Redes Neuronales Aplicadas a Problemas Científico-Técnicos

INTRODUCCIÓN A LA INTELIGENCIA ARTIFICIAL.

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDO

- Big data.
- Programa.
- Motivación.
- Resumen.

INTRODUCCIÓN

Definición de Inteligencia Artificial

- Coloquialmente, el término inteligencia artificial se aplica cuando una máquina imita las funciones cognitivas que los humanos asocian con otras mentes humanas, como por ejemplo:
  - percibir,
  - razonar,
  - aprender y
  - resolver problemas.

INTRODUCCIÓN

Definición de Inteligencia Artificial

- The first modern name for the discipline and entity of AI was given to John McCarthy in 1956, with the other cofounders of the discipline. For its founders, it was defined as based on ”the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it”.

J. McCarthy, M. L. Minsky, N. Rochester, C. E. Shannon, A proposal for the Dartmouth summer research project on artificial intelligence, Computer Based Learning Unit, Leeds University, 1955.
http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html

INTRODUCCIÓN

Definición de Inteligencia Artificial

- The ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings.

B.J. Copeland, Artificial intelligence, Encyclopedia Britannica, 2019.
https://www.britannica.com/technology/artificial-intelligence

INTRODUCCIÓN

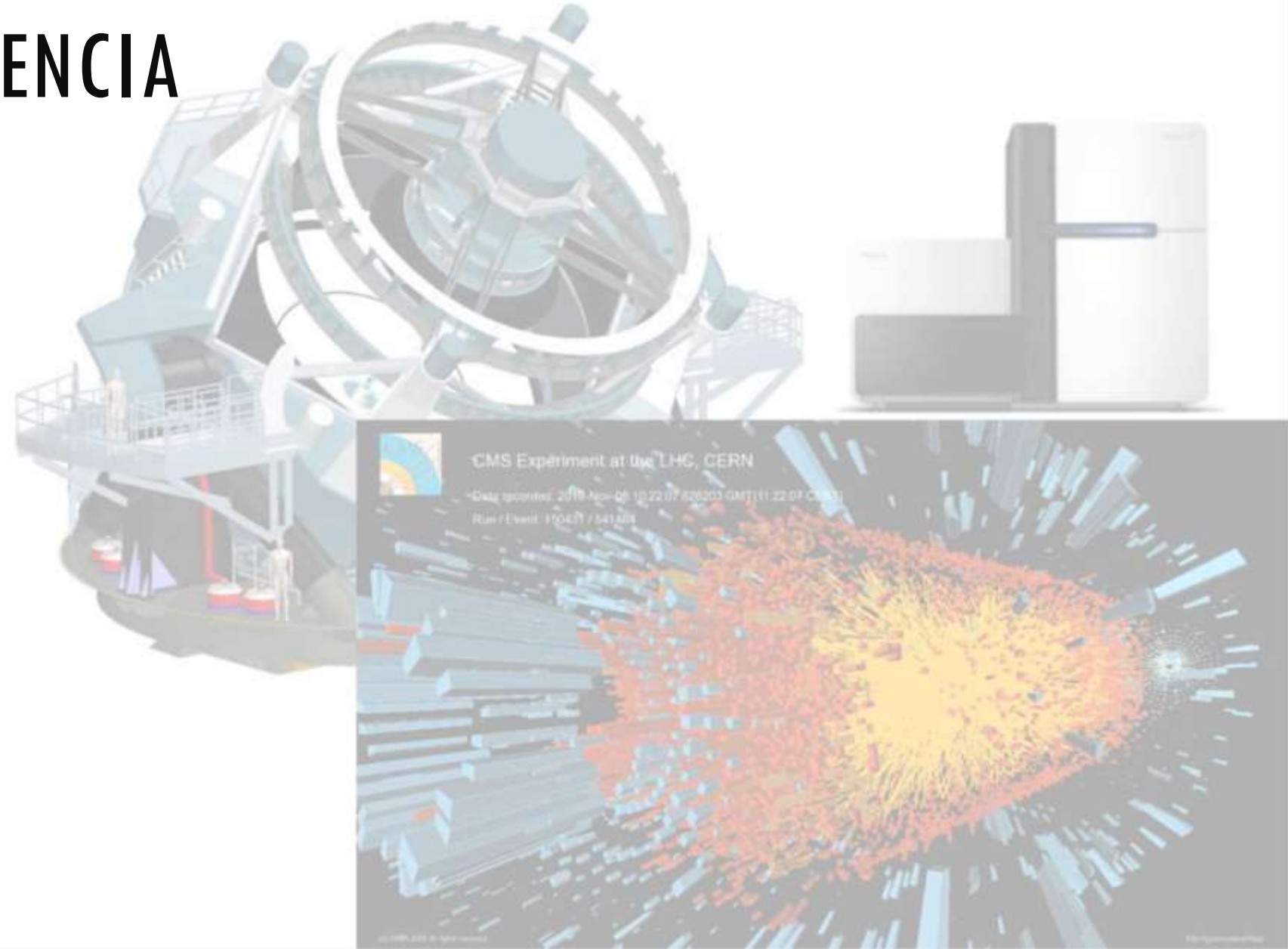
Ramas de Inteligencia Artificial

- Procesamiento de Lenguaje Natural
- Robótica
- Lógica Difusa
- Aprendizaje Automático
- Redes Neuronales

CRONOLOGÍA

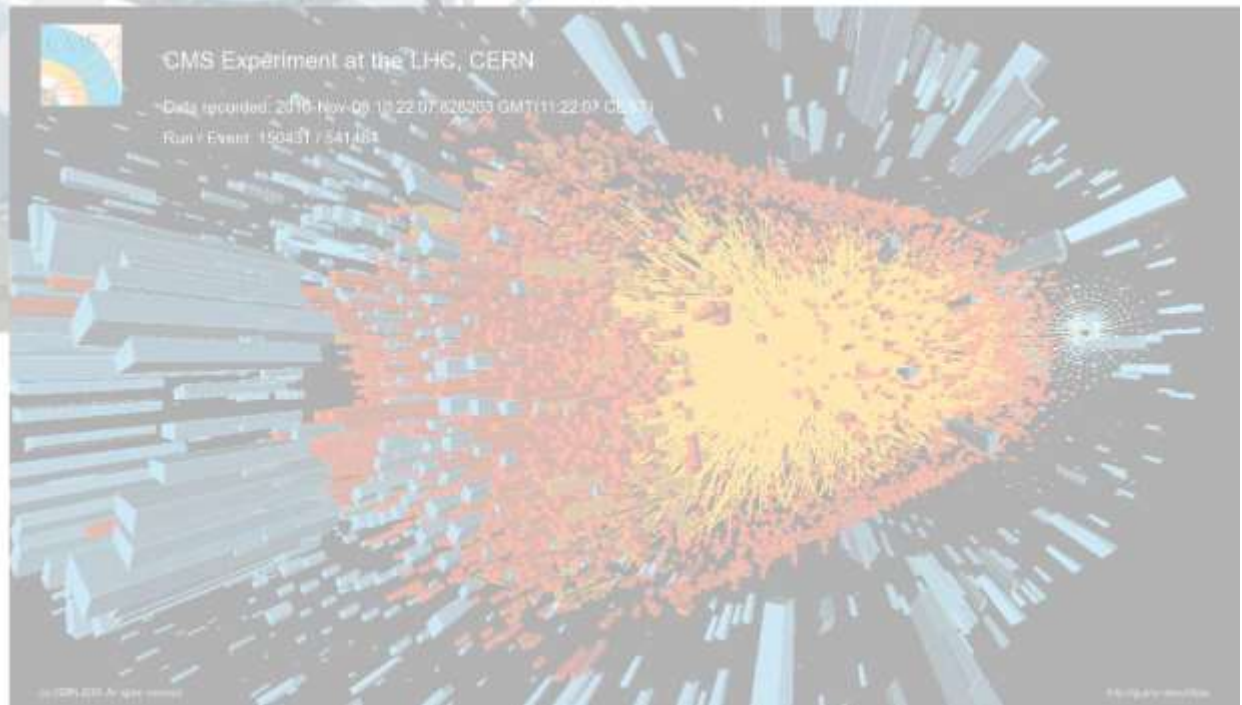
- En 1943 Warren McCulloch y Walter Pitts publican los trabajos *Un cálculo lógico de las ideas inmanentes en la actividad nerviosa* y posteriormente en 1947, *Cómo conocemos los universales: la percepción de las formas visuales y auditivas*, ambos en el Boletín de Biofísica Matemática. Se consideran los primeros trabajos relacionados con la IA
- En 1950 Alan Turing publica su artículo *Computing Machinery and Intelligence* en el que describe el conocido test para la detección de comportamientos inteligentes

BIG DATA EN CIENCIA



BIG DATA EN CIENCIA

- LHC, O(20 PB/año), 1 dispositivo.
- NGS (secuenciación masiva de genes), O(0.5 TB/genoma), O(1/hospital) dispositivos



DESAFÍOS BIG DATA

- Gestión y manejo de los datos.
- Tiempo de procesamiento creciente.
- Pérdida de precisión de los cálculos debido a la representación numérica.
- Extracción de información de los datos.
  - Aprendizaje supervisado (clasificación y regresión).
  - Aprendizaje no supervisado.
  - Identificación de datos anómalos.
  - Análisis de series temporales.

BIG DATA

Las ”V”s. ¿3 ó 5?

- **Volumen:** Tamaño del volumen de datos. Muchos datos en la Ciencia y Industria de hoy día.
- **Velocidad**: Velocidad de generación más elevada que en el pasado. Constante en un experimento.

  Pero también, latencia de procesamiento relativo a la creciente demanda para interactuar con ellos.
- **Variedad:** Diversidad de fuentes, de formato, de calidad, de estructura. Baja diversidad en Ciencia.

ORIGEN DE BIG DATA.

- Almacenamiento masivo y barato. Ya podemos almacenar todo, todo, todo. No hay que descartar o cribar datos.
- Sensores baratos y ubicuos. El mundo se ha llenado de sensores que proveen de grandes volúmenes de información.

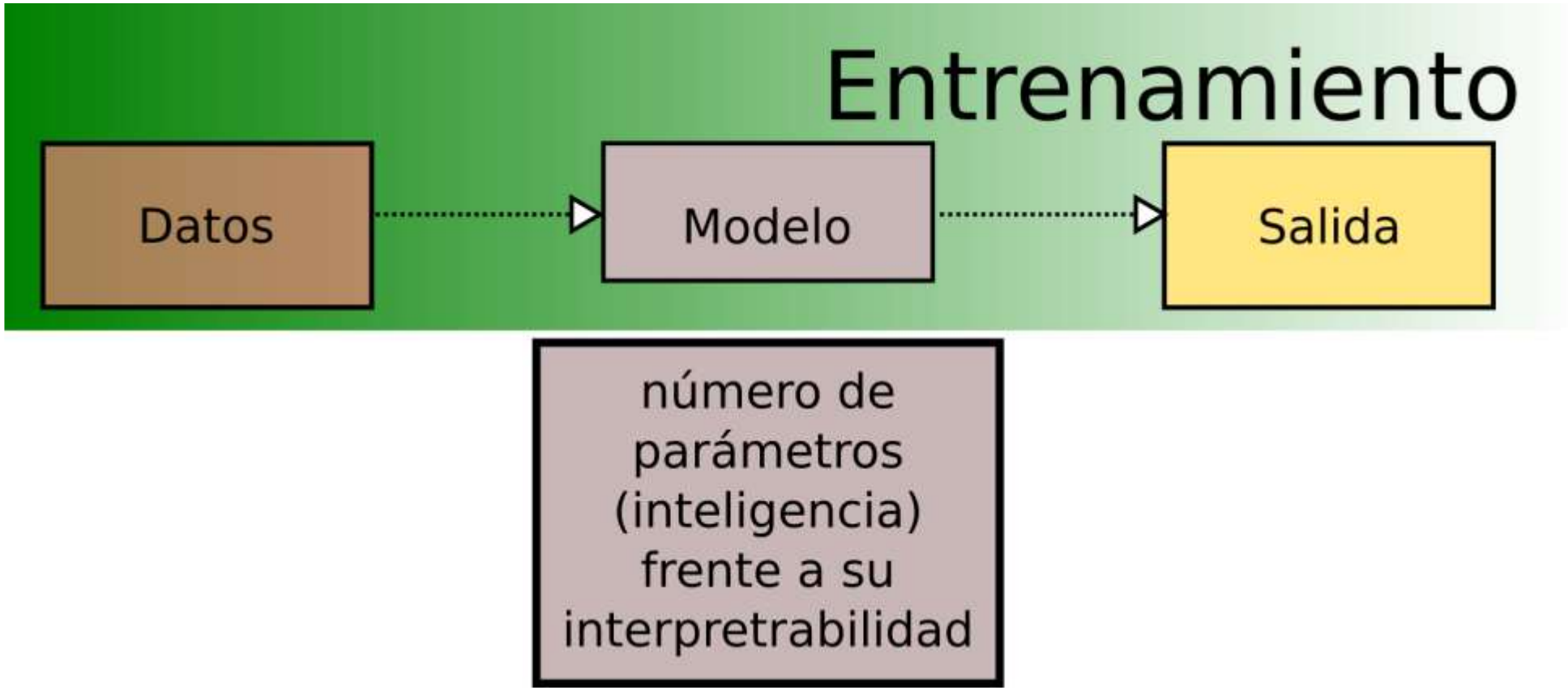
PROGRAMA.

