Escuela Técnica Superior de
Ingeniería Informática

TRABAJO FIN DE GRADO

Título del trabajo

Realizado por
Jaime Rodríguez Albuín

Para la obtención del título de
Grado en Ingeniería Informática - Tecnologías Informáticas

Dirigido por
Isabel De Los Ángeles Nepomuceno Chamorro

Realizado en el departamento de
Lenguajes y Sistemas Informáticos

Convocatoria de Junio, curso 2020/21

Agradecimientos

Quiero agradecer sobre todo a mi hermano y a mi madre por estar apoyándome desde el minuto cero incondicionalmente, siempre dispuestos a llevarme y a traerme de Sevilla y siendo sin duda piezas clave en esta etapa de mi vida.

También quiero agradecer a mis tíos Victoria y Manolo por ayudarnos en momentos tan complicados y haber ejercido siempre de familia.

Agradezco a mi tutora Isabel el hecho de aceptarme como alumno incluso llegando un poco tarde. También a todos los profesores que he tenido a lo largo de la carrera, puesto que sé con certeza que la formación que he recibido a lo largo de estos cuatro años me servirá, tanto en los conocimientos como en la forma de afrontar los problemas.

No me voy a olvidar de los maravillosos compañeros que he conocido a lo largo de este viaje, y sobre todo agradezco las amistades con Arturo, Ventura, Emilio, Jesús, Juan, Juan Carlos, Pedro García, Pedro Escobar, Curro, Antonio, Manuel, Javi y Josemi.

Resumen

Trabajo de investigación sobre redes neuronales convolucionales y la detección de paciences con neumonía, posiblemente provocadas por Covid.

Palabras clave: Redes neuronales, Convolución, Pandas, Keras, Tensorflow, Covid, Kaggle, Google Colab, Inteligencia Artificial, Deep learning, Neumonía, Salud.

Abstract

Research project about convolutional neural networks and pneumonia detection on patients, possibly provoked by Covid

Keywords: Neural networks, Convolution, Pandas, Keras, Tensorflow, Covid, Kaggle, Google Colab, Artificial Intelligence, Deep learning, Pneumonia, Health.

Índice general

Índice de figuras

Índice de extractos de código

1. Introducción

Motivación

La principal fuente de motivación ha sido la pandemia. Mi principal objetivo fué encontrar una manera de predecir un diagnóstico sobre las personas que pudieran padecer de COVID. Entre síntomas hay uno que siempre ha sido uno de los principales indicios de dicha enfermedad, y es la neumonía provocada por el virus. Esta neumonía es la que provoca la tos en los pacientes, y el hecho de encontrar esta propuesta en la página web de los TFG de la ETSII fué un flechazo.

Además, el hecho de ver las redes neuronales tan por encima en IA y AIA me dejó con las ganas de investigar. No conocía las redes convolucionales, y me han parecido un descubrimiento espectacular.

También el hecho de desarrollar algo que puede ayudar en un campo tan noble como la salud es una motivación enorme, aunque tengo muy claro que hay gente con mucha más experiencia que yo en el sector y no sé si podría a llegar a competir con ellos.

2. Planificación temporal y costes

3. Estudio Previo

3.1. Introducción

En este capítulo introduciremos las redes convolucionales desde el perceptrón hasta el estado del arte y el dataset que trabajaremos.

3.2. Aprendizaje supervisado y no supervisado

Primero deberemos preguntarnos, ¿qué significa el "aprendizaje" en máquinas? La respuesta depende de qué tipo de aprenzidaje buscamos:

3.2.1. Aprendizaje supervisado

Este acercamiento al aprendizaje automático busca obtener una función a partir de un conjunto de datos, es decir, relaciones que asocian entradas con salidas. Este acercamiento al aprendizaje automático incluso llega a proporcionar las formas de clasificar más comunes. Dependiendo de la salida diferenciaremos entre modelos de clasificación, si la salida es un valor categórico (como una enumeración, o un conjunto finito de clases), y modelos de regresión, si la salida es un valor en un espacio continuo.

