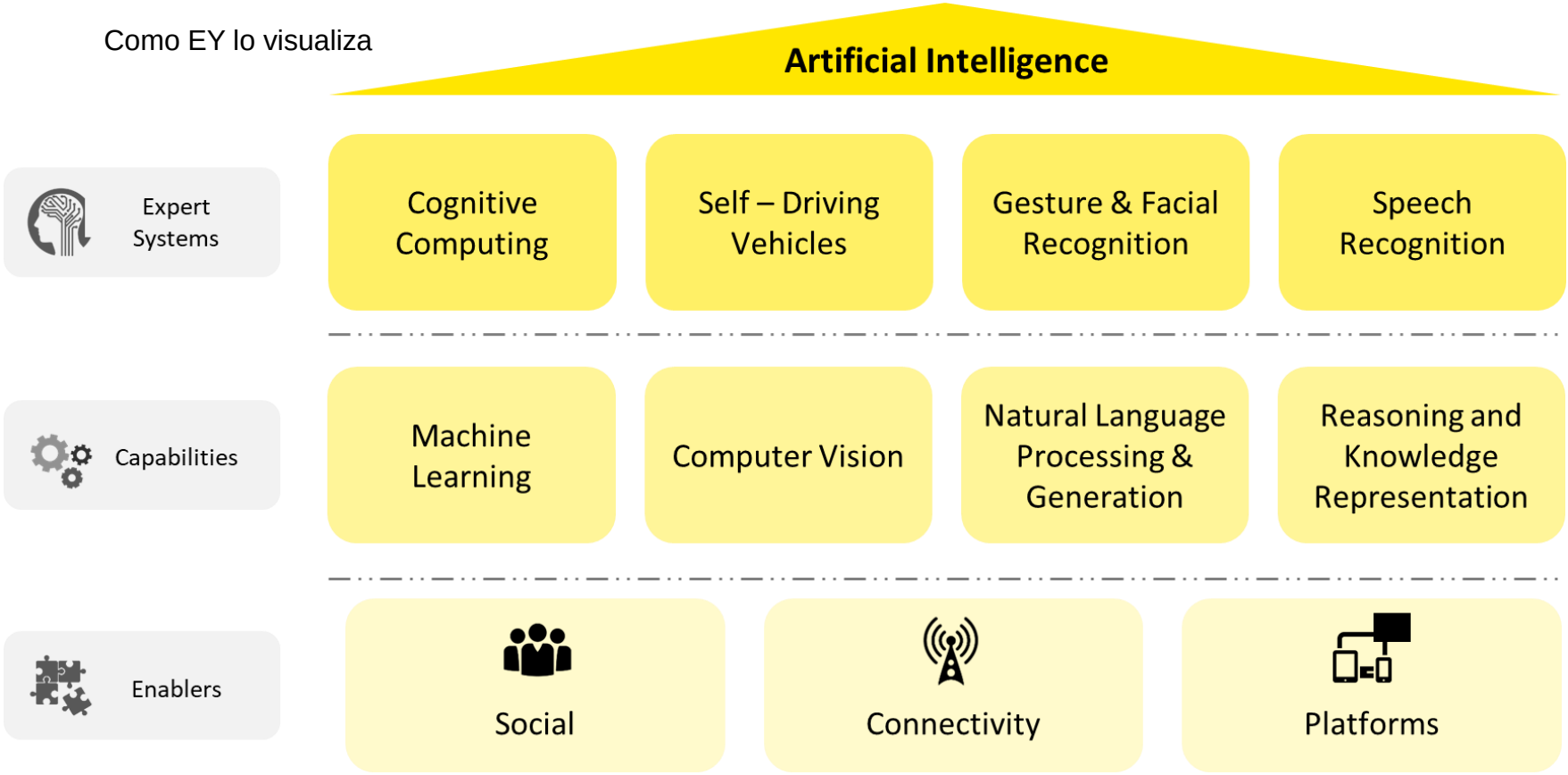


Deep Learning es una subcategoría de ML

Deep Learning

¿Por qué ha surgido esta explosión del ML y DL?

Como EY lo visualiza



New Capabilities and Expert Systems may appear in the future. They may also re-categorize in different/new blocks

Deep Learning

¿Por qué ha surgido esta explosión del ML y DL?