- Usos del aprendizaje automático.
- Relación entre la regresión lineal y la regresión logística y las redes neuronales.
- Construir redes neuronales:
  - para problemas de regresión,
  - de clasificación y
  - para pronóstico en series temporales.
- Redes neuronales explicables: importancia de las variables independientes.

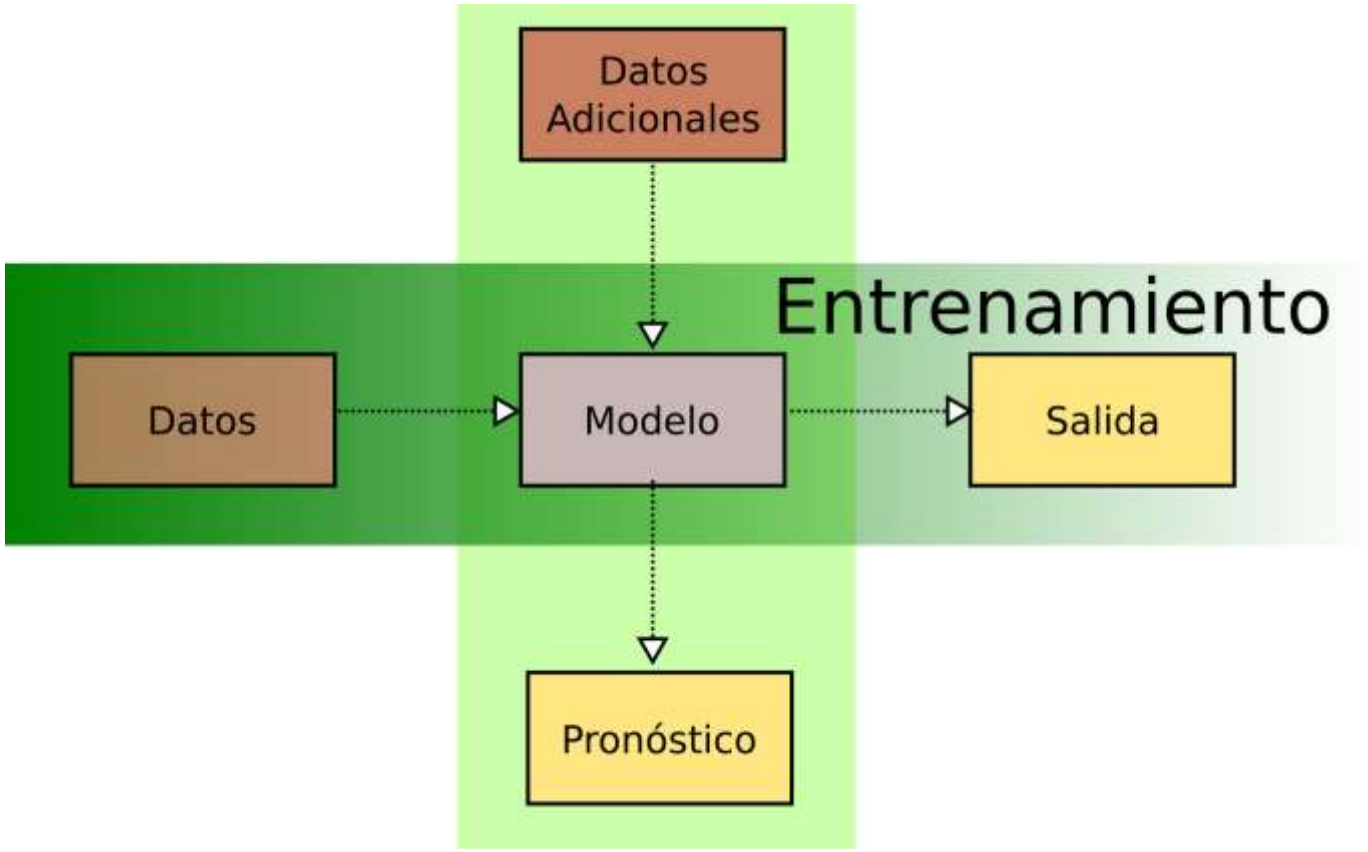
PROGRAMA.

- Redes neuronales convolucionales:
  - para clasificación de imágenes,
  - para pronóstico en series temporales.
- Redes neuronales convolucionales explicables.
- Transferencia de aprendizaje en redes neuronales convolucionales.
- Aprendizaje no supervisado: identificación de imágenes anómalas.

MOTIVACIÓN DEL CURSO.



MOTIVACIÓN DEL CURSO.



RESUMEN

- Definiciones de IA.
- Ejemplos de usos de IA y tipos de problemas.
- Desafíos asociados al Big Data.

Código:

- Esta lección no incluye códigos de ejemplo.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

DE LA REGRESIÓN LINEAL A LAS REDES NEURONALES.

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDOS.

- Funciones Básicas
- Regresión Logística
- Sobreajuste
- Resumen

FUNCIONES BÁSICAS.

FUNCIONES BÁSICAS.

- Los modelos de regresión lineal son los modelos más simples. En su forma más simple son funciones lineales de las variables de entrada.
- Los modelos de regresión lineal cumplen una doble función: el pronósticos de nuevos valores de las variables dependientes en función de las variables independientes y la construcción de un modelo interpretable de los datos experimentales.

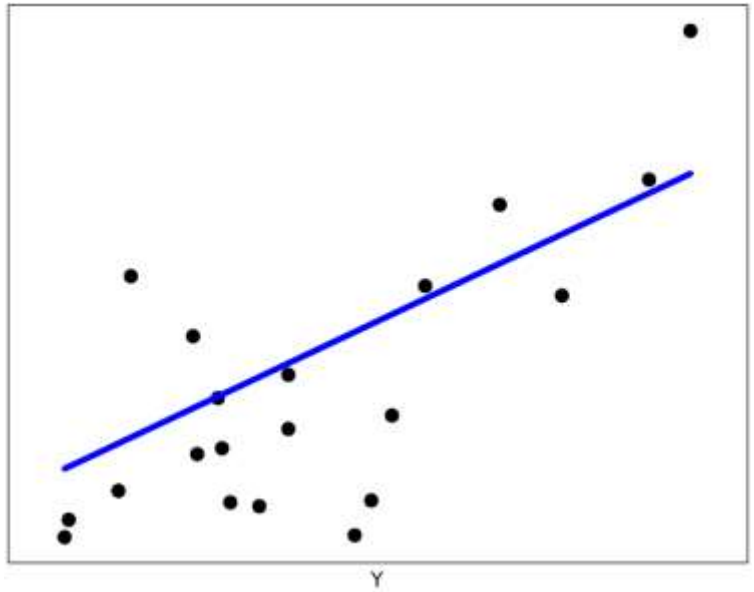
FUNCIONES BÁSICAS.

- La forma más simple de regresión lineal implica la
- combinación lineal de las variables de entrada x_i y de pesos w_i (ecuación 1).

$$\hat{y}(x, w) = w_0 + w_1 x_1 + \cdots + w_D x_D \qquad (1)$$

donde $x = (x_i, \ldots, w_D)^T$.

- La propiedad clave es que es una función lineal de los parámetros $w_0, \ldots, w_D$.
- El modelo lineal impone **limitaciones** al modelo.
- Modelo **interpretable:** $x_1 := x_1 + 1 \implies \hat{y} := \hat{y} + w_1$.



FUNCIONES BÁSICAS.

- Las limitaciones anteriormente citadas pueden ser solventadas a través de la propuesta de un modelo de regresión lineal en los parámetros $w_0, \ldots, w_D$, pero no en las variables independiente (ecuación 2).
- Esto es una extensión del modelo, considerando como entrada la combinación lineal de funciones fijas no lineales.

$$y(\mathrm{x}, \mathrm{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathrm{x}) \qquad (2)$$

donde $\phi(x)$ son conocidas como funciones básicas. El máximo valor del índice j , $M-1$ en la ecuación 2, el numero total de parámetros de este modelo se limita a M.

FUNCIONES BÁSICAS.

- La ecuación 2 puede ser escrita de forma más simplificada si se define $\phi_0(x) = 1$

(ecuación 3)

$$y(\mathrm{x}, \mathrm{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathrm{x}) = \mathrm{w}^T \phi(\mathrm{x}) \tag{3}$$

$$w = (w_0, \ldots, w_{M-1})^T \text{ y } \phi = (\phi_0, \ldots, \phi_{M-1})^T$$

donde

- Cuando $\phi_i(x) = x_i$, se tiene el caso de la regresión linear ordinaria. Se asume frecuentemente que las funciones básicas son fijas y ortogonales entre ellas. Además se busca una combinación óptima de estas funciones.

FUNCIONES BÁSICAS.

- Usando funciones básicas no lineales, se permite que la función y(x,w) sea una función no lineal del vector de entrada x.
- A pesar de esto, los modelos que siguen la ecuación 3 siguen denominándose como
- modelos líneas, ya que siguen siéndolos en w.
- De esta forma la regresión polinómica es un caso de esta forma de regresión donde $\phi_j(x) = x^j$.
- Una limitación de la regresión lineal es que son funciones globales de la variable de entrada, de forma que los cambios en una región del espacio, afectará a todas las regiones.

FUNCIONES BÁSICAS.

- Existen otras opciones para las funciones básicas, por ejemplo la ecuación 4.

$$\phi_j(x) = exp\left(-\frac{(x-\mu_j)^2}{2\,s^2}\right) \qquad (4)$$

donde μ_j gobierna la localización de la función básica en espacio de entrada, y el parámetro *s* gobierna la escala espacial. Normalmente nos referiremos a este tipo de función básica como gaussiana.

FUNCIONES BÁSICAS.

- Otra posibilidad es la que se denomina función básica sigmoide (ecuación 5). Otra posibilidad es la tangente hiperbólica que se relaciona con la sigmoide a través de la siguiente ecuación

$$tanh(\cdot) = 2\frac{1}{1+exp(\cdot)} - 1$$

$$\phi_j(x) = \frac{1}{1 + exp((x - \mu_j)/s)} \quad (5)$$

FUNCIONES BÁSICAS.

- La Regresión Polinómica flexibiliza la Regresión Lineal permitiendo relaciones entre las variables independiente y las variables dependiente con relaciones en diversos grados de potencia.
- En base a las funciones básicas, la regresión polinómica puede ser escrita como

$$y(x, w) = \sum_{j=0}^{M-1} w_j \phi_j(x), \text{ con } \phi_j(\cdot) = (\cdot)^j$$

REGRESIÓN LOGÍSTICA

REGRESIÓN LOGÍSTICA

- La Regresión Logística es un algoritmo de elevada utilidad con una amplia utilización en problemas de clasificación binaria.
- A este interés se une el hecho de que la regresión logística es un pilar básico en el aprendizaje de redes neuronales.
- El modelo de regresión logística binaria es usado para estimar la probabilidad de una respuesta binaria basada en un conjunto de variables independientes.

REGRESIÓN LOGÍSTICA

- La técnica de regresión logística se usa principalmente en problemas de clasificación binaria. Junto con los árboles de decisión y las redes neuronales es una de las técnicas más populares en clasificación.
- En regresión logística la hipótesis es elegida como una función sigmoidea:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{6}$$

la cual puede escribirse también como la combinación de las siguientes ecuaciones:

REGRESIÓN LOGÍSTICA

- Límites de la función sigmoidea son:

$$\lim_{z \to -\infty} g(z) = 0$$
$$\lim_{z \to +\infty} g(z) = 1$$

- Con esta propiedad la función $g(z)$ se puede utilizar para clasificación binaria.

REGRESIÓN LOGÍSTICA

- La escritura de la Regresión Logística:

$$h_\theta(x) = g(\theta^T x)$$
$$g(z) = \frac{1}{1+e^{-z}}$$

- muestra similitudes con las funciones básicas:

$$y(\mathrm{x}, \mathrm{w}) = f\left(\sum_{j=0}^{M-1} w_j \phi_j(\mathrm{x})\right)$$

donde f (·) es una función de activación no lineal y ϕ es la función identidad.