Las aplicaciones son muy numerosas y siguen estando a la orden del día, tales como reconocimiento de voz, clasificación de imágenes (como en este trabajo) o traducción de idiomas. También existen otras aplicaciones más exóticas que merecen la pena mencionar:

- Detección de objetos: dada una imagen dibujar una caja que delimite los objetos a detectar. Este problema puede ser presentado como un problema de clasificación (obteniendo muchos candidatos a los que dibujar la caja, elegir si se dibujan o no) o como una unión entre un problema de clasificación y uno de regresión, donde las esquinas de dicha caja se predigan mediante vectores de regresión.

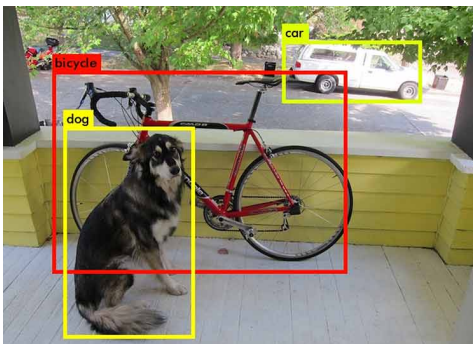

Figura 3.1: Resultado de aplicar la detección de objetos

- Segmentación de imágenes: dada una imagen, dibujar una máscara píxel a píxel sobre un objeto en específico.

Figura 3.2: Ejemplo de segmentación de imágenes

- Generación de secuencias: dada una imagen, predecir un subtítulo que la describa. Este problema a veces puede ser reformulado como una secuencia de

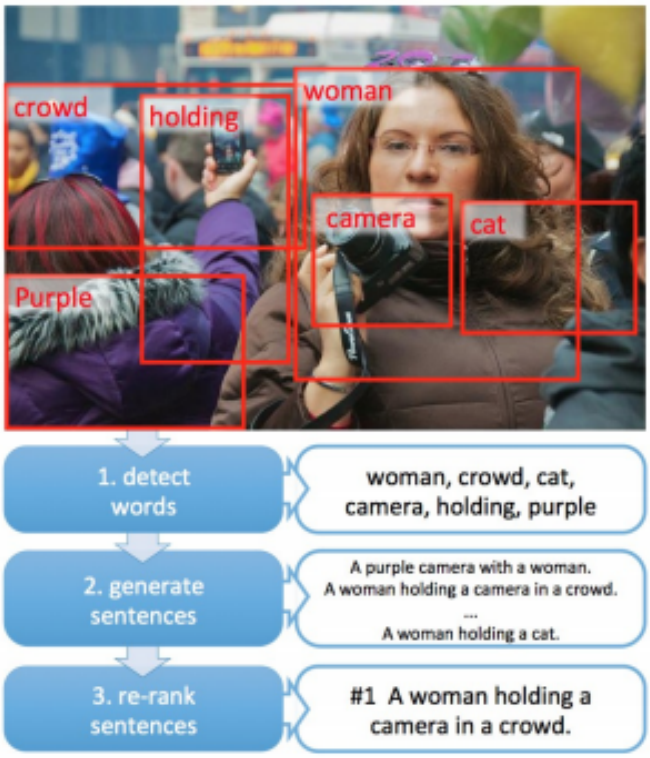

Figura 3.3: Imagen que demuestra el flujo de la generación de subtítulos

3.2.2. Aprendizaje no supervisado

Esta rama del aprendizaje automático se utiliza cuando no nos interesa ajustar pares (entrada/salida), sino en aumentar el conocimiento estructural de los datos de entrada. Consiste en encontrar transformaciones interesantes de los datos de entrada sin la ayuda de etiquetas, con el propósito de visualizar los datos, comprimirlos, eliminar ruido o simplemente aumentar el entendimiento de los datos en cuestión.

Su principal aplicación se encuentra en el análisis de datos, a suele ser un paso necesario en la mejora del entendimiento de los datos de un dataset antes de la resolución por un algoritmo de aprendizaje supervisado.

Tipos de algoritmos no supervisados más comunes:

- Clustering: busca agrupar una serie de elementos según su semejanza. Por ello, un *cluster* es una agrupación de elementos similares.