AI Enablers

Social

- Privacy
- Open Source
- Sharing
- Social Networks

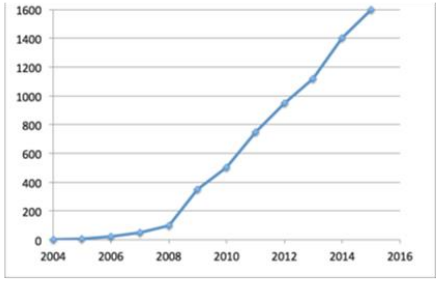
Connectivity

- Multimedia
- 4G Mobile Networks
- Geospatial
- Sensors IoT

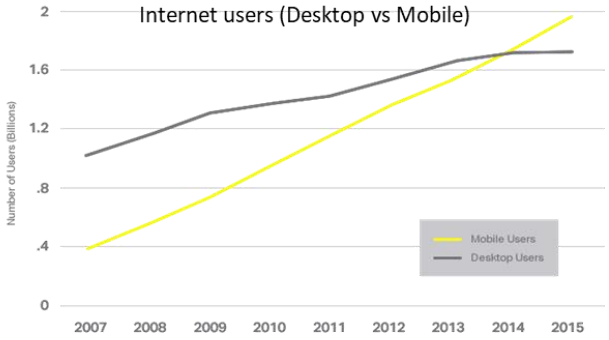
Platforms

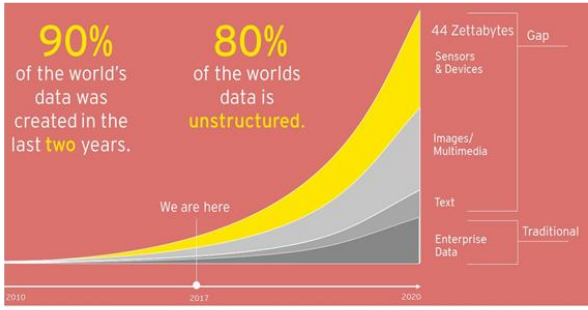
- Cloud Computing
- Massive Storage
- Hadoop
- Smartphones

Facebook million of users



Internet users (Desktop vs Mobile)





Deep Learning

¿Por qué ha surgido esta explosión del ML y DL?

AI Capabilities / Expert Systems

Machine Learning

- Rule System
- Deep Learning
- Neural Networks
- Regularization
- Ensemble
- Clustering
- Regression
- Bayesian
- Decision Tree
- Dimensionality Reduction
- Instance based

Computer Vision

- Image recognition
- Pattern recognition
- RNN
- Motion analysis
- Scene reconstruction

Natural Language Processing & Generation

- Text Mining
- Bag of words
- Word2vec / doc2vec
- Sentiment Analysis

Reasoning and Knowledge Representation

- Interface engines
- Inference engines
- Decision-rule engines
- Ontology engineering
- Knowledge graphs

Cognitive Computing

- Medical Diagnosis Robot
- Chat Bots
- Cognitive Robotics
- RPA
- Personal Recommendation Engines

Self – Driving Vehicles

- Self – Driving Cars
- Self – Driving Trucks
- Unmanned Aerial Inspections
- Drone

Gesture & Facial Recognition

- Virtual Reality
- Biometrics (Identification)