REGRESIÓN LOGÍSTICA

- La función sigmoidea $g(z) = \frac{1}{1+e^{-z}}$ tiene la siguiente propiedad respecto a su derivada:

$$g'(z) = g(z) \cdot (1 - g(z))$$

- Esta función de coste $J(\theta)$ se puede minimizar por cualquiera de las técnicas habituales:
- gradiente descendiente, ecuación normal, o algoritmo evolutivo.
- La función $g(z)$ tiene las siguientes características:
  - $g(z) \geq 0.5$ si $z \geq 0$, lo cual indicará la etiqueta "1"
  - $g(z) \leq 0.5$ si $z \leq 0$, lo cual indicará la etiqueta "0"

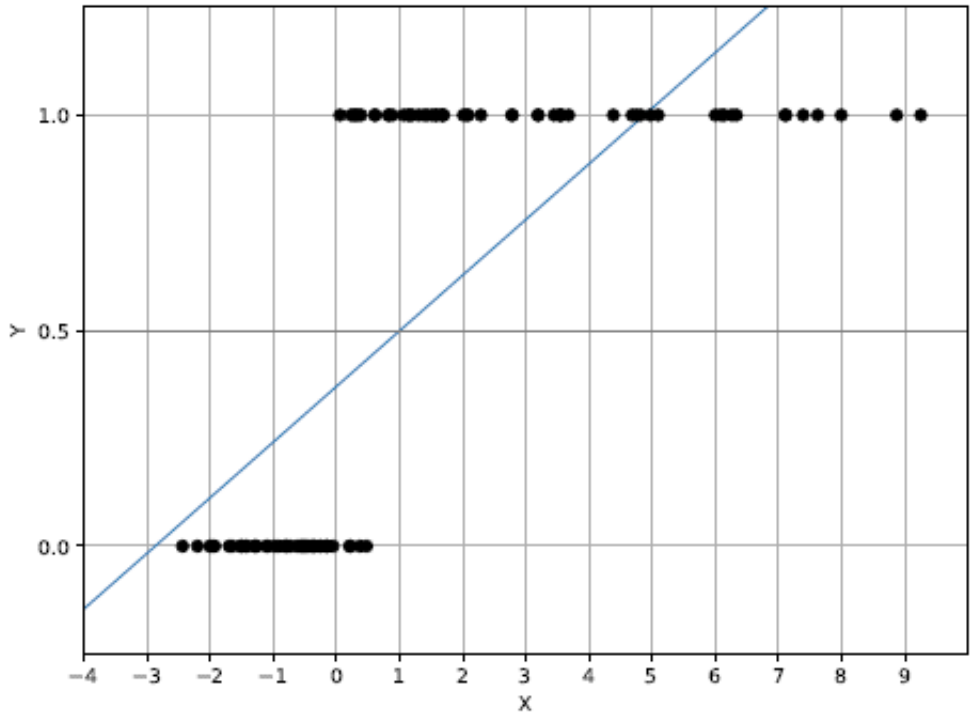
REGRESIÓN LOGÍSTICA

- Se define la frontera de decisión por aquello puntos que cumplen que $h_\theta(x) = 0.5$. La frontera de decisión es una propiedad de la función de hipótesis, y no de los datos de entrenamiento.
- De esta manera la función de coste se expresa como aparece en la ecuación 7.

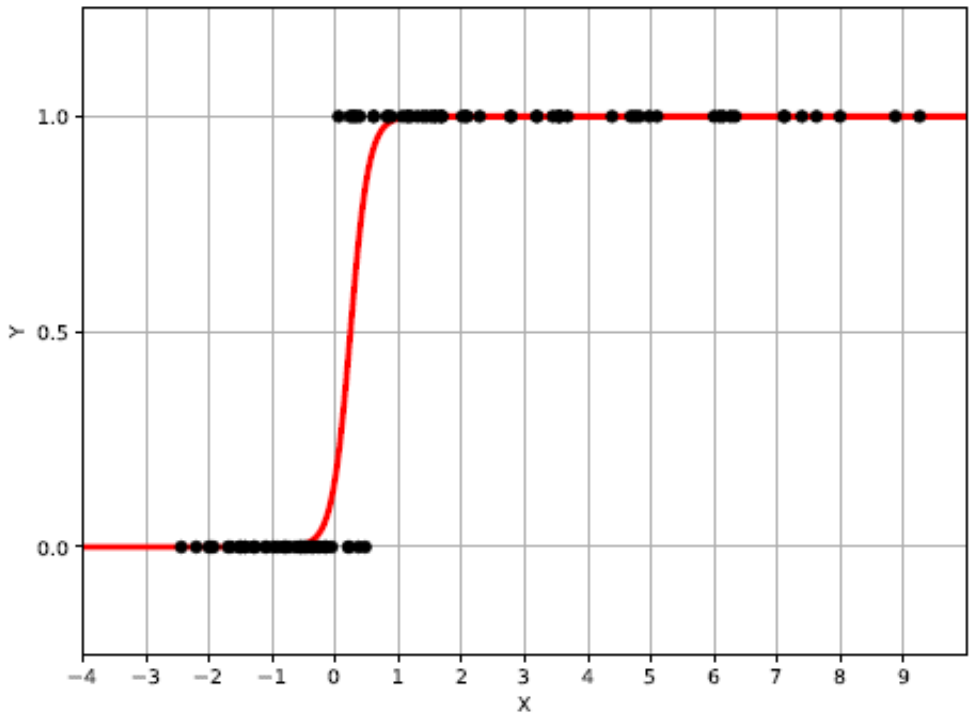
$$J(\theta) = -\frac{1}{m}\left(\sum_{i=1}^{m} y^i log(h_\theta(x^i)) + (1 - y^i) log(1 - h_\theta(x^i))\right) \quad (7)$$

- La función de coste J(θ) (ecuación 7) está compuesta por dos términos, los cuales se anulan para cada una de las etiquetas: 0 ó 1.

REGRESIÓN LOGÍSTICA



- **Figura**: Ejemplo de regresión lineal.



- Figura: Ejemplo de regresión logística.

REGRESIÓN LOGÍSTICA

$$\frac{P(y=1)}{P(y=0)} = \frac{P(y=1)}{1-P(y=1)} = e^{z}$$

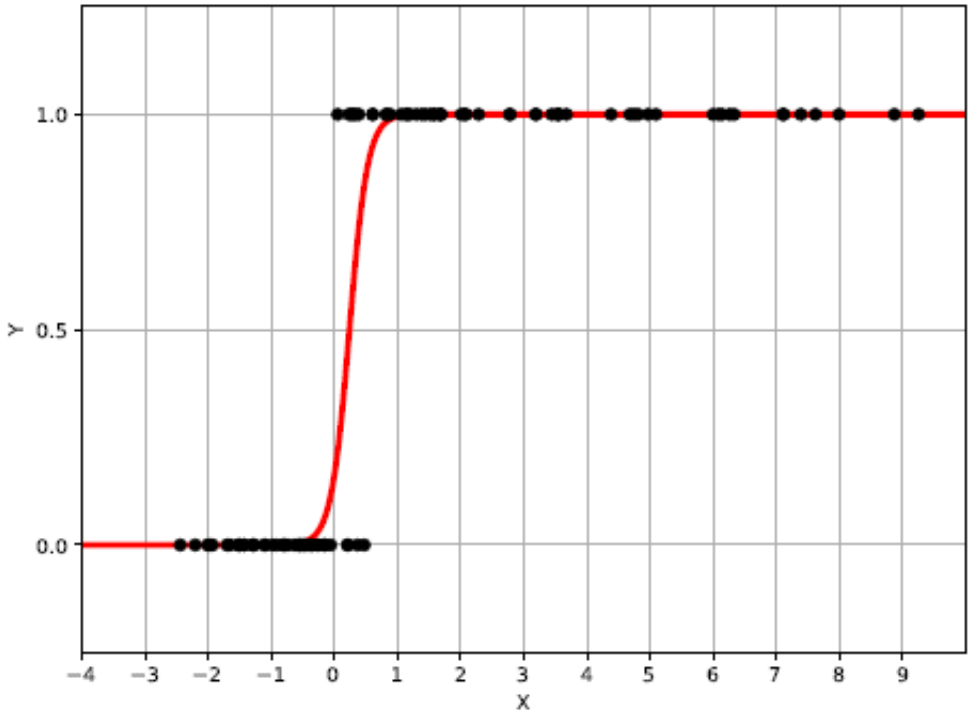


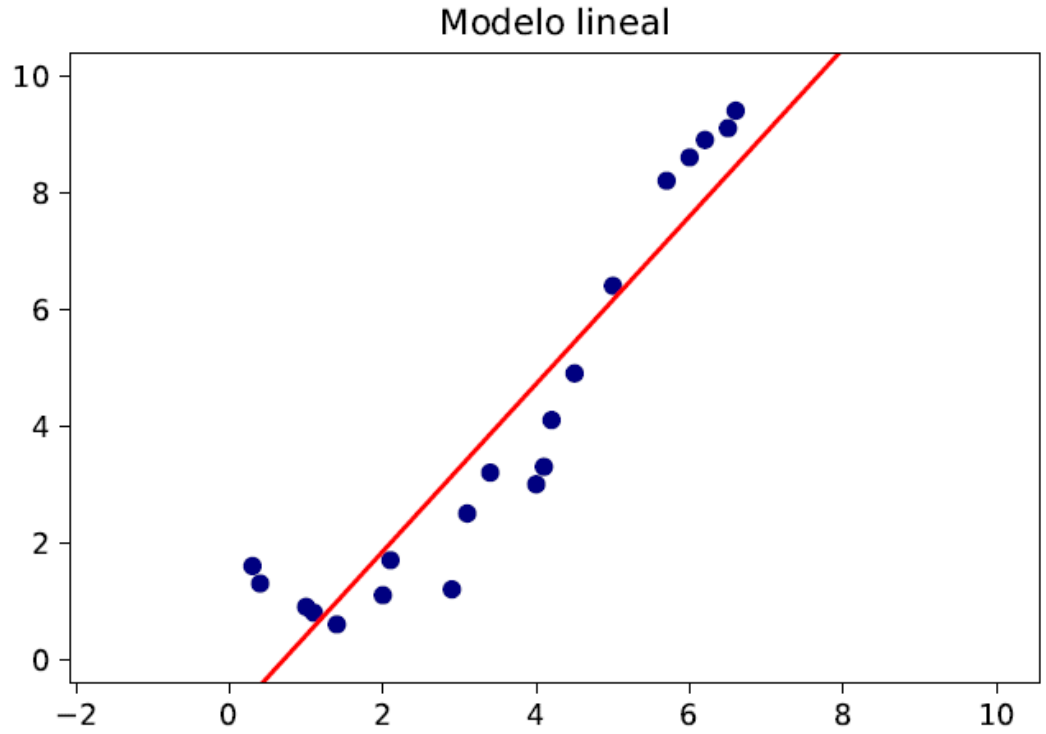
Figura: Ejemplo de regresión logística.

$$\frac{\left(\frac{P(y=1)}{P(y=0)}\right)_{x_i+1}}{\left(\frac{P(y=1)}{P(y=0)}\right)_{x_i}} = \frac{e^{z'}}{e^{z}} = \frac{e^{w_0+w_1\cdot(x_1+1)+\cdots+w_n\cdot x_n}}{e^{w_0+w_1\cdot x_1+\cdots+w_n\cdot x_n}} = e^{w_1}$$

SOBREAJUSTE.

SOBREAJUSTE.

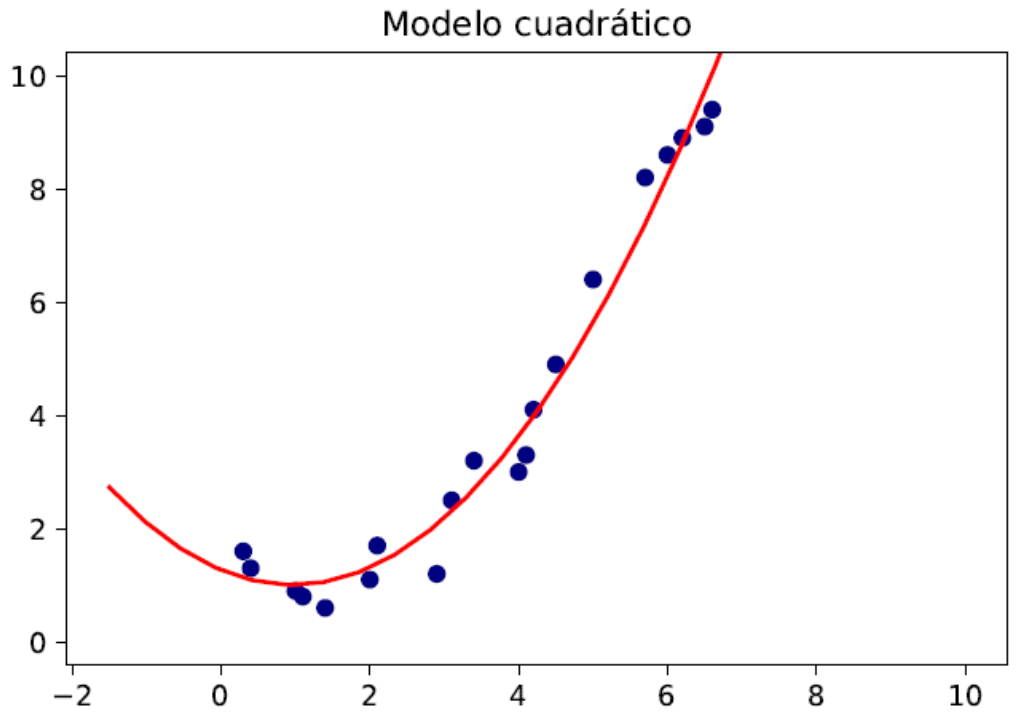
- El ajuste por un polinomio de orden lineal representa un caso de ajuste de baja calidad (sub-óptimo). Los datos no son bien representados por una recta.
- Aparentemente los datos se ajustarían mejor si la hipótesis correspondiese a un polinomio de grado mayor.
- El valor de la función de coste es 1.26.