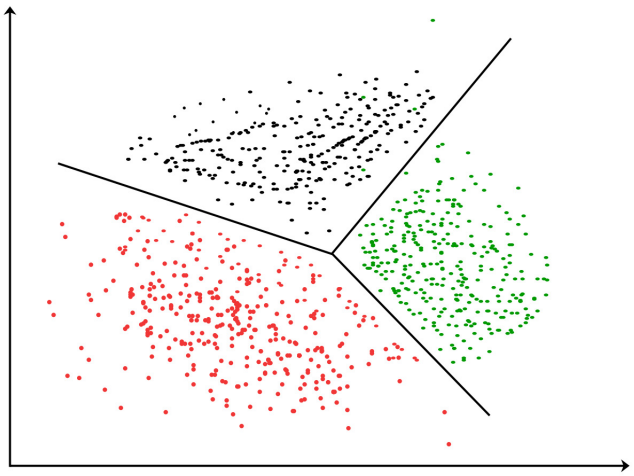

Figura 3.4: Resultado de la aplicación de clustering sobre un conjunto de individuos

- Reducción de dimensiones: es la transformación de datos de un espacio dimensional elevado a otro de menores dimensiones, manteniendo en este nivel menor de dimensionalidad características importantes.

Este tipo de algoritmos es muy común en campos que manejan grandes cantidades de observaciones y/o variables, como el procesamiento de señales, el reconocimiento de voz o la bioinformática.

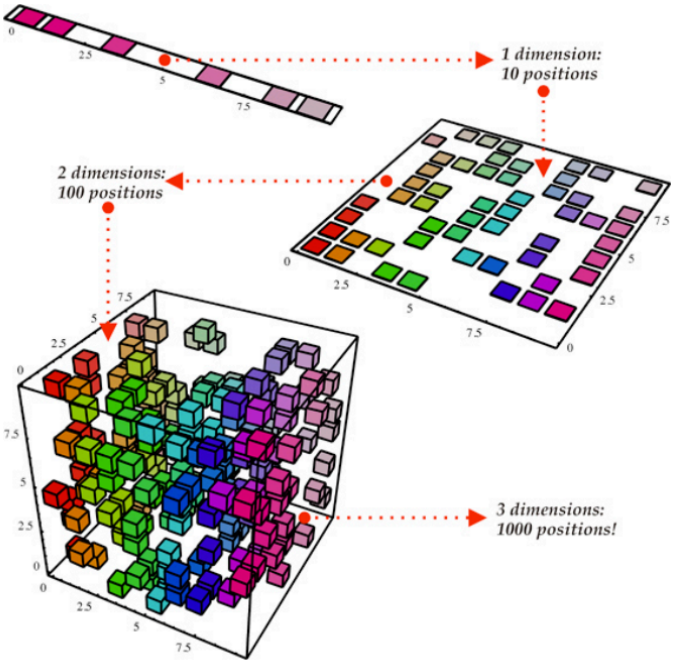

Figura 3.5: Etapas en la reducción de dimensiones.

3.3. Redes neuronales

En esta sección profundizaremos en las redes neuronales, desde la inspiración biológica hasta las redes convolucionales más famosas y desarrolladas.

3.3.1. Inspiración biológica

Una red neuronal biológica está compuesta por aproximadamente 86 billones de neuronas, conectadas entre ellas. Los científicos estiman que existen más de 500 millones de billones de conexiones en el cerebro humano. Incluso las redes neuronales artificiales más largas ni se acercan a este número.

Desde un punto de vista informatizado, podemos decir que una neurona es una unidad excitable que puede procesar y transmitir información mediante señales eléctricas y químicas. Esta unidad (la neurona) es el principal componente en nuestro sistema nervioso.