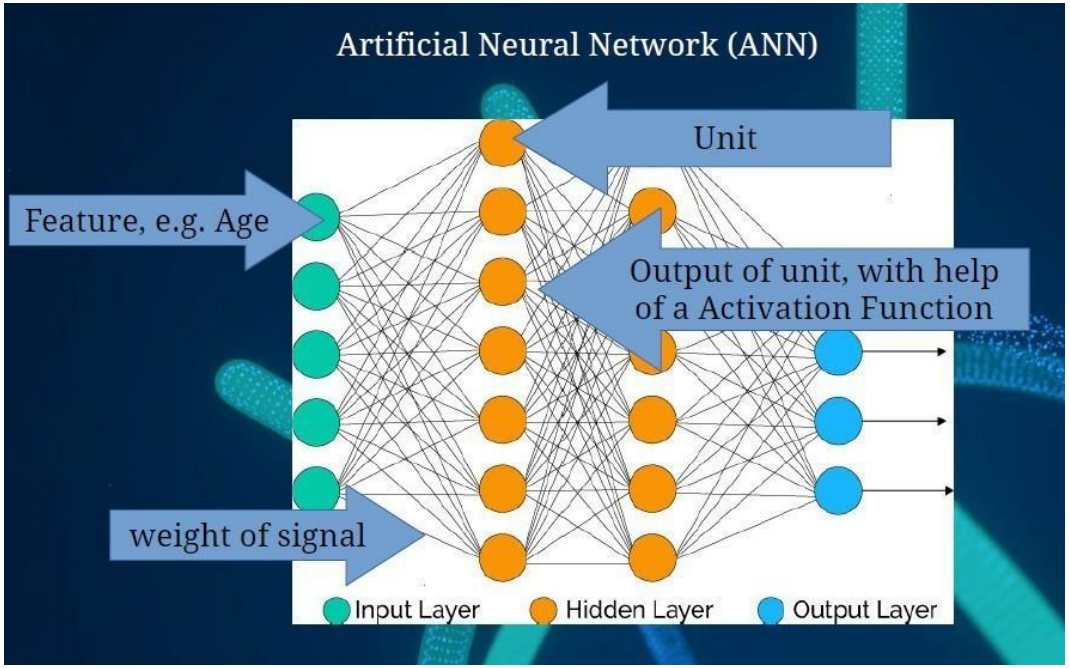
Speech Recognition

- Virtual Assistant
- Interactive Voice Response (Call Center)

Deep Learning

¿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.

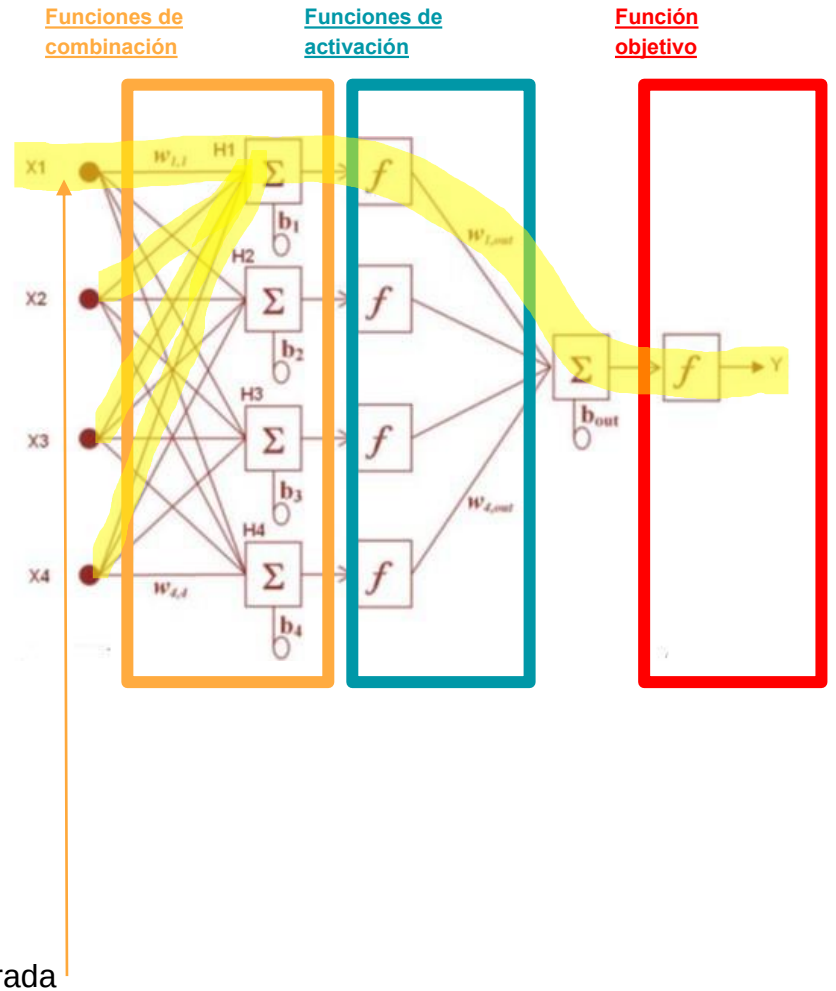
Debemos trabajar en crear una abstracción práctica de una red neuronal

