UNIVERSIDAD REY JUAN CARLOS
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
GRADO EN MATEMÁTICAS

TRABAJO FINAL DE GRADO

REDES NEURONALES PARA LA DETECCIÓN DE OBJETOS

AUTOR: AMANDA DEL ÁLAMO CABALLERO

España - Madrid - Febrero - 2022

ÍNDICE GENERAL

Pág.

Capítulo 1

Introducción

INDICE
escribir el capítulo de introducción+estado del arte, y por otra parte entrenar alguna red para reconocimiento de objetos. Como es un TFG de carácter general, utilizaremos la base de datos COCO para entrenar los modelos. Yo había pensado centrarnos en las redes YOLO y sus variantes, pero si has visto alguna que te llame la atención la incluimos también. La estructura del trabajo sería:

1.1. Motivación

1.1.1. Problema a resolver

1.1.2. Por qué es importante

1.1.3. Qué es deep learning y por qué es importante

1.1.4. Utilidades del deep learning en este problema

1.2. Objetivos

1.3. Estructura de la memoria

Capítulo 2

Estado del arte

2.1. Introducción a las NN

Podemos clasificar las redes neuronales dentro del campo de la inteligencia artificial. Para comenzar, hace 25 años no se conocía Internet pero sin embargo las redes neuronales ya existían desde los 60. En 1994, los ordenadores poseían muy poca capacidad es por eso que era inviables utilizar las redes neuronales en esos tiempos. Las redes neuronales simulan el comportamiento del sistema nervioso del ser humanos con nodos conectados entre sí. En 1943, McCulloch y Pitts propusieron el primer modelo de red neuronal que consistía en un modelo binario donde las neuronas tenían un límite o umbral predeterminado. Este modelo sirvió para futuros modelos. Podemos empezar clasificando las redes neuronales en modelos inspirados en la biología o modelos artificiales aplicados. Éstas últimas tienen varias características fundamentales:

- Fácil organización y adaptación. Se utilizan algoritmos creados para mejorar las posibilidades de procesado de adaptación
- Procesamiento no lineal: Se consigue una mayor capacidad para aproximar funciones, clasificar diferentes patrones o rutinas de procesado y se insonoriza el ruido exterior.
- Procesamiento paralelo: Al existir varios nodos conectados es habitual procesar varias instancias a la vez sin que se altere la interconectividad entre ellos.

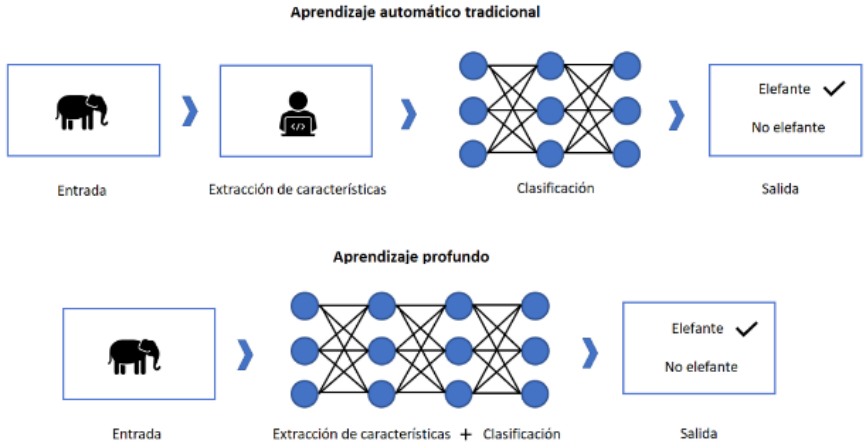
El componente principal de computación es el nodo o unidad. Estos nodos reciben ‘’input” de otros nodos o de un agente externo de datos. Las conexiones de neuronas artificiales son como sinapsis de las neuronas del cerebro. Una ilustración que muestra como es una neurona artificial: AQUI VAn dos IMAGEN

Antes de ahondar en las redes neuronales y sus componentes, se hará una breve introducción al aprendizaje automático o Machine Learning. Este aprendizaje podemos definirlo como un desarrollo de sistemas que son capaces de modificar su comportamiento de manera autónoma con el fin de aprender a partir de unos datos. Esto consiste en tratar de realizar multiples tareas de manera correcta y cada vez que podamos, mejorar la red minimizando el error. El aprendizaje automático podemos clasificarlo en dos grupos:

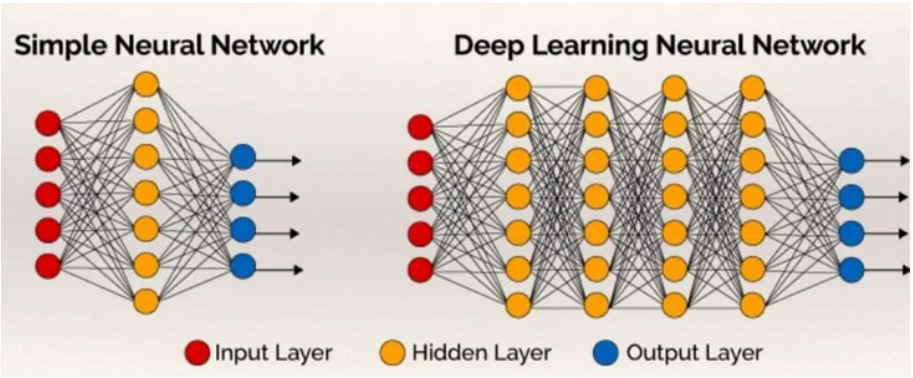
- Aprendizaje no supervisado(Unsupervised Learning): La finalidad de este aprendizaje es aprender características o patrones de los datos aportados que le permitan estructurarlos o clasificarlos.
- Aprendizaje supervisado(Supervised Learning): El objetivo de este aprendizaje es etiquetar o clasificar los datos nuevos que pudiese recibir encontrando el algoritmo adecuado para el correcto etiquetado de los datos de entrada. Uno de los algoritmos

más conocidos de este aprendizaje es el perceptrón multicapa que será explicado más adelante.

Las redes comienzan aprendiendo de manera sencilla en las primeras capas transfiriendo la información desde cada una a la siguiente, la cual se va combinando hacia algo más complejo hasta llegar a la capa final capaz de mostrar un objetivo.



2.1.1. Red neuronal simple y red neuronal profunda

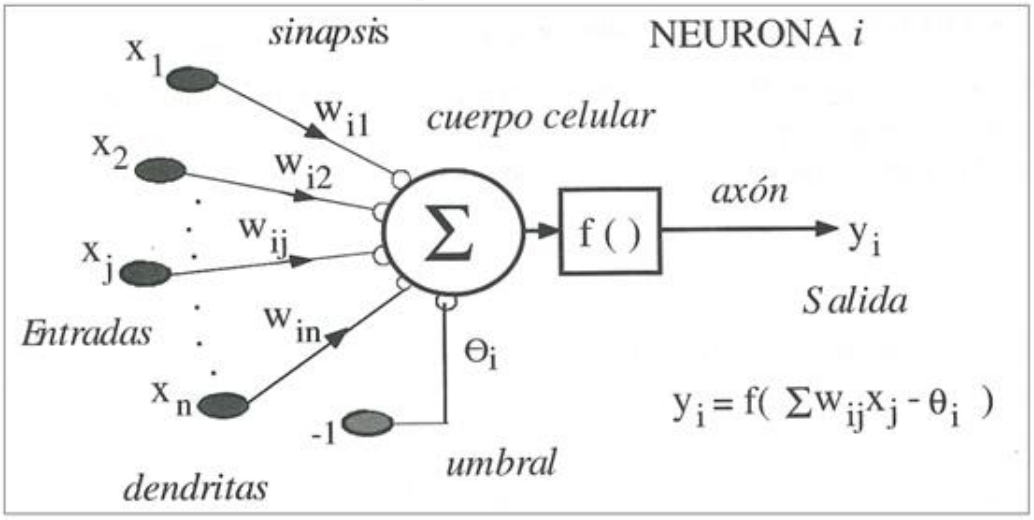


Existen muchas interconexiones y ramificaciones entre nodos, donde se observa que las ramificaciones de salida de algunas neuronas son las ramificaciones de entrada de otras. Así, las neuronas de color rojo no tienen ramificaciones de entrada porque la información la reciben desde el exterior o con algún estimulo inicial. Por otro lado están las neuronas azules, tienen ramificaciones de salida que no se conectan a otras neuronas si no que es la información final de salida o estímulo final. Y por último, tenemos las neuronas amarillas, que son las capas ocultas, a partir de las cuales determinamos si se trata de una red neuronal simple o profunda.

Cada conjunto de datos que se pasa a los nodos tienen un peso determinado que va cambiando o se va modificando y se conoce como proceso de aprendizaje. Cada nodo hace aplicar una función f de la suma de todos los datos entrantes ponderados mediando los pesos.

Por otro lado, aparece el término núcleo así como en las neuronas biológicas, ahí es donde se procesan todos los estímulos de entrada y los diferentes pesos. Una forma de procesar toda esa información consiste en multiplicar cada señar de entrada por su peso y sumar todo, es decir, realizar una combinación de pesos ente las conexiones y las entradas.