- **Figura**: Ejemplo de ajuste lineal a un conjunto de datos, $h_\theta(x) = \theta_0 + \theta_1 \cdot x$. Como se puede apreciar es un caso de underfitting.

SOBREAJUSTE.

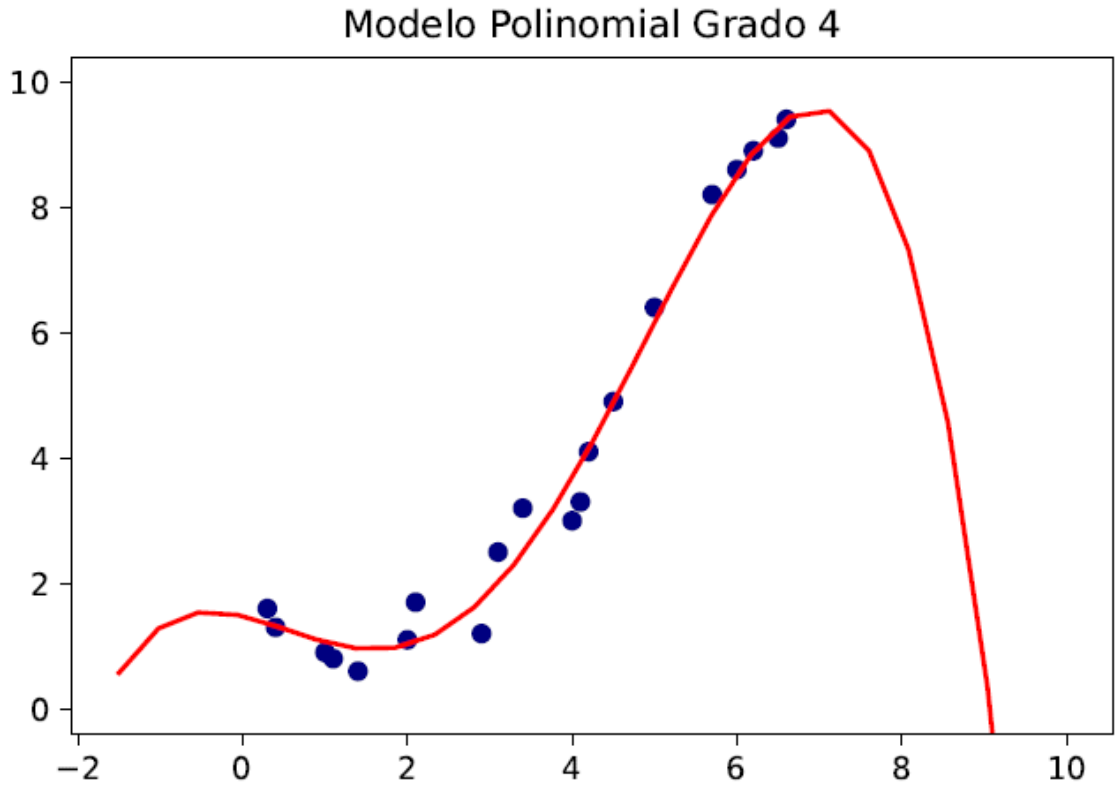
- El ajuste por un polinomio de orden cuadrático (figura 5) presenta un mejor ajuste a los datos, especialmente en comparación con el ajuste lineal.
- En este caso el valor de la función de coste es 0.24.



- **Figura**: Ejemplo de ajuste cuadrático. En este caso la hipótesis es $h_\theta(x) = \theta_0 + \theta_1 \cdot x + \theta_2 \cdot x^2$

SOBREAJUSTE.

- A medida que se incrementa el orden del polinomio, el mínimo de la función de coste disminuye constantemente.
- En este caso el valor de la función de coste es 0.14.

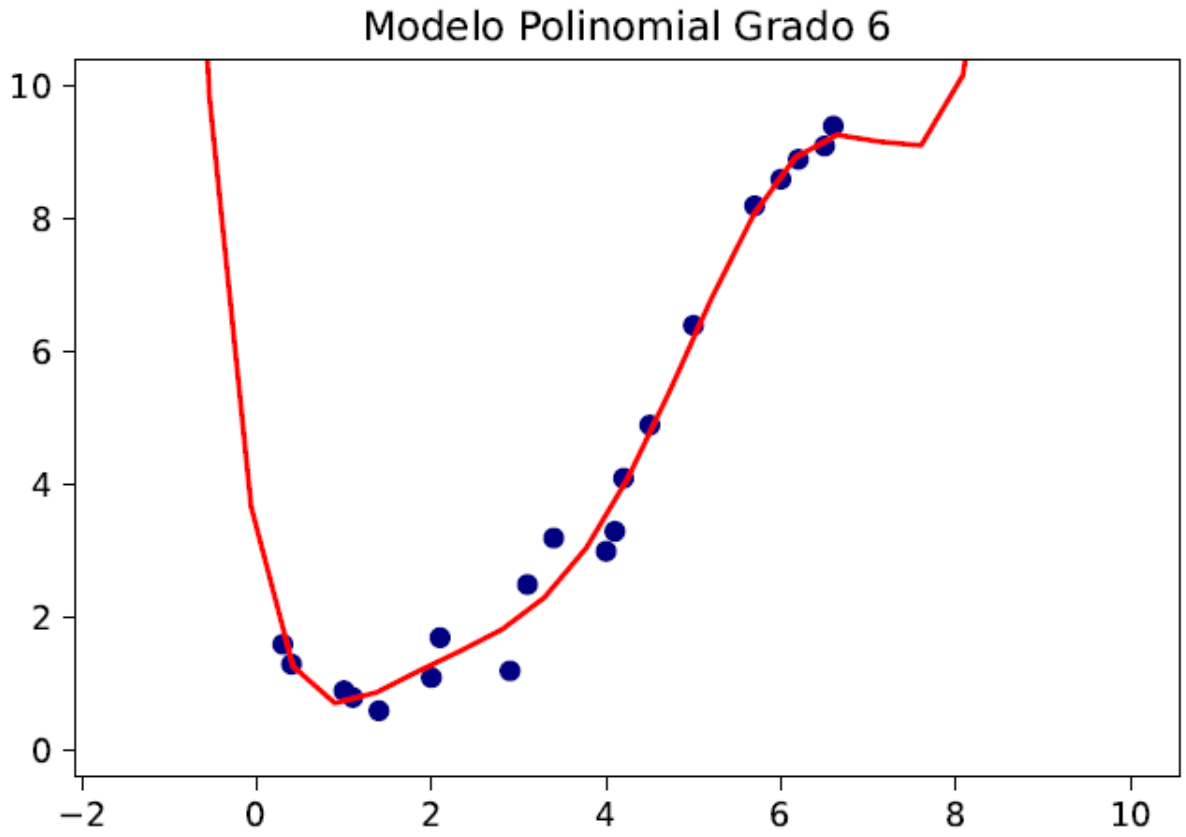


- **Figura**: Ejemplo de ajuste con un polinomio de orden cuarto. En este caso la hipótesis es

$$h_\theta(x) = \theta_0 + \theta_1 \cdot x + \theta_2 \cdot x^2 + \theta_3 \cdot x^3 + \theta_4 \cdot x^4$$

SOBREAJUSTE.

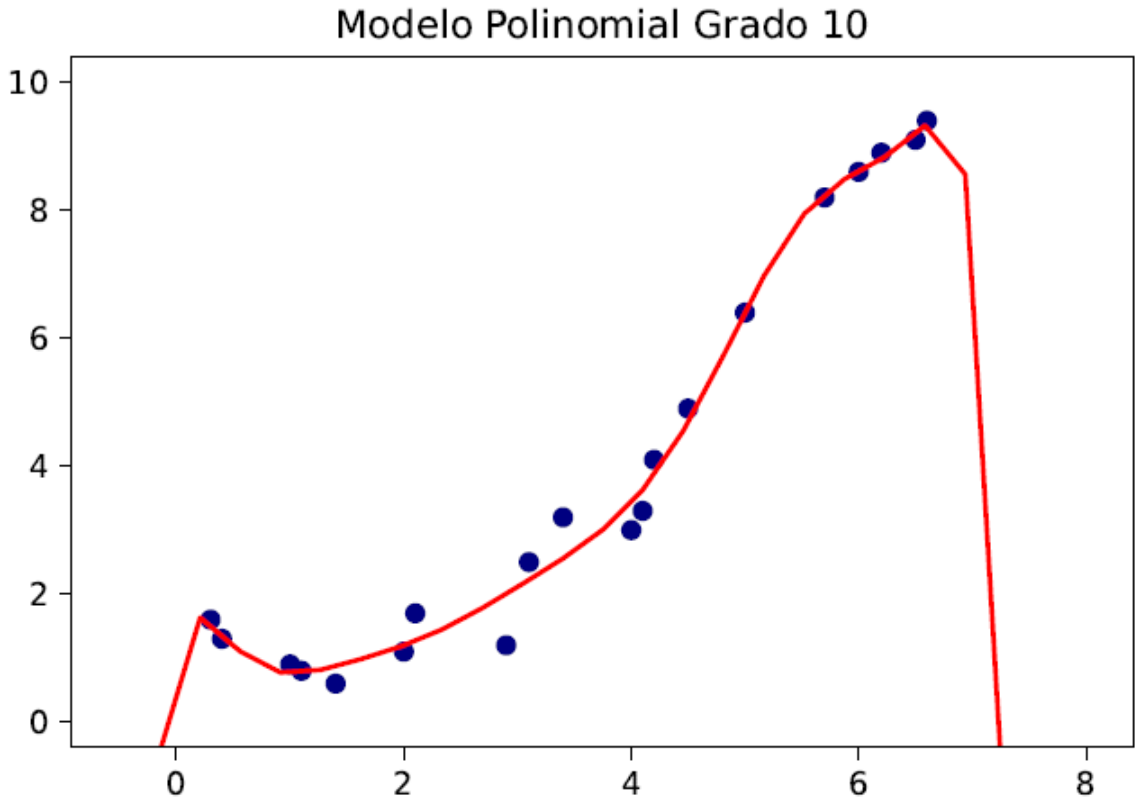
- En este caso el valor de la función de coste es 0.11.



- **Figura**: Ejemplo de ajuste con un polinomio de orden sexto.

SOBREAJUSTE.