Estructura

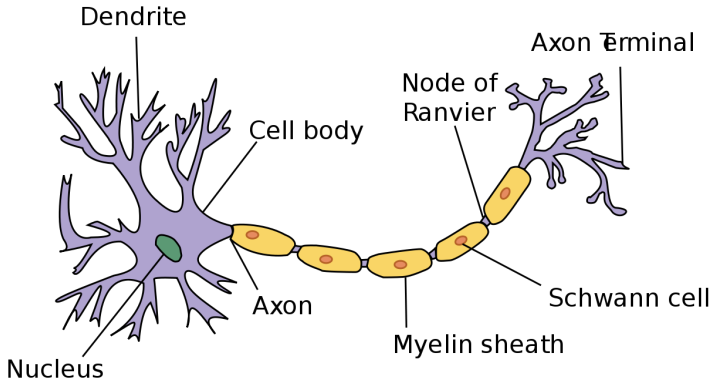

Figura 3.6: Estructura de una neurona biológica.

Las partes más importantes que componen una neurona son:

- El soma (o cuerpo celular): procesa las activaciones entrantes y las convierte en activaciones de salida.

- El núcleo: contiene material genético en forma de ADN.

- Las dentritas: son fibras que nacen desde el soma y provee de zonas receptoras para recibir las activaciones de otras neuronas.

- Axones: son fibras que actúan como líneas transmisoras que envían la activación a otras neuronas.

Funcionamiento

Los pasos clave en el funcionamiento de una neurona son:

1. Las señales de otras neuronas se almacenan en las dentritas.
2. El soma agrupa las señales entrantes (espacial y temporalmente).
3. Cuando se recibe la suficiente cantidad de señal la neurona se "dispara"(produce una diferencia de potencial).
4. Esta diferencia de potencial es transmitido a lo largo del axón hacia otras neuronas o hacia estructuras fuera del sistema nervioso (Por ejemplo músculos).
5. En caso de no recibir suficiente señal la señal almacenada en las dentritas caerá rápidamente y la neurona no se disparará.

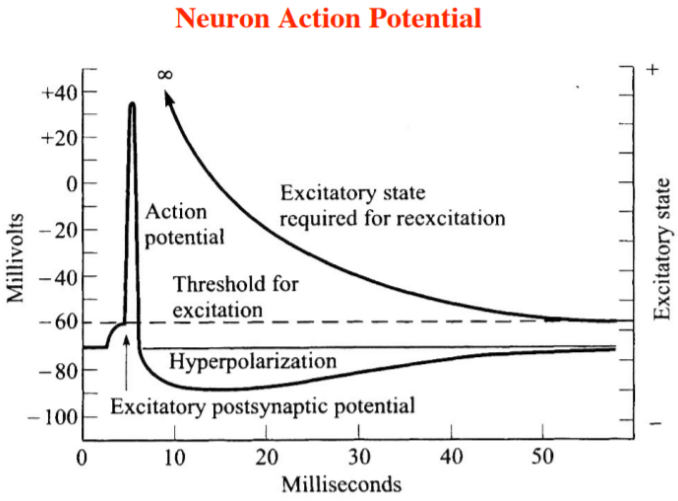

Figura 3.7: Diagrama que muestra el disparo de una neurona

3.3.2. El perceptrón

El perceptrón es un modelo lineal usado para la clasificación binaria. En el campo de las redes neuronales el perceptrón es considerado la neurona artificial.

El primer modelo matemático de dicha neurona artificial fué presentado por el psiquiatra y neuroanatomista Warren McCulloch y el matemático Walter Pitts en 1943. Este modelo se considera el precursor del perceptrón, y es llamado "Threshold Logic Unit.º "TLU". Dicho modelo era capaz de simular puertas lógicas, como la AND, OR o la XOR.