$y_1 = \sum_j w_{ij} * y_j$
Después de haber procesado esa información hay que aplicar una función de activación al resultado de la combinación lineal y el resultado final tiene que ser propagado hacia la salida.

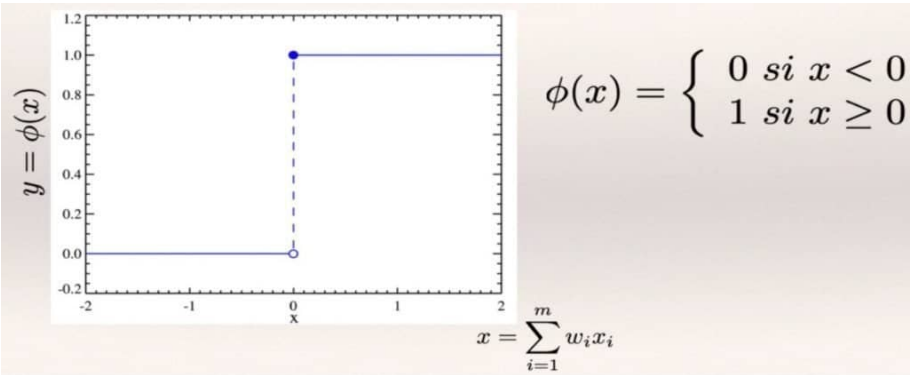


2.1.2. Funciones de activación

Estas funciones no son más que la forma de transmitir la información, previamente habiendo pasado por la combinación lineal de los pesos y las entradas, por las conexiones de salida. Si en algún caso se necesitase la información sin modificar o sin alteraciones, se aplica la función identidad y si en vez de sin modificar simplemente queremos que no se emita determinada información, entonces se emitiría un cero en las salidas de la neurona. En general, las funciones de activación tienen la utilidad de no aplicar linealidad al modelo y así conseguir que la red sea capaz de resolver problemas más complejos. De esta manera, si todas las funciones de activación fueran lineales, la red que se originaría sería idénticamente similar a una red sin capas ocultas o intermedias. Exsiten determinadas familias de funciones que son las más utilizadas en redes neuronales.

Función escalón (threshold)

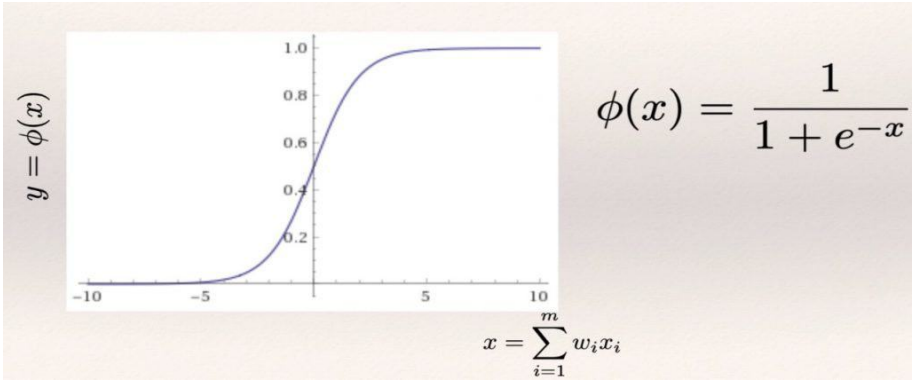
En esta función, el eje x representa el valor de la información de entrada ya ponderado y sus respectivos pesos y en el eje y se muestra el resultado del valor de la función escalón. Esta función realiza la prpagación de un 0 si el valor de la variable x es negativo o un 1 si el valor es positivo. No hay grises, es una clasificación estricta, de hecho es la función más rígida y exigente.



Función sigmoide

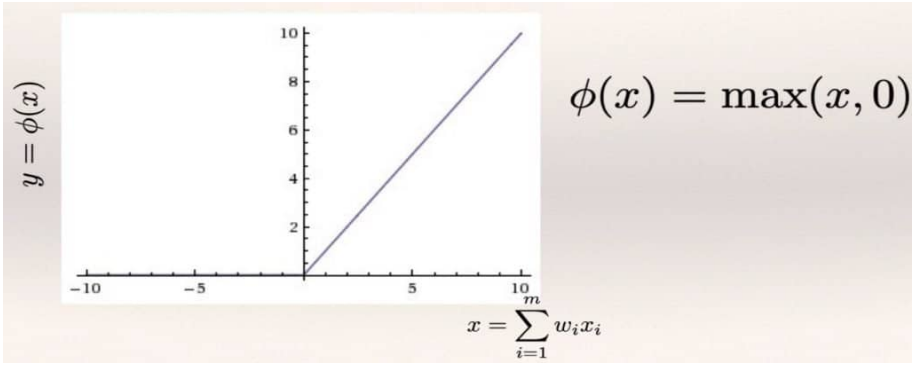
Existe una diferenciación entre valores negativos y positivos como en la función escalón

