“¡Hola mundo!” de las Redes Neuronales Artificiales

Encuentro 2

Ramiro M. Irastorza

Departamento de Mecánica
UTN

GMG-UTN-2020

Resumen

Resumen

AI, Machine learning, Deep learning

- En sus comienzos la **Inteligencia Artificial** (AI) comenzó por resolver problemas difíciles para los humanos pero fáciles para las computadoras (se podían escribir con reglas matemáticas más o menos claras).
- Luego fue cambiando y el desafío de la AI es resolver problemas que los humanos pueden resolver de manera casi intuitiva (una vez que aprendieron), tales como, reconocimiento de voz, caras en una imagen, etc.

AI, Machine learning, Deep learning

- Con la AI se puede tratar de codificar cosas que son intuitivas para el humano, pero muy complejas para traducir a reglas de una computadora, como reconocer una letra en una imagen.
- Otro enfoque es tratar que la AI adquiera su propio conocimiento, extrayendo patrones de datos crudos (datos de entrenamiento). Esta capacidad es conocida como **Machine Learning**.
- Ejemplos: la regresión logística, clasificador bayesiano naive (ingenuo), etc.
- Ejemplo: Modelo de Markov Oculto.
- Su desempeño depende fuertemente de la **representación** de los datos con los que aprendió. Cada elemento con el que aprende tiene una característica (feature), está fuertemente correlacionada con la salida del modelo funcionará bien... si no, no.

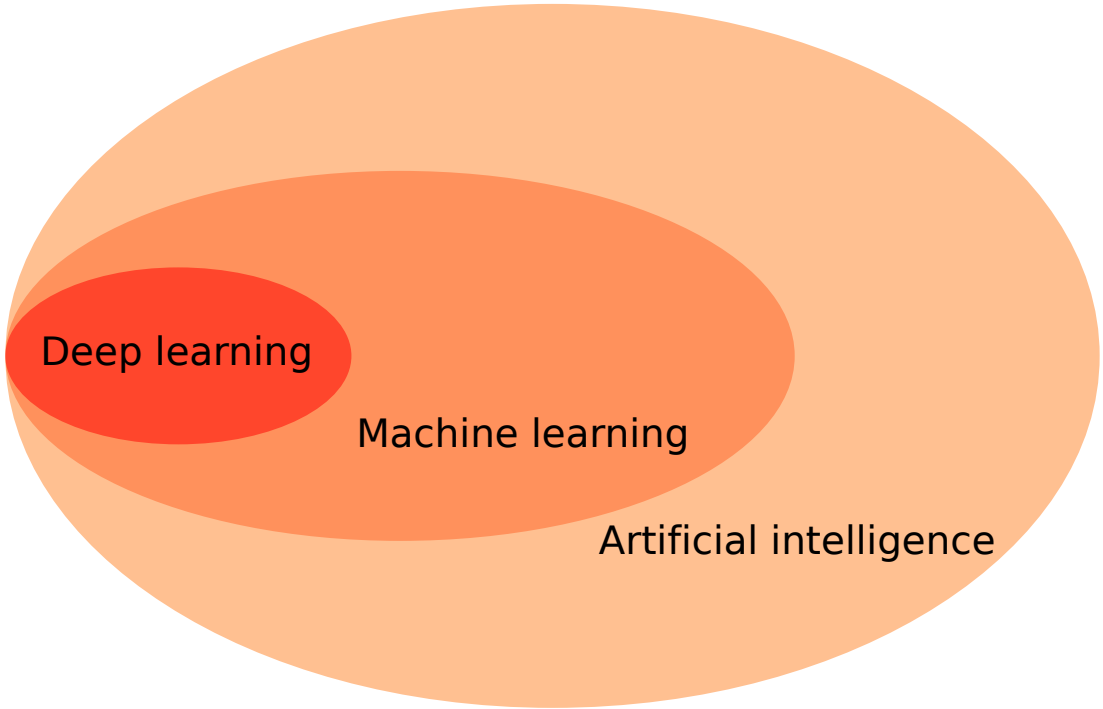
AI, Machine learning, Deep learning

- Si a Machine Learning, se le pide que aprenda también la representación, es decir, que detecte las características que están correlacionadas con la salida, entonces se le dice **Representation Learning**.
- Ejemplo: Autoencoder.
- Cuando se quieren aprender las características lo que buscamos es reconocer y clasificar los **factores de variación**. Por ejemplo: cuando se analiza la voz, los factores de variación incluyen: la edad del que habla, el género, su acento, y las palabras que está hablando.