Deep Learning

Sin entrar en detalles matemáticos, una buena forma de entender una red neuronal es viéndola como un "túnel" que va modificando los valores de entrada para generar unos valores de salida

Los valores iniciales se modifican a través de tres funciones (combinación, activación y función objetivo) para aprender el algoritmo que definirá el modelo que se quiere realizar. La dificultad del algoritmo es calcular los pesos o "weights" para que funcionen de manera adecuada. Es decir, los algoritmos aprenden a través la modificación de los pesos "weights" de las neuronas y decidir que funciones de combinación, activación y objetivo elegir

Una red neuronal no es más que un sumatorio muy muy grande de regresiones lineales (ya hay hasta millones de parámetros)



- Ejemplo de una de las ramas de la red neuronal mostrada

Función_objetivo(Función_Activación(Función_Combinación($x\_1 \cdot w\_1,1$, $x\_2 \cdot w\_1,2$, $x\_3 \cdot w\_1,3$, $x\_4 \cdot w\_1,4$)) $\cdot w\_(1,out)$)

Por favor, intentad crear una abstracción práctica de una red neuronal cuanto antes

Deep Learning

Funciones de combinación

TABLE OF COMBINATION FUNCTIONS

FUNCTION	DEFINITION
Additive	$\sum_i x_i$
Linear	$b_j + \sum_i w_{ij} x_i$
EQSlopes	$b_j + \sum_i w_i x_i$
XRadial	$f \log(a_j) - b_j^2 \sum_i (w_{ij} - x_i)^2$
EHRadial	$-b_j^2 \sum_i (w_{ij} - x_i)^2$
EVRadial	$f \log(b_j) - b_j^2 \sum_i (w_{ij} - x_i)^2$
EWRadial	$f \log(a_j) - b^2 \sum_i (w_{ij} - x_i)^2$
EQRadial	$-b^2 \sum_i (w_{ij} - x_i)^2$

Funciones de activación

TABLE OF ACTIVATION FUNCTIONS

FUNCTION	RANGE	FUNCTION OF NET INPUT *g*
Identity	$(-\infty, +\infty)$	g
Exponential	$(0, \infty)$	$\exp(g)$
Reciprocal	$(0, \infty)$	$1/g$
Square	$[0, +\infty)$	g^2
Logistic	$(0, 1)$	$\frac{1}{1+\exp(-g)}$
Softmax	$(0, 1)$	$\frac{\exp(g)}{\sum exponentials}$
Gauss	$(0, 1]$	$\exp(-g^2)$
Sine	$[-1, 1]$	$\sin(g)$
Cosine	$[-1, 1]$	$\cos(g)$
Elliott	$(0, 1)$	$\frac{g}{1+\lvert g \rvert}$
Tanh	$(-1, 1)$	$\tanh(g) = 1 - \frac{2}{1+\exp(2g)}$
Arctan	$(-1, 1)$	$\frac{2}{\pi} \arctan(g)$

Función objetivo

Las funciones objetivo es lo que mapea la salida de la red neuronal con el objetivo de la red que se está construyendo.

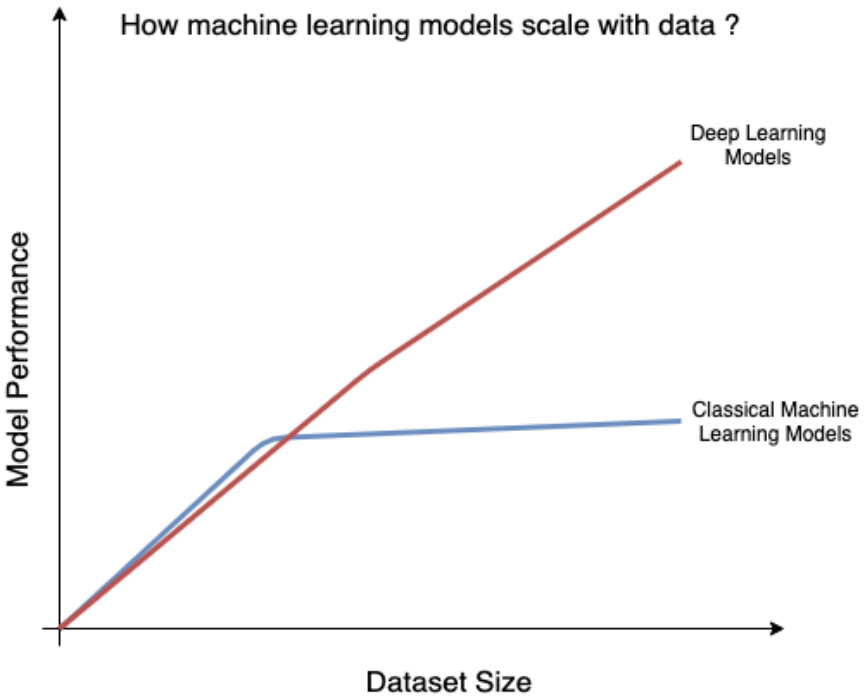
Por ejemplo, si se está realizando una red de clasificación, se tendrá que realizar un mapeado de la salida de las funciones de activación (que puede ser n variables) a el número de clases que se desea clasificar. Esto se observará muy bien en los ejercicios de ejemplos