pero sin embargo, la función sigmoide no es tan estricta, las modificaciones se realizan de manera suave. La utilidad más común es la regresión logística, que es una de las técnicas más usadas también en Machine Learning. Además, la función es derivable y cuanto más positivo sea el valor de las x, más se acerca la función al valor 1, y viceversa, cuanto más negativo sea el valor de x, más nos acercamos al 0. La función sigmoide es de gran utilidad también en la última capa de la red, es decir, en la capa de salida porque clasifica diferentes valores y además trata de predecir con exactitud las probabilidades de que determinado valor pertenezca a cierta categoría, sabiendo que la probabilidad de que un acontecimiento sea seguro es de 1 y la de un hecho imposible es de 0.



Función rectificadora (ReLU)

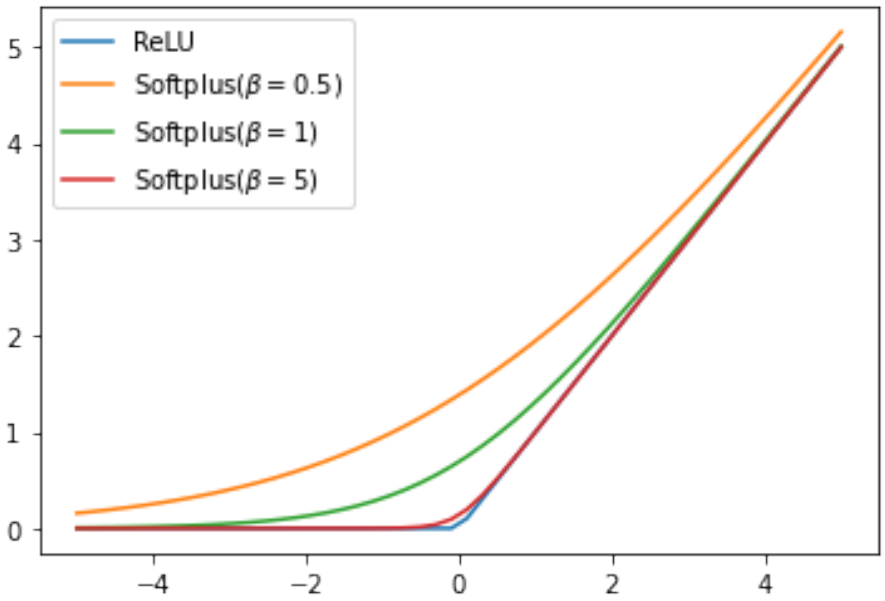
Esta función estará representada por una recta y para todo valor no positivo de x se tendrá un valor de 0. Si el valor de x es positivo, la función devuelve el mismo valor de x. Por tanto, esta función en realidad ignora todas las entradas que una vez ponderadas son negativas y se queda con el valor exacto de las positivas. Además, los valores no se restringen entre 0 y 1. ReLu(x) = max(0, x)



Funcion ReLu vs SoftPlus

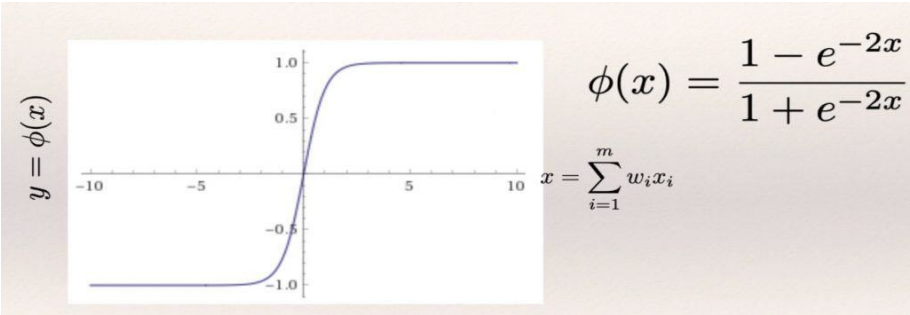
Softplus(x) = log(1 + exp(x))

Son similares pero la función SoftPlus posee continuidad en la segunda derivada en todo el dominio y ello es de gran utilidad en la optimización de los pesos de la red. Sin embargo, la función ReLu tiene menor coste computacional porque no necesita calcular exponenciales ni logaritmos.