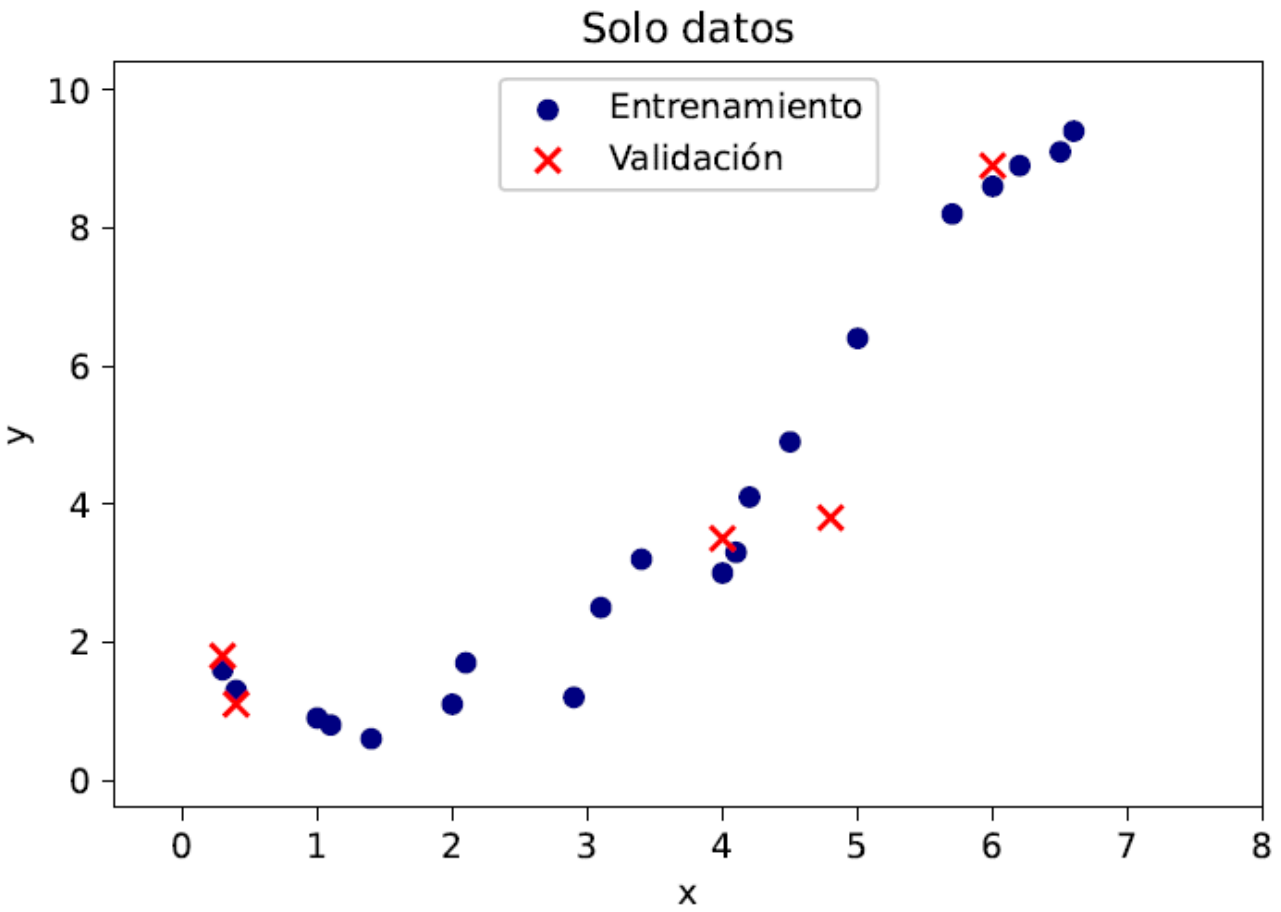
- Es esperable que para un orden muy alto, el polinomio pase por todos los puntos haciendo el valor de la función de coste nulo.
- En este caso el valor de la función de coste es 0.05.



- **Figura**: Ejemplo de ajuste con un polinomio de orden décimo.

SOBREAJUSTE.

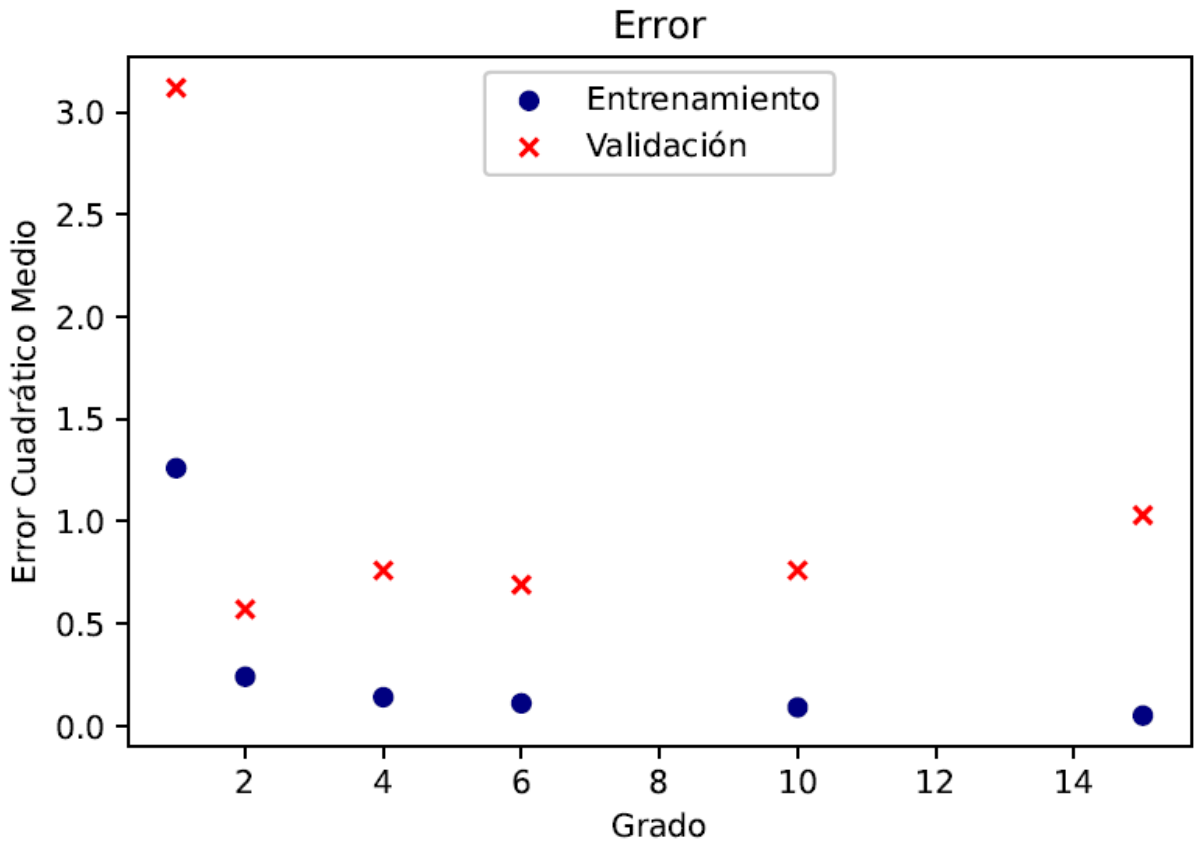
- Cómo obtener un buen ajuste para el conjunto de datos de entrenamiento (ajuste de los parámetros del modelo), y a su vez que sea generalista del comportamiento del fenómeno representado (ajuste de la complejidad del modelo).
- Para resolver esta cuestión se utiliza el error de validación. En este caso, se divide el conjunto de datos originales, apartando un pequeño subgrupo de los mismos



- **Figura**: Ejemplo de división de datos en conjunto de entrenamiento, en azul, y conjunto de validación, en rojo.

SOBREAJUSTE.

- Cómo obtener un buen ajuste para el conjunto de datos de entrenamiento (ajuste de los parámetros del modelo), y a su vez que sea generalista del comportamiento del fenómeno representado (ajuste de la complejidad del modelo).
- Para resolver esta cuestión se utiliza el error de validación. En este caso, se divide el conjunto de datos originales, apartando un pequeño subgrupo de los mismos



- **Figura**: Error de los conjuntos de entrenamiento y de validación en función del grado polinomial

SOBREAJUSTE.

Orden	Error Entrenamiento	Error Validación
Lineal	1.26	3.12
Cuadrático	0.24	0.57
Cuarto	0.14	0.76
Sexto	0.11	0.69
Décimo	0.09	0.76
Décimoquinto	0.05	1.03

- **Tabla**: Valores del error de entrenamiento y validación.

SOBREAJUSTE.

- En los modelos sobreajustados los valores absolutos de los parámetros tomas valores muy grandes
- Sin embargo, un modelo con muchos parámetros (θ_i) proporcionará una mayor flexibilidad para capturar comportamientos más complejos en los datos.
- Pero hay que evitar el sobreajuste.

Orden	θ_1	θ_2
Lineal	1.4	
Cuadrático	−0.54	0.28
Cuarto	−0.31	0.28
Sexto	−7.2	7.4
Décimo	6.3	−26
Décimoquinto	−42	93

- **Tabla**: Parámetros θ_1 y θ_2 en diversos modelos

SOBREAJUSTE.

- Dado que los parámetros de la regresión aumentan en valor absoluto con el sobreajuste, una modificación de la función de coste con una penalización proporcional al valores de los parámetros $\sum_i ||\theta_i||$

- Ejemplos de regularización son:

  Lasso o L1, $\alpha \cdot \sum_i |\theta_i|$
  Ridge o L2, $\beta \cdot \sum_i (\theta_i)^2$
  ElasticNet con la conjunción de ambos
  $\alpha \cdot \sum_i |\theta_i| + \beta \cdot \sum_i (\theta_i)^2$
  siendo α y β dos parámetros para modular la importacia de la regularización.

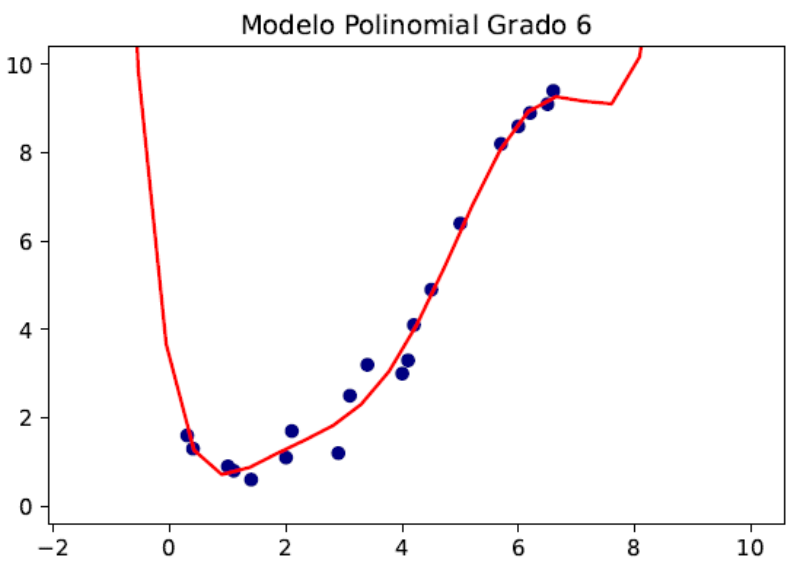
$$J = \frac{1}{N} \left(\sum_i (y_i - \hat{y}_i)^2 \right) + \beta \cdot \sum_i (\theta_i)^2 \qquad (8)$$

SOBREAJUSTE.

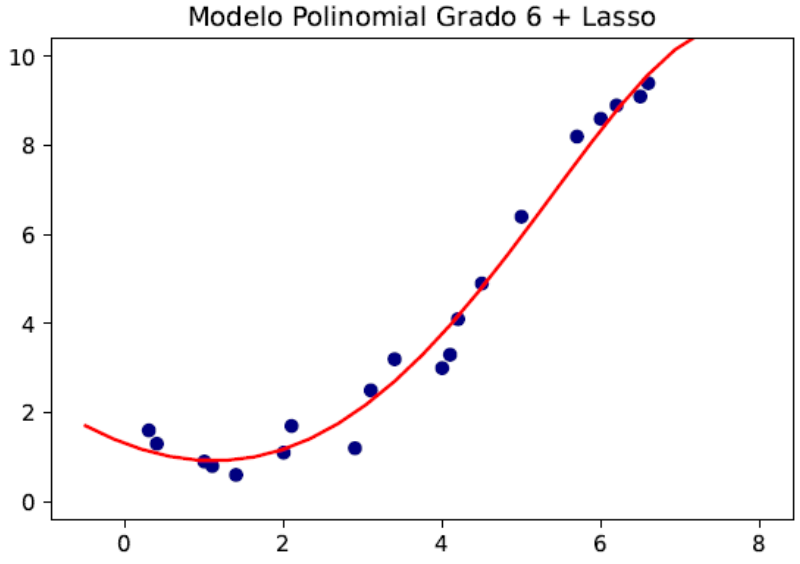
Orden	θ_1	θ_2
Sexto	-7.2	7.4
Lasso ($\alpha = 0.01$)	-0.7	0.3
Ridge ($\beta = 0.01$)	-2.1	1.3

- **Tabla**: Parámetros θ_1 y θ_2 en diversos modelos.