La experiencia empírica mezclada con análisis e intuición matemática ha permitido desarrollar los ejemplos actuales

Deep Learning

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.



Las arquitecturas más destacadas de Deep Learning son:

- Redes convolucionales (CNN)
- Redes recurrentes (RNN) – los modelos “encoder-decoder” son la combinación de CNN con LTSM
- Redes contrarías generativas (GAN – Generative Adversarial Networks)
- Redes de aprendizaje por refuerzo (Reinforcement Learning)

La IA está aún algo “verde” para ser capaz de generalizar, pero se ha convertido en una experta en problemas concretos

04

Frameworks de DL

(Tensorflow, PyTorch y Keras)

Instalación de los Frameworks

- Todo este módulo va a ser ejecutado a través de Python. Se asume que el alumno está más o menos familiarizado con Python. 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 (chequear la compatibilidad de la tarjeta gráfica con CUDA), se podrá ejecutar Tensorflow y PyTorch con ella siguiente 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.

Instalación de los Frameworks

- 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, aunque con TF 2.0 y Keras la abstracción funciona muy fácilmente
- 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

Instalación de los Frameworks

- 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

05

Entendimiento de las redes neuronales

Arquitectura de la red

Las diferentes arquitecturas han sido descubiertas a lo largo del tiempo mediante estudios y ensayo/error

Las arquitecturas de redes neuronales se construyen añadiendo diferentes capas. Por ello, lo más importante es entender qué es lo que hace cada capa. A continuación vamos a listar las capas más comunes que existen, resumiendo brevemente que es lo que hacen y añadiendo enlaces para entenderlos mejor. Una vez entendidas, los ejercicios incluirán diferentes ejemplos en los diferentes *frameworks* de código

Tutorial completo desde 0 en esta lista de reproducción sobre qué es una red neuronal (útil si no se tiene ningún conocimiento sobre redes neuronales)

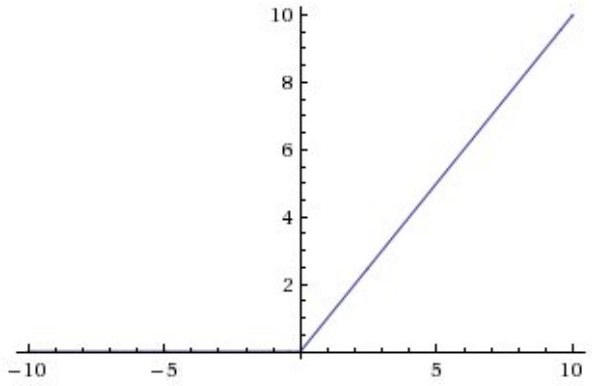
https://www.youtube.com/watch?v=3fmSmydiuvQ&list=PLMF23FOyQZQskct_KyHdIBePhMPXa809j

Se recomienda leer los enlaces y ver los vídeos para entender los conceptos que se van introduciendo (es complejo)

Funciones de Activación (algunas de las más importantes)

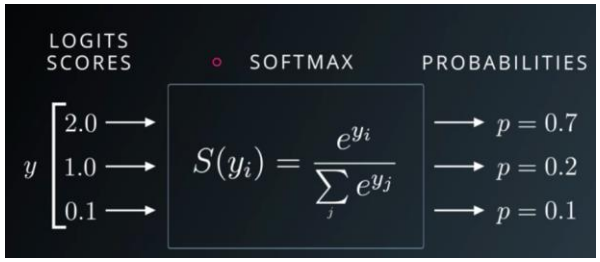
RELU (no lineal)