Estructura

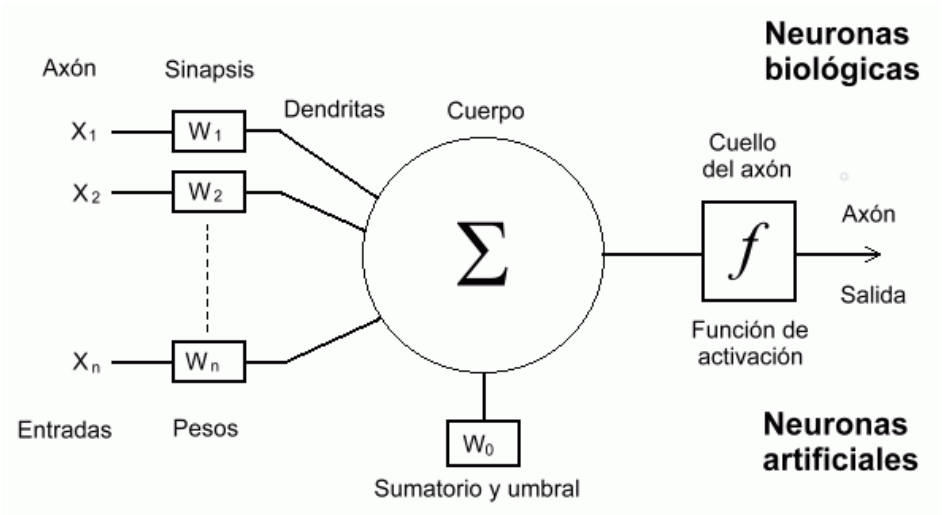

Figura 3.8: Comparativa de la estructura de la neurona biológica con el perceptrón

En la figura anterior podemos apreciar las distintas partes de la estructura de una neurona artificial:

- El conjunto de elementos de entrada (x_1,x_2,...,x_n).
- Los pesos asociados a cada entrada (w_1,w_2,...,w_n), los cuales se multiplicarán por cada elemento de entrada.
- Una función de agregación ($\sum$) que sumará las multiplicaciones de los pesos con sus respectivos elementos de entrada. A esta agregación se le suma un factor de parcialidad (bias). Este factor de parcialidad es utilizado para acelerar o frenar la activación de un nodo.
- Una función de activación (f).

Como hemos dicho, el perceptrón es un algoritmo para **aprender** un clasificador binario. Los elementos externos llegarán al perceptrón para devolver un valor. Por ello, teniendo nuestro perceptrón **f(x)**, siendo **x** dichos elementos externos, siendo **b** el factor de parcialidad (bias), siendo **w** los pesos asociados a cada elemento externo y siendo **w*x** el producto escalar de x y w:

$$f(x) = \begin{cases} 1 & si \quad w * x + b > 0 \\ 0 & E.O.C \end{cases}$$

Este clasificador funciona solo en caso de que el conjunto de datos de entrada sea linealmente separable. Este hecho produce que durante el entrenamiento de dicho perceptrón la clasificación erre, produciendo un clasificador de muy baja confianza.

Para poder entender cómo funciona dicho algoritmo primero tendremos que observar las distintas funciones de activación que podemos utilizar.

Funciones de activación

Las funciones de activación son funciones que dependiendo de lo que reciban se comportarán de una manera u otra. Un ejemplo es la función escalón de Heaviside, que devuelve 0 si el número que recibe es negativo o 1, si el número es mayor o igual que 0:

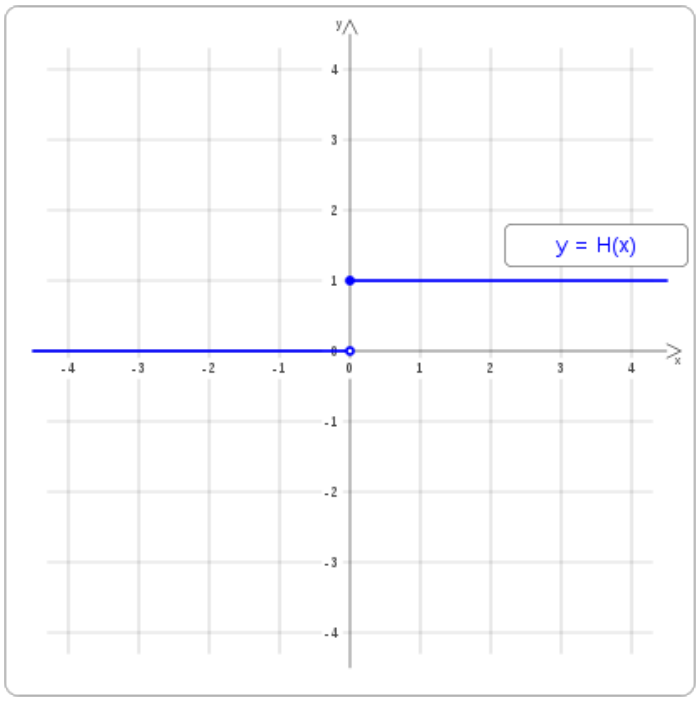

Figura 3.9: Gráfica de la función escalón de Heaviside.