AI, Machine learning, Deep learning

- Si tomamos otro ejemplo, reconocer autos en una imagen. Si estamos interesados en detectar el color, habrá factores de variación que serán muy difíciles de aprender. Por ejemplo, de noche un pixel rojo puede parecerse mucho a un pixel negro.
- Ejemplos como estos, hacen que el Representation Learning no sea tan existoso.
- Este problema lo salva el **Deep Learning**, que introduce representaciones que son expresadas en términos de otras más simples.
- Ejemplo: Perceptrón Multi Capa (MLP).

AI, Machine learning, Deep learning



Herramientas de esta presenatación

- Keras: biblioteca de código abierto escrita en Python (corre sobre TensorFlow o Theano), una **interfaz** en lugar de ser una framework de machine learning standalone. ¡Se lanzó en marzo de 2015!.
- TensorFlow: es una biblioteca de código abierto para aprendizaje automático de Google. Su predecesor fue DistBelief (de código cerrado). Fue desarrollado por Google Brain para uso interno en Google, luego se publicó bajo licencia de código abierto.
- Para empezar se puede correr TensorFlow utilizando Colab: seguir enlace.

Pasos

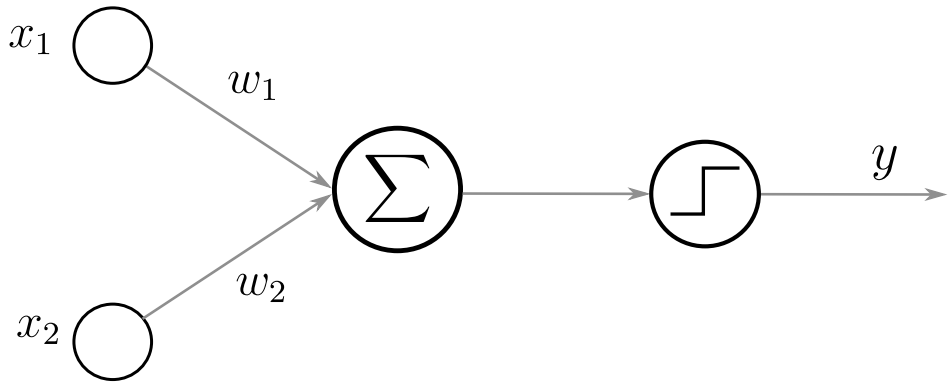
En un procedimiento clásico de obtención de una red neuronal se pueden seguir los siguientes pasos:

1. **Pre-procesar los datos** para normalizar las entradas, y estandarizar la respuesta.
2. Establecer la **arquitectura** de la red, incluyendo la cantidad de capas ocultas, en número de nodos en cada capa oculta, los bias, etc.
3. Separar los datos en: **entrenamiento, testeo, y validación**.
4. **Iniciar** los pesos para el algoritmo de retropropagación.
5. Establecer un **criterio para detener** el entrenamiento.

Resumen

Perceptrón

- Es un tipo de neurona artificial



Si x_i es una entrada (binaria, 0 o 1) y w_i es el peso que se le da a esa entrada entonces la salida de la neurona será:

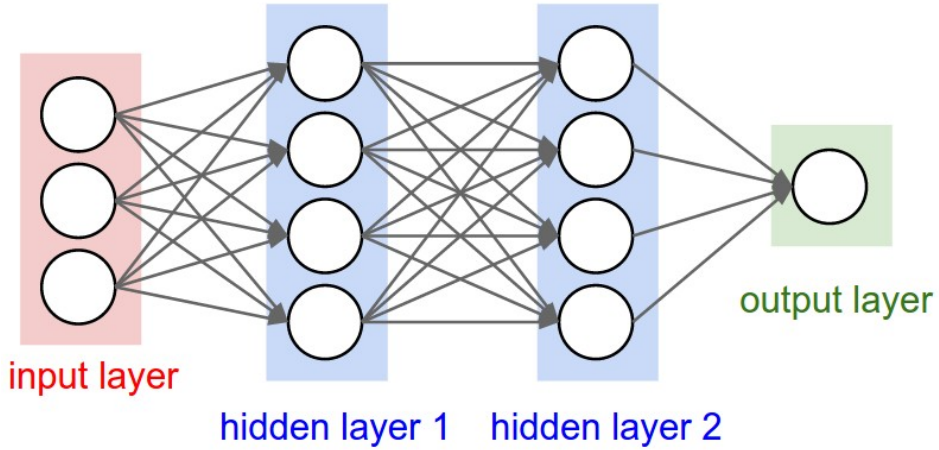
$$y(x) = \begin{cases} 1 & \text{si } \sum_i w_i x_i + b = \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{caso contrario} \end{cases} \tag{1}$$

aquí b es el bias (o umbral negado). A la Ec. 1 se la llama **función signo**.

Topología de la red

Topología o arquitectura de la red: **Perceptrón Multicapa**

- Red **completamente conectada**
  Cada nodo corresponde a una neurona, y existe un vector de entrada y uno de salida. Las capas que no contactan con ninguno de ellos se llaman capas ocultas. Cada nodo recibe entradas de todas las neuronas de la capa anterior (o del vector de entrada).



Función de costo

- **Entropía cruzada**. Es la función de pérdida predeterminada que se usa para problemas de clasificación binaria. Los valores objetivo están en el conjunto $\{0, 1\}$.
- Cuando se usa Entropía cruzada multi **clase** (es el ejemplo que vamos a ver, hay 10 clases en lugar de 0 o 1) se utiliza como función de activación en la última capa la función **softmax**[1]. La función nos da la probabilidad de cada clase.

[1] http://numerentur.org/funcion-de-activacion-softmax/

Retropropagación de errores

Idea general:

- Los algoritmos lo que hacen es actualizar iterativamente:

$$\mathbf{w} \longleftarrow \mathbf{w} - \epsilon \frac{\partial C}{\partial \mathbf{w}} \tag{2}$$

$$\mathbf{b} \longleftarrow \mathbf{b} - \epsilon \frac{\partial C}{\partial \mathbf{b}} \tag{3}$$

  aquí ϵ se la llama **tasa de aprendizaje** y se utiliza para modificar los pesos de cada neurona.
- Se denomina **algoritmo del gradiente decreciente**.

SGD

- El algoritmo de gradiente decreciente no es aplicable, mucho dato muchas cuentas!
- Se soluciona con el: **stochastic gradient descent** (SGD).
- Calcula el gradiente pero no para todos los datos de entrenamiento, sino para una pequeña muestra (**mini-batch**).
  - Batch o mini batch son submuestras del conjunto de datos tomadas sin reemplazo. Sobre cada una se calcula el vector de gradientes.
  - Época o **epoch** designa un ciclo de cálculo en el que se agotaron todos los elementos del conjunto de datos, es decir, que todos los elementos formaron parte de una submuestra.

Resumen

Hello world

Haremos un ejemplo con Jupyter Notebook utilizando **Tensorflow**, reconociendo números escritos a mano.

- Lo saqué de acá, escrito por Aymeric Damien.
- Un set de datos clásico
  El **MNIST** (Modified National Institute of Standards and Technology) es un conjunto de 60.000 dígitos escritos a mano por diferentes personas de los EEUU, originalmente se generó el NIST en 1995, y luego se publicó una versión modificada.
- Imágenes en escala de gris de 28×28, genera vectores de entrada de 784 elementos.

Referencias

1. http://neuralnetworksanddeeplearning.com/
2. Y. Bengio, I. J. Goodfellow, A. Courville,“Deep Learning”, (2015).