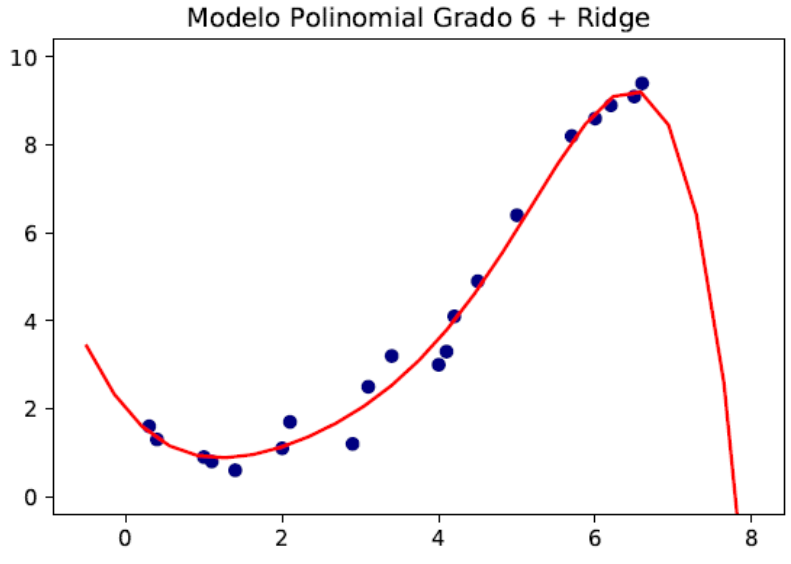
SOBREAJUSTE.



(a) RP g6



(b) RP g6 L1



(c) RP g6 L2

RESUMEN

- Propiedades de las Funciones Básicas.
- Construir Redes Neuronales a partir de Funciones Lineales Básicas.
- Se ha ilustrado el problema del sobreajuste y una posible solución mediante la regularización.

Código:

- Código ejemplo de regresión lineal:

RegresiónLineal.ipynb.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

REDES NEURONALES.

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDOS.

- Redes Neuronales
- Funciones de Activación
- Regularización
- Resumen

REDES NEURONALES.

- Compuesta por una capa de lectura de datos, *input layer*; capas ocultas (varias), *hidden layer*; y una capa de salida *output layer*.
- NN trata de mapear, o pronosticar, la salida $\hat{y}$ en función de la entrada *x*, siendo $\hat{y}$ tan similar como sea posible a *y*.

REDES NEURONALES.

- *Input Layer*, *Capa de Entrada* solo lee los datos.
- *Hidden Layers*, *Capas Ocultas* (caso para una sola capa) los datos de entrada alimentan la regresi´on lineal y las funciones g (cada una es una función no lineal).
- *Output Layer*, *Capa de Salida* es alimentada por las salidas de la capa oculta y tras la función g produce la salida $\hat{y}$.
- Existen algoritmos para optimizar los pesos w de forma que la salida $\hat{y}$ se lo más parecida posible a y.

REDES NEURONALES.

- A medida que se incremente el número de capas oculta nuestra función será capaz de mapear funciones más complejas. Pero también se corre el riesgo de caer en sobreajuste.
- Como se visualizó en la regresión polinómica, existe un riesgo de sobreajustar la red neuronal si se utilizan muchas capas ocultas. En este caso, la red producirá un muy buen resultado para los datos de entrenamiento pero deficiente para los datos de validación.
- Claramente la red entrenada será incapaz de generalizar adecuadamente para datos no vistos previamente.
- Hay que recordar que el error de nuestro modelo debe basarse en el error que se produce al pronosticar el conjunto de datos de validación y jamás el del conjunto de entrenamiento.

REDES NEURONALES.

- Los modelos lineales para regresión y para clasificación están basados en combinaciones lineales de funciones básicas fijas lineales $\phi_i(x)$ (ecuación 2).

$$\hat{y}(\mathrm{x}, \mathrm{w}) = f\left(\sum_{j=1}^{M} w_j \phi_j(\mathrm{x})\right) \tag{1}$$

donde $f(\cdot)$ es una función de activación no lineal en el caso de la clasificación y la función identidad en el caso de regresión.

REDES NEURONALES.

$$\hat{y}(x, w) = f\left(\sum_{j=1}^{M} w_j \phi_j(x)\right) \quad (2)$$

- La idea subyacente es extender este modelo haciendo que las funciones básicas $\phi_j(x)$ dependientes de parámetros y entonces permitir a estos parámetros ser optimizados durante el entrenamiento junto con los coeficientes w_j .
- Hay múltiples formas de construir estas funciones básica paramétricas no lineales. Las redes neuronales usan funciones básicas que siguen la forma de la ecuación 2, de forma que cada función básica es a su vez una función no lineal de una combinación lineal de entradas, donde los coeficientes de la combinación lineal son parámetros adaptativos w_j

REDES NEURONALES.

- Esto conduce a modelo básico de red neuronal, el cual puede ser descrito como una serie de transformaciones funcionales. El primer paso es construir M combinaciones lineales de las variables de entrada $x_1, \ldots, x_D$ (ecuación 3).

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \tag{3}$$

donde $j = 1, \ldots, M$, y el superíndice (1) indica que los parámetros son de la primera capa de la red neuronal, los parámetros $w^{(1)}_{ji}$ se denominan pesos, y a las cantidades a_j se denomina activaciones.

REDES NEURONALES.

- Cada una de estas cantidades son transformada usando una función de activación diferenciable no lineal $h(\cdot)$ para producir la ecuación 4.

$$z_j = h(a_j) \tag{4}$$

- Estas cantidades, z_j , corresponde a las salidas de las funciones básicas en la ecuación 2, que en el contexto de las redes neuronales se denominan unidades ocultas, *hidden units*. La función no lineal $h(\cdot)$ suele ser funciones con forma sigmoidea, por ejemplo, la función logística sigmoidea, la tangente hiperbólica o la función relu. Siguiendo la idea de la ecuación 2, estos valores se vuelven a combinar linealmente para producir las salidas de la unidades de activación (educación 5).

REDES NEURONALES.

- Siguiendo la idea de la ecuación 2, estos valores se vuelven a combinar linealmente para producir las salidas de la unidades de activación (educación 5).

$$a_k = \sum_{j=1}^{M} w_{kj}^{(2)} z_j + w_{k0}^{(2)} \tag{5}$$

donde $k = 1, \ldots, K$, y K es el número total de salidas. Esta transformación corresponde con la segunda capa de la red neuronal, y de nuevo los parámetros w_{kj} son los pesos.

Finalmente, las salidas de las unidades de activación son transformadas de nuevo para proporcionar el resultado final y_k . Aquí se supone que la red neuronal solo tiene dos capas ocultas.

REDES NEURONALES.

- La elección de la función de activación de la última capa vendrá determinada por el tipo de problema: clasificación o regresión. Para las capas ocultas la elección depende del rendimiento de la red. Todo lo anterior puede ser combinado, de forma que la ecuación 6 o de forma más compacta como la ecuación 7.

$$\hat{y}(\mathrm{x}, \mathrm{w}) = f\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)})\right) \tag{6}$$

$$\hat{y}(\mathrm{x}, \mathrm{w}) = f\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right) \tag{7}$$

REDES NEURONALES.

- Debido a la forma de esta ecuación, este modelo se denomina perceptrón multicapa (MLP).
- El modelo de esta ecuación puede ser ampliado formalmente con la adición de más capas.

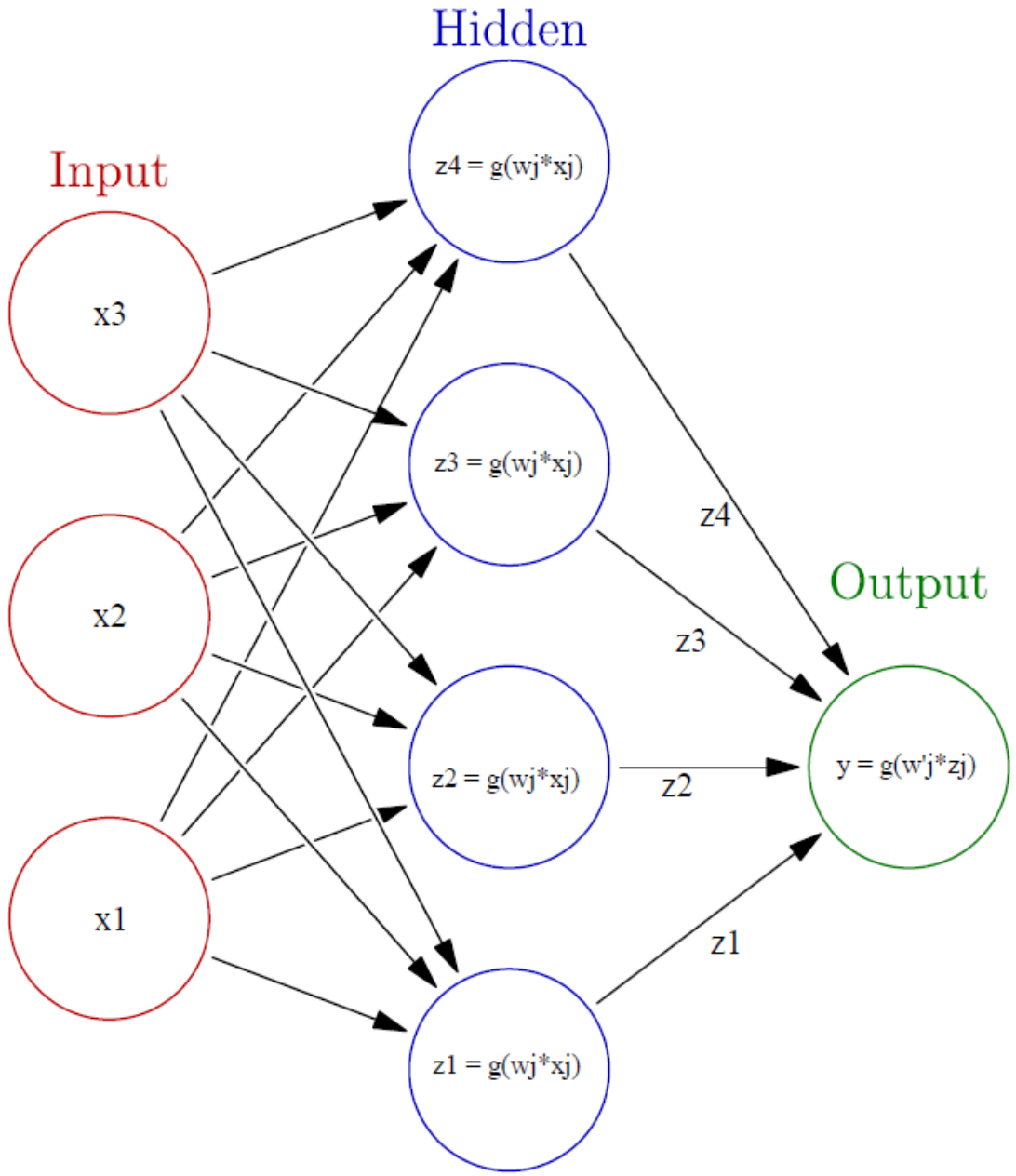
$$\hat{y}(\mathrm{x}, \mathrm{w}) = f\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

- Sin embargo, el rendimiento de los modelos puede no mejorar con la adición de más capas.

REDES NEURONALES.

$$\hat{y}(x, w) = f\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

siendo $f(\cdot)$ la función sigmoidea, tanh, lineal, softmax o relu, y $h(\cdot)$ cualquiera de las anteriores excepto la función lineal o la softmax.



REDES NEURONALES.

- Una configuración demasiado simple de las capas ocultas determinará una incapacidad del modelo para capturar la información de los datos de entrada.
- Una configuración demasiado compleja (gran número de capas ocultas con muchas neuronas) conducirá a una pobre generalización del modelo debido a un elevado sobreajuste (*overfitting*) del modelo a los datos. Además muchas capas conduce a la imposibilidad de ajustar los pesos debido al desvanecimiento del gradiente.