Empezaremos por comentar las funciones de activación más comunes en el campo de las redes neuronales:

- ReLU (Rectifier Linear Unit): es la función que define la parte positiva de sus argumentos.

$$f(x) = x^+ = max(0, x)$$

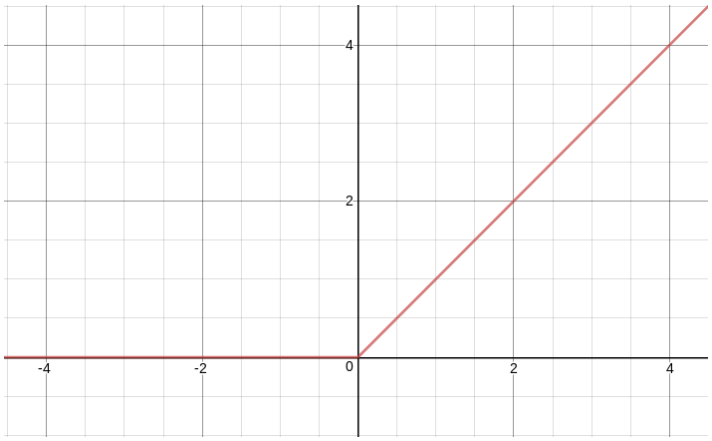

Figura 3.10: Gráfica de la función ReLU.

- Sigmoide (σ): es una de las funciones de activación más utilizadas. En este campo sirven para introducir la "no linealidad.[en] un modelo. Ya que el producto escalar entre los pesos y x es una combinación lineal, al añadirle esta "no linealidad.[el] resultado de dicha combinación se suaviza.

$$\sigma(x) = \frac{1}{1+e^{-x}}$$

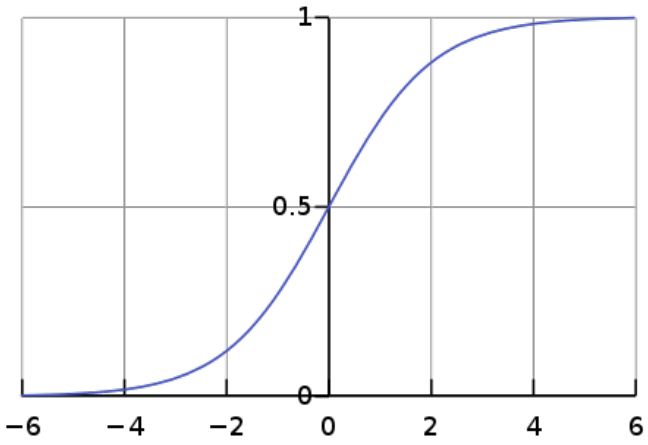

Figura 3.11: Gráfica de la función sigmoide.

- Softmax (función exponencial normalizada): es utilizada en la última capa de los clasificadores basados en redes neuronales, debido a que devuelve un vector de K valores reales a otro vector con K valores reales que suman 1. Si uno de los valores es negativo o pequeño, la función le dará poca probabilidad, mientras que si algún valor es más grande que el resto le dará más probabilidades.

  Esta función solo se puede utilizar cuando las clases del clasificador son mutuamente exclusivas.

$$f(x) = \ln 1 + e^x$$

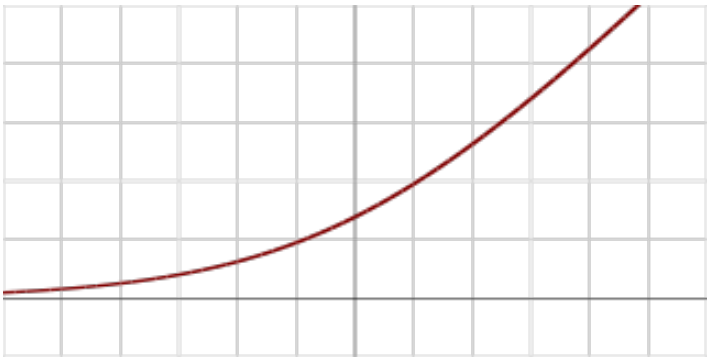

Figura 3.12: Gráfica de la función softmax.