Función tangente hiperbólica

Tiene cierta similitud con la función sigmoid pero en vez de ir de 0 a 1, se mueve de -1 a 1.



2.1.3. Configuración de la red neuronal

Una vez explicados los componentes principales de la red neuronal, lo resumimos en: una capa de neuronas de entrada para pasar información y datos a la red a modo de estímulos externos, después nos topamos con un conjunto de capas ocultas que son las encargadas de asumir, procesar, generar y entender toda la información proveniente de las capas de entrada para llevarlo a las capas de salida en forma de resultados.

En función del problema que queramos resolver tenemos que usar la función que mejor se ajuste, esto depende de varios factores. El factor que hay que tener en cuenta seguro es el rango de valores que se quiere para los outputs de la neurona. Dependiendo si hay o no restricciones en ese rango, se elegirá una u otra función. Si no las hay, se puede usar la función identidad. Si la restricción por ejemplo es que la salida deba ser positvo, sin límites en cuántos, se podría elegir la función rectificadora. También puede ser de utilidad la función escalón para obtener un output binario. Por otro lado, si lo que queremos es utilizar uno o varios algoritmos que utilicen derivadas lo más apropiado es usar funciones cutas derivadas estén plasmadas en todo el dominio, pese a que esto originará un aumento del coste computacional, como por ejemplo la función sigmoidal o la tangente hiperbólica. Otro aspecto a tener en cuenta en la creación de nuestra red neuronal será prestar atención a la hora de diseñar la arquitectura cuantas más capas y neuronas metamos a la red, la

complejidad y el coste computacional aumentarán. No hay ninguna estipulación acerca de cual será la mejor arquitectura, es más bien un proceso de prueba y error hasta lograr el ajuste que se adecue más y mejor a la salida que buscamos.

2.1.4. Función de costes

Se puede explicar con un ejemplo bastante visual para entender lo que estamos haciendo. La finalidad de esto es ajustar los pesos para que la red neuronal aprenda y así los valores de salida se asimilarán más a los valores reales. Supongamos que nuestra finalidad es construir una red neuronal que prediga el precio de los coches en la India, obtenemos varias variables que van a ser fundamentales en el estudio como: número de asientos, número de dueños que ha tenido, km recorridos, marca, año de fabricación... Todos estos registros los tenemos organizados en una base de datos ya limpiada, filtrada y analizada lo suficiente como para entender el problema. Para cada una de estas variables, se introducen a la red los valores correspondientes a cada casa para los cuales iremos obteniendo resultados que no serán como los valores reales, e incluso a veces puede al principio no estar bien ajustada y salir poco similares.

La función de coste es una función que calcula la variación entre el resultado del output de nuestra red y el valor original. La función más sencilla y usada es la del error cuadrático medio, pero se pueden utilizar otras de mayor complejidad. El error cuadrático medio se puede calcular consiguiendo el cuadrado de la diferencia entre el output de la red y el valor real de cada coche, asumiendo que los valores negativos no son válidos. Después, sumamos el resultado para todas las casas y lo dividimos entre el número de muestras que estemos estudiando:

$$MSE = \frac{1}{M}\sum_{i=1}^{M}(real_i - estimado_i)^2 \tag{1}$$

Se seleccionan las iteraciones que realizará la red con los datos de entrenamiento que tenemos e intentamos en la mayor medida posible minimizar esa diferencia, esa función de coste. Esto se consigue ajustando los pesos que proporcionamos a cada conexión entre neuronas al terminar cada iteración. Una manera de calcular la función de costes es como la que vemos en la figura de abajo. Calculamos el cuadrado de la diferencia entre el valor de salida y el real (evitando valores negativos) de cada casa. Luego sumamos el valor para todas las casas y dividirlo entre el número de muestras que poseamos.

2.2. MLP

2.3. CNN

2.4. Redes de segmentación

2.4.1. Segmentación semántica vs segmentación instancias

2.5. NN para segmentación semántica

2.5.1. Yolo y sus variantes

2.5.2. Single Shot MultiBox Detector (SSD)

Capítulo 3

Aplicaciones

3.1. Detección de objetos en condiciones de baja luz

3.2. Detección de peatones

3.3. Detección de vehículos

3.4. Seguimiento de un balón

Capítulo 4

Metodología

4.1. Redes entrenadas

Capítulo 5

Experimentos

5.1. Mi propia red

5.2. Comparación con otras redes

Capítulo 6

Conclusiones