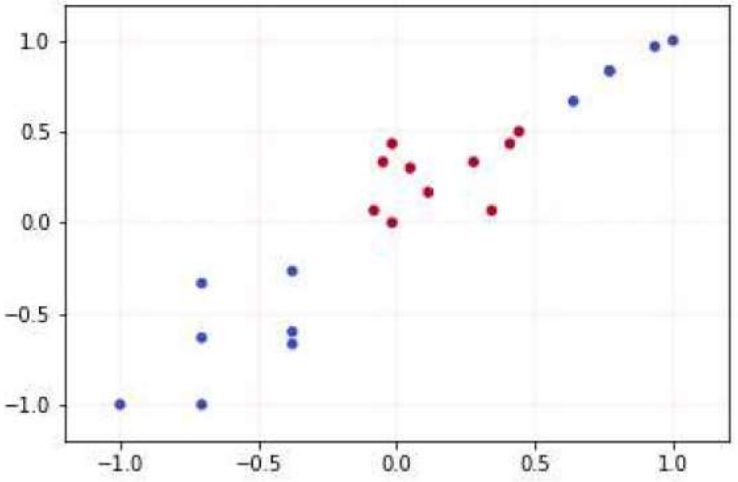
TEOREMA DE APROXIMACIÓN UNIVERSAL

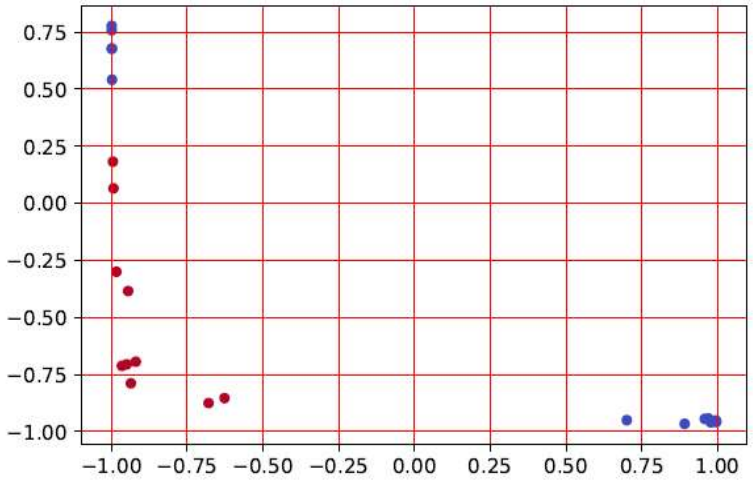
- El teorema de Aproximación Universal dice que una red neuronal con un número finito de neuronas en su única capa oculta $\mathcal{N}_{w,c} : \mathbb{R}^n \to \mathbb{R}^m$

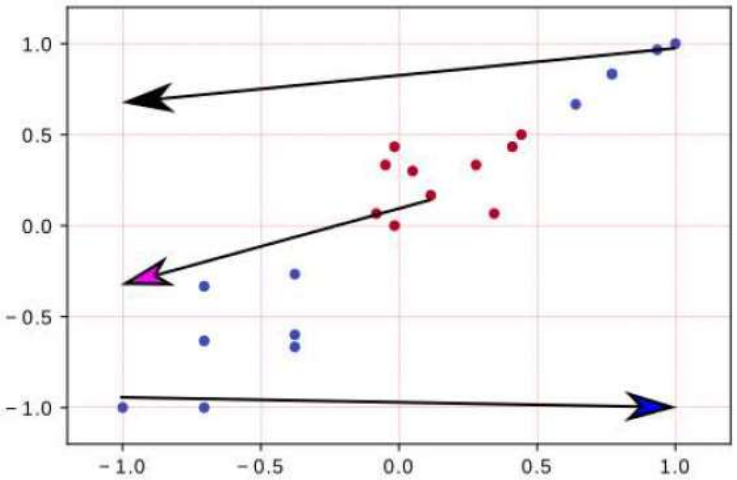
  es una aproximación a cualquier función contir $g,\ ||g - \mathcal{N}|| < \epsilon$

DISTORSIÓN DEL ESPACIO

- En el ejemplo de clasificación mostrado no es linealmente separable en el espacio original (panel izquierdo).
- Una vez distorsionado el espacio por la red neuronal, la última capa oculta tiene dos neuronas y eso es lo que se muestra en el panel central, es linealmente separable.

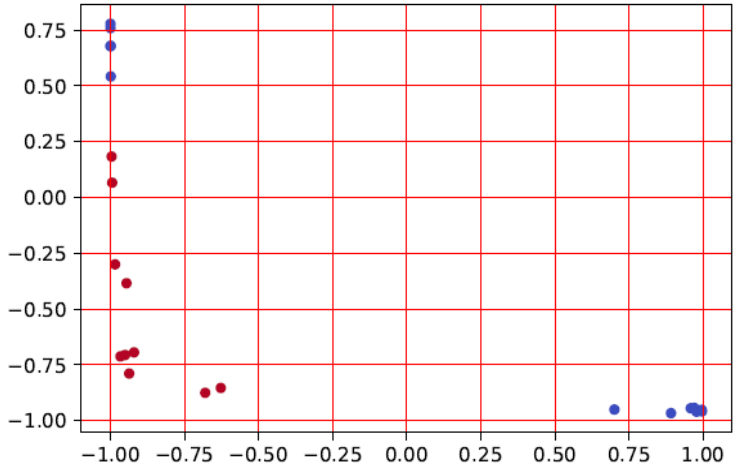


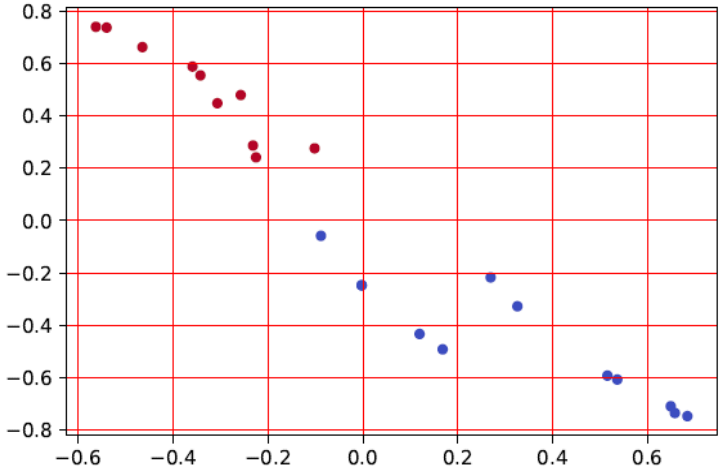


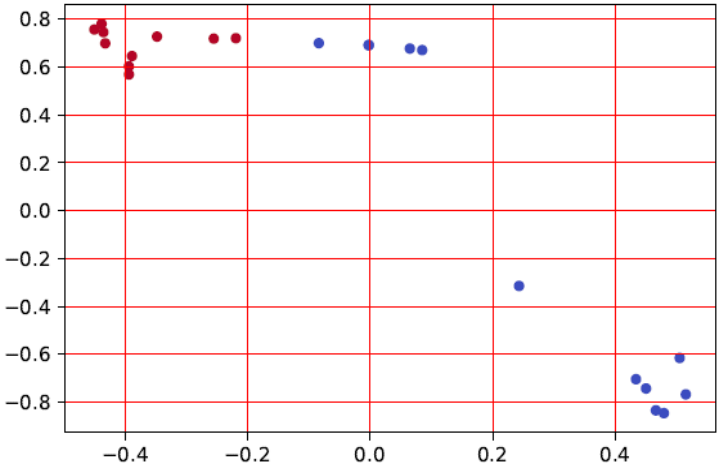


DISTORSIÓN DEL ESPACIO

- Diferentes ejecuciones puede producir distorsiones diferentes y por lo tanto diferentes soluciones.
- Cada uno de ellos es un homeomorfismo.







FUNCIONES DE ACTIVACIÓN

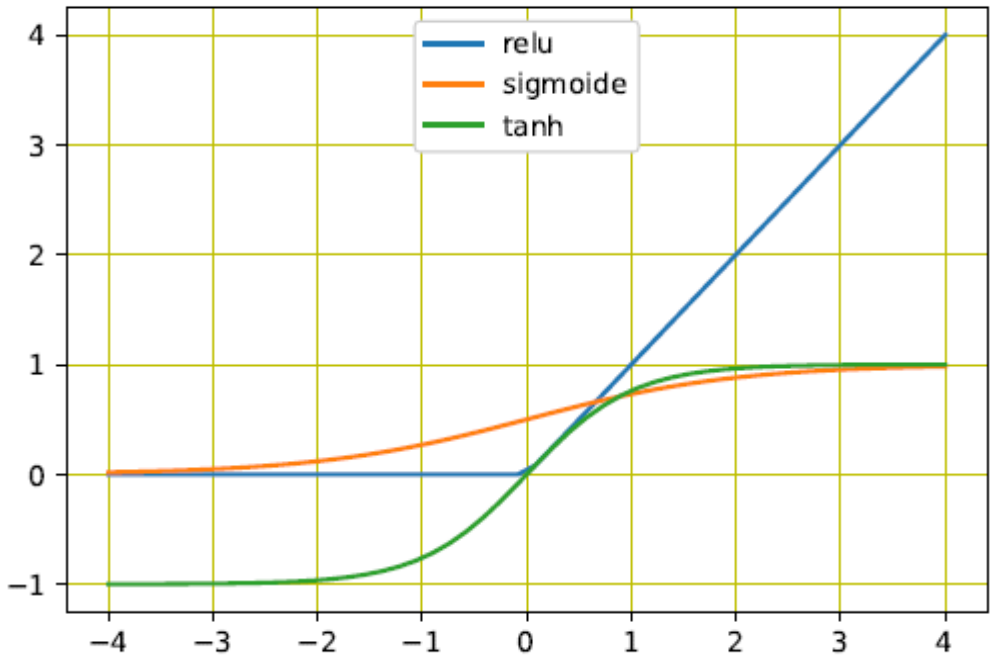
MLP

- La función de activación en las capas ocultas g preferentemente tiene que ser una función no lineal: tanh

$$y = \frac{exp(z)-exp(-z)}{exp(z)+exp(-z)}$$, sigmoide

$$y = \frac{1}{1+exp(-z)}$$, relu y = max(0, x).

- La convergencia de la red durante el entrenamiento, la duración del mismo, y el rendimiento dependerán de esta elección.



MLP

- La función de activación en la capa de salida f depende que la naturaleza de nuestra salida.
- Para problemas de regresión: función lineal (salida de $-\infty$ a $+\infty$).
- Para problemas de clasificación binaria: tanh (salida $-1 \leq y_i \leq 1$).
- Para problemas de clasificación no binaria (probabilidad equidistribuida entre las etiquetas):

softmax $y = \frac{exp(z_i)}{\sum_i exp(z_i)}$ (salida $0 \leq y_i \leq 1$)

- Para problemas donde la salida es una probabilidad: sigmoide (salida $0 \leq yi \leq 1$).

MLP

- Los criterios de entrenamiento para entrenar la red son:
  - Error cuadrático medio $= \frac{1}{N} \sum (y_i - \hat{y}_i)^2$
  - Error absoluto medio $= \frac{1}{N} \sum |y_i - \hat{y}_i|$
  - Entropía cruzada entre dos clases.
  - Entropía cruzada entre múltiples clases.

MLP

- La forma más simple para la optimización de los pesos de las neuronas es *back propagation*.
- Se basa en minimizar (localmente) la función de error *E* con respecto a los pesc

$$w_i^{t+1} = w_i^t - \eta \cdot \frac{\partial E^t}{\partial w_i^t}$$

- Se puede utilizar solo un subconjunto aleatorio del conjunto de entrenamiento en cada época (menos sobreajuste y más rápido).

REGULARIZACIÓN EN REDES NEURONALES

REGULARIZACIÓN EN REDES NEURONALES

- Se suele acepta que el número de neuronas (unidades) en la capa de entrada y de salida dependerá de la dimensionalidad de los datos, mientras que el número de unidades ocultas es un parámetro libre que puede ser ajustado para producir la mejor predicción posible.
- Aunque se espera que la minimización del error de entrenamiento, el error de generalización, sea mínimo para un cierto valor del número de neuronas, esta no es una función sencilla ni directa. Y esto se debe a la presencia de mínimos locales.
- Existen mecanismos para controlar la complejidad de la red neuronal y así evitar el sobreajuste.

REGULARIZACIÓN EN REDES NEURONALES

- Una estrategia válida es sobredimensionar el número de parámetro *s* y añadir un término de regularización a la función de error.
- El regularizador (decaimiento de peso, λ) más simple es el cuadrático (ecuación 8). La complejidad efectiva del modelo es determinada por el coeficiente de regularización λ.
- Este parámetro puede ser interpretado como el logaritmo negativo de una distribución gaussiana de priores con media nula sobre el vector de pesos *w*.

$$\tilde{E}(\mathbf{w}) = E(\mathbf{w}) + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \tag{8}$$

PARADA TEMPRANA.