Entrenamiento:

Poseyendo un conjunto de datos con N muestras, un vector de pesos (w_i), los valores asociados a cada característica de la muestra (x_i),

El entrenamiento del perceptrón sigue el siguiente procedimiento:

Algoritmo de entrenamiento del perceptrón 1:

Input: Datos de entrenamiento **D**
Output: Pesos **w** ya entrenados
Inicializar $\mathbf{w} \leftarrow 0, \mathbf{b} \leftarrow 0$
while *no estén correctamente clasificados todos los* x **do**
  for $(\boldsymbol{x}, y) \in \boldsymbol{D}$ **do**
    if $y(\boldsymbol{w}^*\boldsymbol{x} + b) \leq 0$ **then**
      $\mathbf{w} \leftarrow \mathbf{w} + y * \mathbf{x}$
      $\mathbf{b} \leftarrow \mathbf{b} + y$
    end
  end
end

Componentes:

- b: es el factor de parcialidad (bias).
- y: la etiqueta esperada

3.3.3. Red Neuronal Tradicional

Una serie de perceptrones pueden agruparse entre sí para formar redes neuronales. Para ello cada neurona se conectará con todas las neuronas de las capas posteriores y anteriores.

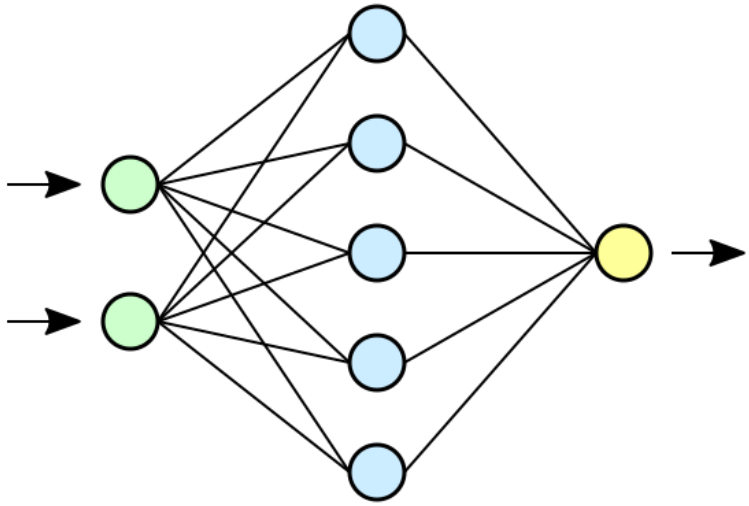

Figura 3.13: Perceptrón multicapa.

Estructura:

Las redes neuronales tradicionales poseen esta estructura:

- Capa de entrada: la primera capa de nuestra red neuronal. Por ella alimentaremos nuestra red con datos.
- Capas ocultas (o totalmente conectadas): estas serán las capas intermedias de nuestra red neuronal. Las neuronas de estas capas estarán totalmente conectadas tanto a la capa anterior como a la posterior. Son la clave para que una red neuronal pueda modelar funciones no lineales.
- Capa de salida: esta capa será la encargada de predecir la clase del elemento de entrada.
- Conexiones: Los pesos asociados a estas conexiones serán los que se ajustarán a la hora de realizar las predicciones a lo largo del entrenamiento. El algoritmo que usarán para entrenarse será el algoritmo de propagación hacia atrás.