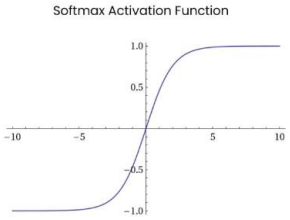
Una función muy sencilla pero que se ha demostrado muy útil para acelerar el entrenamiento de las redes neuronales. Matemáticamente es muy sencillo, 0 si x<0 y x si x>0. Gráficamente se puede representar como



SOFTMAX (no lineal)

Una función sencilla para normalizar la salida y asignar una función de probabilidad sobre los valores en las redes de clasificación. Es comúnmente utilizado como última capa de la red





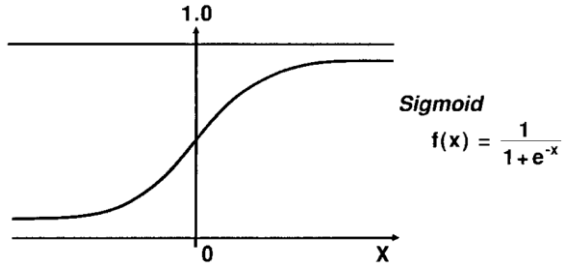
Arquitectura de la red

Las diferentes arquitecturas han sido descubiertas a lo largo del tiempo mediante estudios y ensayo/error

Funciones de Activación

SIGMOID (no lineal)

Muy similar a la función SOFTMAX pero se utiliza únicamente para clasificación binaria. Fija los valores entre 0 y 1. Esto también ayuda a que los valores internos de la red no se disparen (normalizando) ya que un valor muy elevado se transforma en 1.



Enlaces para entender bien las funciones de activación

Vídeos

https://www.youtube.com/watch?v=lFODTDO8mMw
https://www.youtube.com/watch?v=N3jqMX2DADY

Lecturas

https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/
https://www.diegocalvo.es/funcion-de-activacion-redes-neuronales/

Diferentes Capas de Red

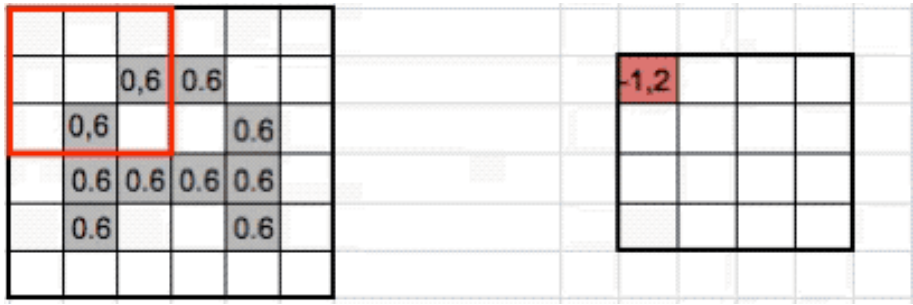
CAPA LINEAL

Representa una capa lineal de neuronas. Suelen ser utilizadas al principio y al final de las redes como capa de adición. Matemáticamente es algo tan simple como una ecuación lineal que todos conocemos y = ax + b

CAPA CONVOLUCIONAL + MAX POOLING

Representa una capa que aplica un filtro convolucional a los datos de entrada. Esta operación consiste en aplicar un producto escalar a los datos de entrada con una pequeña matriz (kernel) que será de una dimensión 3x3, 5x5, etc. Las capas convolucionales son muy efectivas en visión por ordenador y cada capa puede especializarse en diferentes jerarquías de la imagen (unas pueden detectar líneas, otras curvas y se van especializando. Finalmente, las capas más profundas pueden llegar a detectar cosas complejas como rostros o animales).