La salida de cada neurona (n_i) vendrá dada por la suma ponderada de cada peso de cada conexión de entrada (w_{ij}) por su elemento de entrada correspondiente(x_i) mas el valor bias propio de cada neurona (b_i), siendo f la función de activación de cada nodo en la capa oculta y h la función de activación de la capa de salida, quedarán las ecuaciones de la figura 3.13 en las capa oculta y de salida de esta manera:

$$
\begin{aligned}
\mathrm{a}^{(1)}_1 &= f(w_{11}x^{(0)}_1 + w_{21}x^{(0)}_2 + b_1),\\
\mathrm{a}^{(1)}_2 &= f(w_{12}x^{(0)}_1 + w_{22}x^{(0)}_2 + b_2),\\
\mathrm{a}^{(1)}_3 &= f(w_{13}x^{(0)}_1 + w_{23}x^{(0)}_2 + b_3),\\
\mathrm{a}^{(1)}_4 &= f(w_{14}x^{(0)}_1 + w_{24}x^{(0)}_2 + b_4),\\
\mathrm{a}^{(1)}_5 &= f(w_{15}x^{(0)}_1 + w_{25}x^{(0)}_2 + b_5),\\
\mathrm{o} &= \mathrm{h}(\mathrm{w}_{25}x^{(1)}_1 + w_{25}x^{(1)}_2 + w_{25}x^{(1)}_3 + w_{25}x^{(1)}_4 + w_{25}x^{(1)}_5 + b_6)
\end{aligned}
\tag{3.1}
$$

Entrenamiento de la red neuronal.

Esta es la etapa más importante a la hora de conseguir un modelo preciso y rápido. Cuanto mejor es una red neuronal, mejor captará las señales y mejor detectará lo que es ruido.

Este proceso de aprendizaje consistirá en re-ajustar los pesos y los bias para cambiar la relevancia de ciertas características, aumentándola o disminuyéndola en base a los bits de cada peso.

En la mayoría de los dataset hay características fuertemente relacionadas con ciertas etiquetas (como puede ser en una imagen la relación entre la forma esférica de un balón). Las redes neuronales aprenden a base de prueba y error, aprendiendo ciegamente estas características ajustando los pesos para poder dar veredictos más acertados según avanza el proceso de aprendizaje.

Algoritmo de propagación hacia atrás:

Este algoritmo es el responsable de reducir el error durante la fase de entrenamiento de una red.

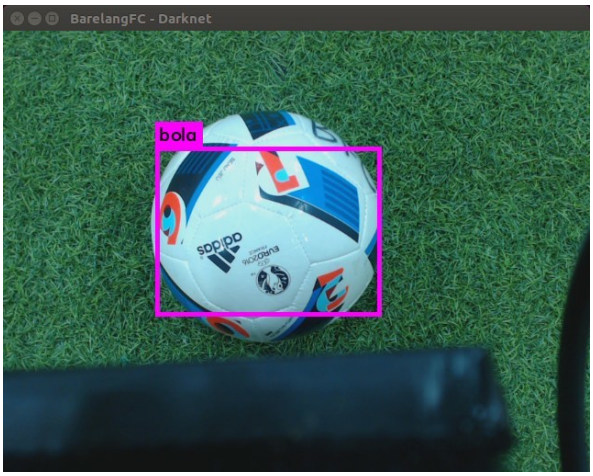

Figura 3.14: Detección de una pelota por una red neuronal.

3.4. Estado del arte

3.5. Conclusiones

4. Análisis del problema

4.1. Introducción

En este capítulo explicaremos...

4.2. Requisitos de información

Los requisitos de información son...

4.3. Requisitos funcionales

Los requisitos funcionales son...

4.4. Requisitos no funcionales

Los requisitos no funcionales son...

4.5. Conclusiones

En este capítulo concluimos que...

5. Diseño del problema

5.1. Introducción

En este capítulo explicaremos...

5.2. Conclusiones

En este capítulo concluimos que...

6. Implementación

6.1. Introducción

En este capítulo explicaremos...

6.2. Herramientas

6.3. Conclusiones

En este capítulo concluimos que...

7. Pruebas

7.1. Introducción

En este capítulo explicaremos...

7.2. Conclusiones

En este capítulo concluimos que...

8. Conclusiones

9. Bibliografía