Ejemplo de operación de convolución con un kernel 3x3 a una imagen

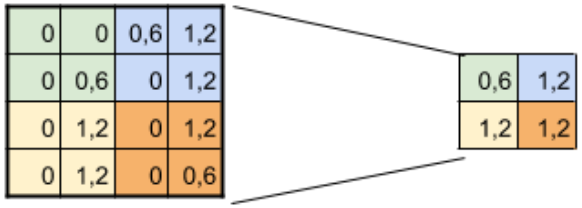


Finalmente, tras realizar una operación de convolución, se pueden añadir diferentes capas de MAX POOLING. Estas capa filtra los datos eligiendo el mayor valor de un kernel AxA, reduciendo el tamaño final de la matriz de entrada. Estas operaciones dependen del tamaño del Kernel, el Padding y el Stride (conceptos explicados en el Vídeo 3)

Arquitectura de la red

CAPA CONVOLUCIONAL + MAX POOLING (cont.)

Imagen de Max Pooling con un kernel de 2x2



SUBSAMPLING:
Aplico Max-Pooling de 2x2
y reduzco mi salida a la mitad

Enlaces para entender bien las redes convolucionales y el max pooling

https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/

Video 1 , Video 2 y Vídeo 3

Diferentes Capas de Red

CAPA DROPOUT

Una capa de *Dropout* desactiva un numero de neuronas de una red neuronal de forma aleatoria. En cada iteración de la red neuronal *dropout* desactivara diferentes neuronas, las neuronas desactivadas no se toman en cuenta para el *forwardpropagation* ni para el *backwardpropagation* lo que obliga a las neuronas cercanas a no depender tanto de las neuronas desactivadas. Este método ayuda a reducir el *overfitting*

CAPA BATCH NORMALIZATION

Batch normalization es un método que normaliza cada lote de datos (bath_size), para evitar que se tengan distancias muy diferentes entre ellos como en una imagen a color que se pueden tener valores de 0 hasta 255. Normalizando los datos las distancias de los datos van de 0 a 1 y esto ayuda a la red neuronal a trabajar mejor y a tener menos problemas, pero cuando normalizamos los datos solo la capa de entrada se beneficia de esto, conforme los datos pasan por otras capas ocultas esta normalización se va perdiendo y si tenemos una red neuronal con muchas capas podemos tener problemas con el entrenamiento. El metodo de batch normalization normaliza los datos antes de que pasen por la función de activación en cada capa que de la red neuronal, de esta manera siempre tendremos los datos normalizados.

Enlaces para entender bien las capas DROPOUT y BATCH_NORMALIZATION

https://vincentblog.xyz/posts/dropout-y-batch-normalization

https://www.youtube.com/watch?v=QyVr3UiTYXg

https://www.youtube.com/watch?v=dXB-KQYkzNU&ab_channel=deeplizard

Arquitectura de la red

CAPA RECURRENTE

Representa una capa que introduce un bucle/ciclo en una red neuronal. Es decir, esta capa ”añade recursividad” en la arquitectura. Ese diseño, hacen que puedan «recordar» estados previos y utilizar esta información para decidir cuál será el siguiente. **Esta característica las hace muy adecuadas para manejar series cronológicas**

Las redes recurrentes (RNN) son comúnmente utilizadas en todo lo que conlleve una serie de eventos o datos como NLP, speech recognition, video tagging, generation of image description, translation, etc.

Existen diferentes tipos de capas recurrentes pero las denominadas LTSM (Long Short-Term Memory) han tenido mucho éxito ya que permiten disponer de “una memoria a largo plazo” permitiéndoles recordar relaciones muy distantes.

Enlaces para entender bien las capas recurrentes

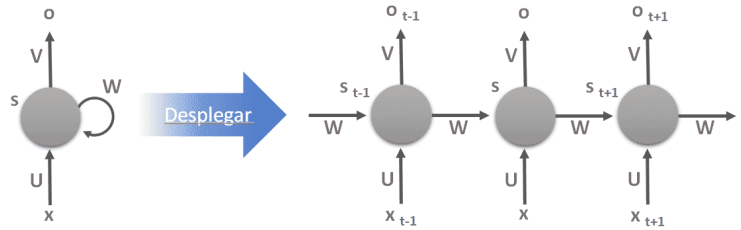
https://www.diegocalvo.es/red-neuronal-recurrente/

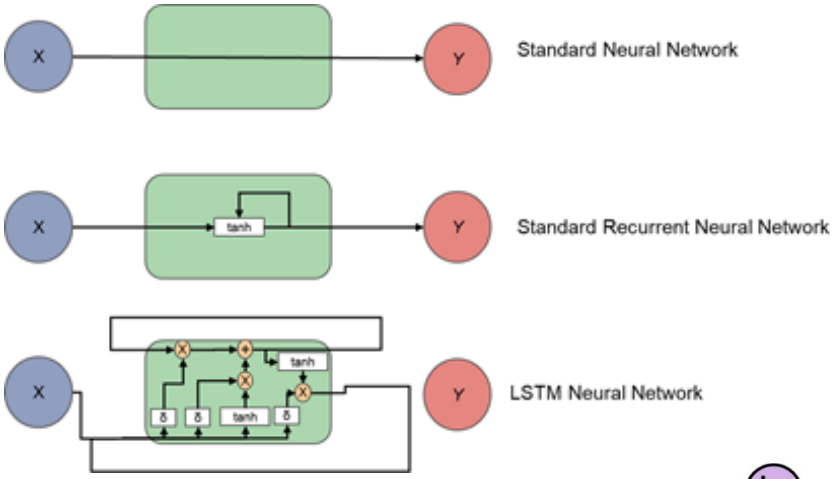
https://www.youtube.com/watch?v=1BubAvTVBYs

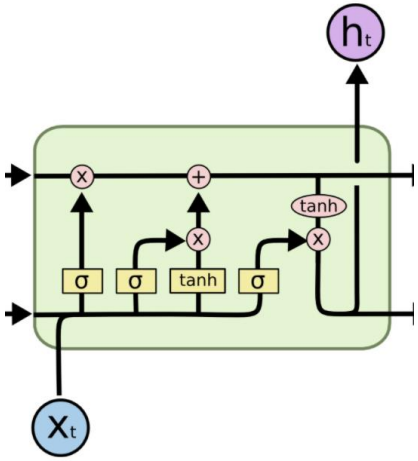
https://www.youtube.com/watch?v=715RuKtX1Xc&ab_channel=AMPTech

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Un bucle no es más que una serie cronológica multiplicada por un peso t+1, t+2, etc.







Resto de Pasos que implementaremos

- **RUTINA DE ENTRENAMIENTO**

Una vez definamos nuestra red, tendremos que definir como vamos a realizar el conjunto de entrenamiento. Esta función será la encargada de realizar el "*backpropagation*" de la red para ir actualizando los pesos de las neuronas y configurar los valores iniciales

- **RUTINA DE INFERENCIA**

Una vez definido el entrenamiento, también tendremos que definir como vamos a realizar la inferencia o comprobación de nuestro modelo. Es decir, como utilizamos la red que se ha entrenado para aplicarla sobre el conjunto de TEST y calcular el acierto o fallo de la red

- **DATA LOADERS**

Para acelerar el entrenamiento, en vez de hacerlo de 1 en 1 lo vamos a realizar en *batches* de X elementos. El tamaño del *batch* tendrá que ser acorde con el volumen de datos que disponemos y con la potencia de nuestro ordenador.

Los *frameworks* ayudan con funciones específicas para aplanar y gestionar los *data loaders* de manera correcta

OPTIMIZADOR

Al definir la rutina de entrenamiento, deberemos de añadir también como queremos actualizar los pesos. Existen diferentes optimizadores que hacen que la red converja más rápido dependiendo del caso de uso que estemos resolviendo. Ejemplos:

- SGD (with or without momentum)
- AdaDelta
- AdaGrad
- RMSProp
- Adam

https://datascience.stackexchange.com/questions/10523/guidelines-for-selecting-an-optimizer-for-training-neural-networks

https://towardsdatascience.com/the-3-best-optimization-methods-in-neural-networks-40879c887873

ENTRENAMIENTO

Finalmente, definiremos los *epochs*, para entrenar y evaluar el modelo

Ejercicios de ejemplo

El módulo contiene diferentes ejercicios resueltos que implementan diversos tipos de redes neuronales en los diferentes *frameworks* de trabajo. La idea es que los carguéis y los ejecutéis en vuestro ordenador.

Dependiendo de vuestro ordenador, el entrenamiento de la red puede durar mucho tiempo, por lo que no se pide que lo ejecutéis entero, pero que vayáis leyendo el código, entendiendo las diferentes capas y fases que se desarrollan en base a la teoría que hemos explicado

Gracias.