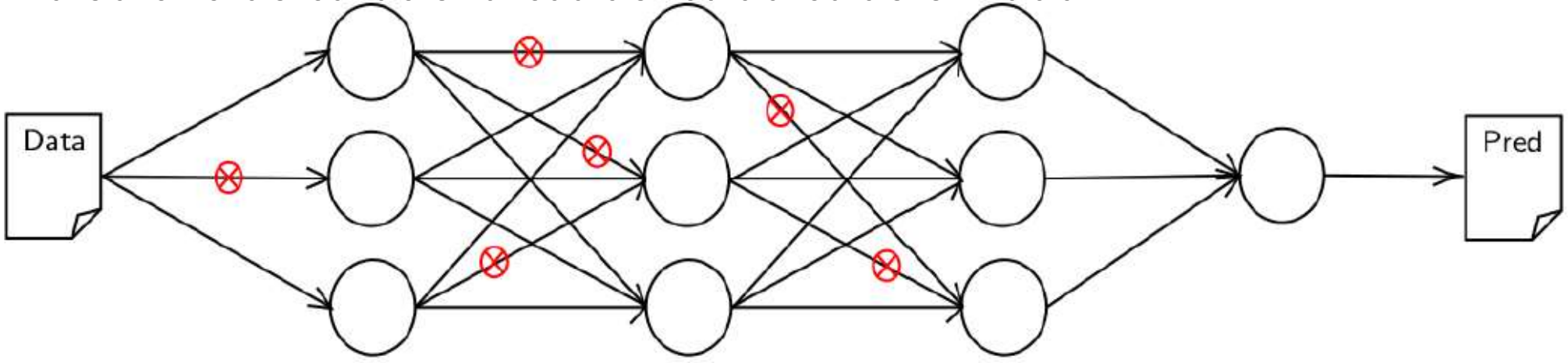
- Una alternativa a la regularización como forma efectiva de control de la complejidad de una red es el procedimiento de parada temprana (*early stopping*).
- El entrenamiento de modelos de red no lineal corresponde con una reducción iterativa del error de la función definida con respecto a un conjunto de entrenamiento. Para muchos algoritmos de optimización usados en el entrenamiento de redes, como el gradiente conjugado, el error no es reducido con el índice de la interacción.
- Sin embargo, el error medido con respecto a un conjunto de datos independientes, llamado conjunto de validación, frecuentemente muestra un decrecimiento primero, seguido por un incremento cuando la red comienza a sobreajustarse.

PARADA TEMPRANA.

- Por lo tanto, el entrenamiento puede ser parado en el punto del error más bajo con respecto al conjunto de validación. De esta forma se obtiene una red que tiene un rendimiento bueno para datos generalizados.
- El carácter de la red es en este caso explicado cualitativamente en términos del número efectivo de grados de libertad de la red, el cual comienza siendo pequeño y entonces crece durante el proceso de entrenamiento, correspondiente a un incremento constante en la complejidad efectiva del modelo.
- La parada del entrenamiento antes del mínimo del error de entrenamiento representa un modo de limitar la complejidad efectiva de la red.

DROPOUT.

- Otra de las técnicas más frecuentemente utilizadas para evitar el sobreajuste es el *dropout*.
- El *dropout* consiste en apagar aleatoriamente algunos pesos en capa época de entrenamiento.
- Con esta técnica se fuerza a que la red aprenda de un subconjunto aleatorio de características de los datos de entrada



RESUMEN.

- Se ha mostrado cómo se construyen, a partir de funciones básicas, las redes neuronales.
- También se han mostrado las funciones de activación más comunes y su relación con la capa de salida.

Códigos:

- Código ejemplo de red neuronal en problema de regresión:

  MLP Regressor Add2.ipynb.

- Código ejemplo de red neuronal en problema de clasificación:

  MLP Clasificador M3.ipynb.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

REDES NEURONALES PARA SERIES TEMPORALES.

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDOS.

- Series Temporales
- Resumen

SERIES TEMPORALES

SERIES TEMPORALES.

- Numerosas medidas científicas son recolectadas a lo largo de un periodo de tiempo.
- Este conjunto de medidas constituye una series temporal, y su análisis es de gran interés.
- Como consecuencia de este análisis se produce una mejor comprensión de los mecanismos que generan los datos, o una mejora en la capacidad de predicción de los valores futuros de la variable.

SERIES TEMPORALES.

Definición

- Se define una serie temporal como una secuencia de medidas de una variable o varias variables recolectada a lo largo del tiempo.

SERIES TEMPORALES.

0,1,2,3,3,2,2,1,0,1,0,1,2,2,3

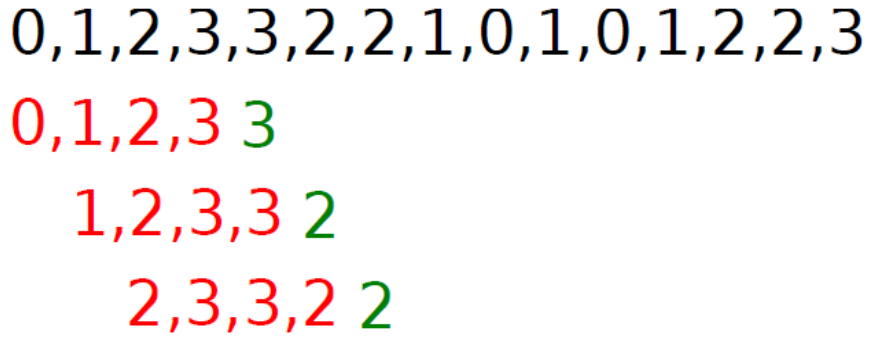
0,1,2,3 3

1,2,3,3 2

2,3,3,2 2

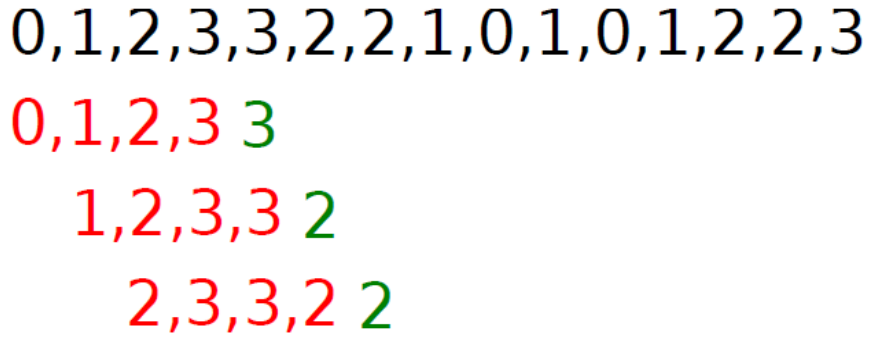
SERIES TEMPORALES.

- En una serie temporal hay que separar las variables independientes y dependientes.
- Es lógico pensar que un conjunto de observaciones anteriores a un instante temporal, tiene información de permitirá un pronóstico de la siguiente observación de la serie.
- A medida que retrocedamos en el tiempo, esta correlación disminuirá, limitando el tamaño de esta ventana.



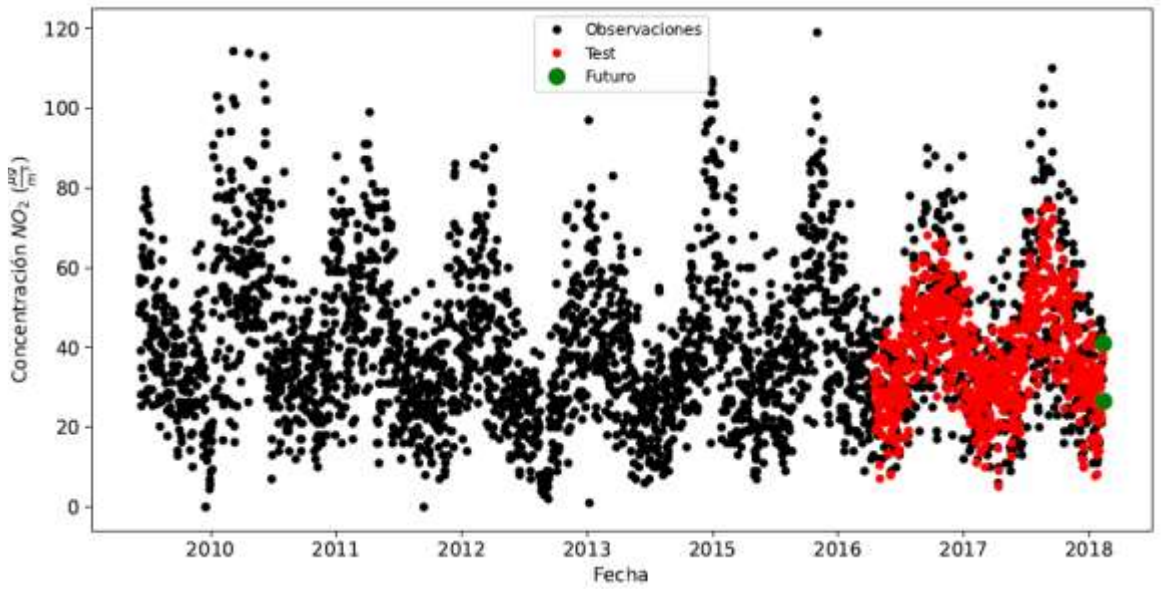
SERIES TEMPORALES.

- Ventanas pequeñas harán que nuestras predicciones sean ruidosas, adaptándose rápidamente a pequeños cambios.
- Ventanas grandes capturan el comportamiento global de la serie.
- En series temporales periódicas, la ventana debería incluir al menos un periodo.



SERIES TEMPORALES.

- Ejemplo de pronóstico en series temporales: concentración media diaria de NO_2 en Madrid.



RESUMEN

- Se ha mostrado como tratar los datos de series temporales para generar pronósticos con redes neuronales.

Código:

- Código ejemplo de red neuronal en problema de pronóstico en series temporales:

MLP_TS_ContMadrid.ipynb.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

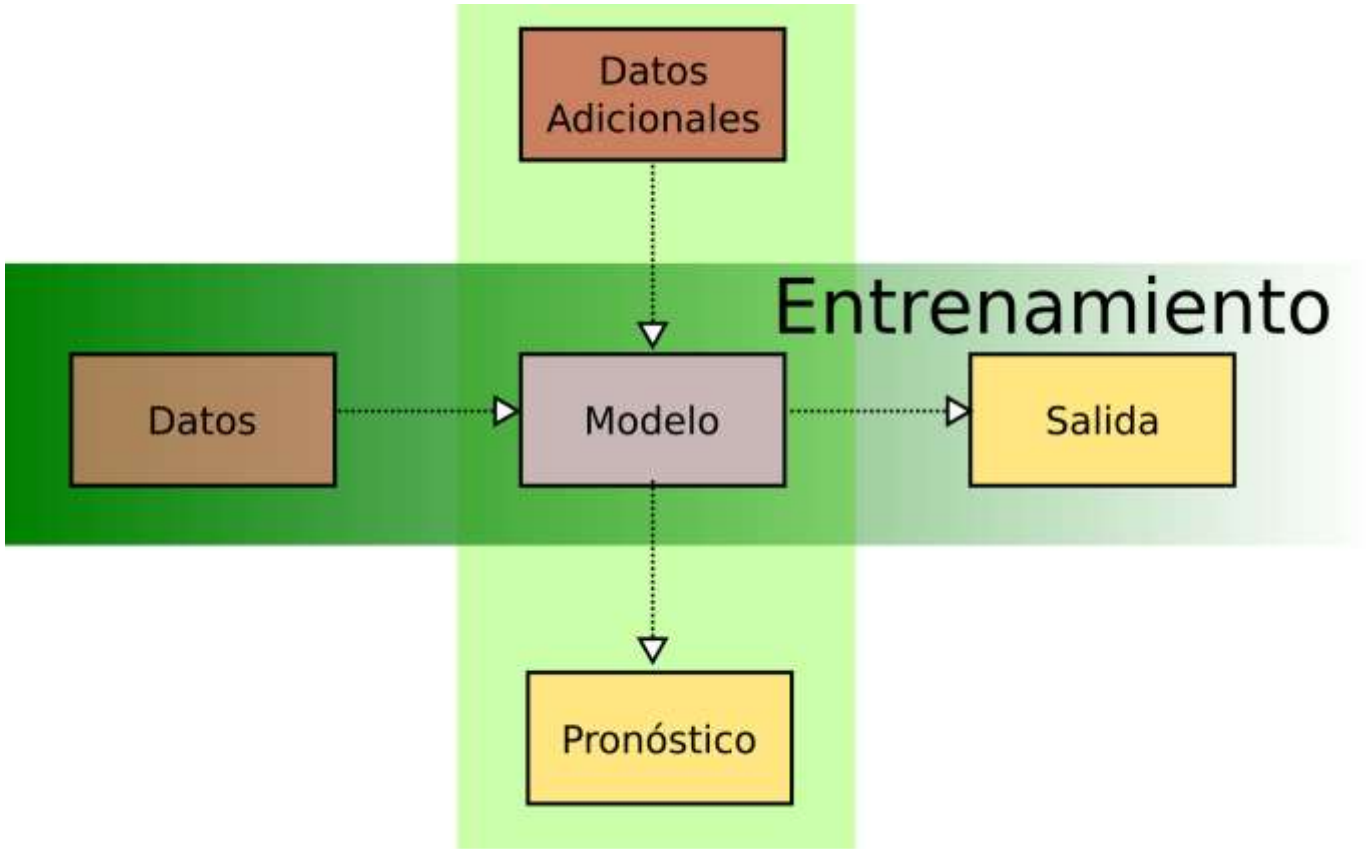
INGELIGENCIA ARTIFICIAL EXPLICABLE. ALGORITMO DE GARSON.

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDOS.

- XAI
- Garson
- Pesos conectados
- Resumen

XAI.


Datos
Adicionales
Entrenamiento
Datos
Modelo
Salida
Pronóstico

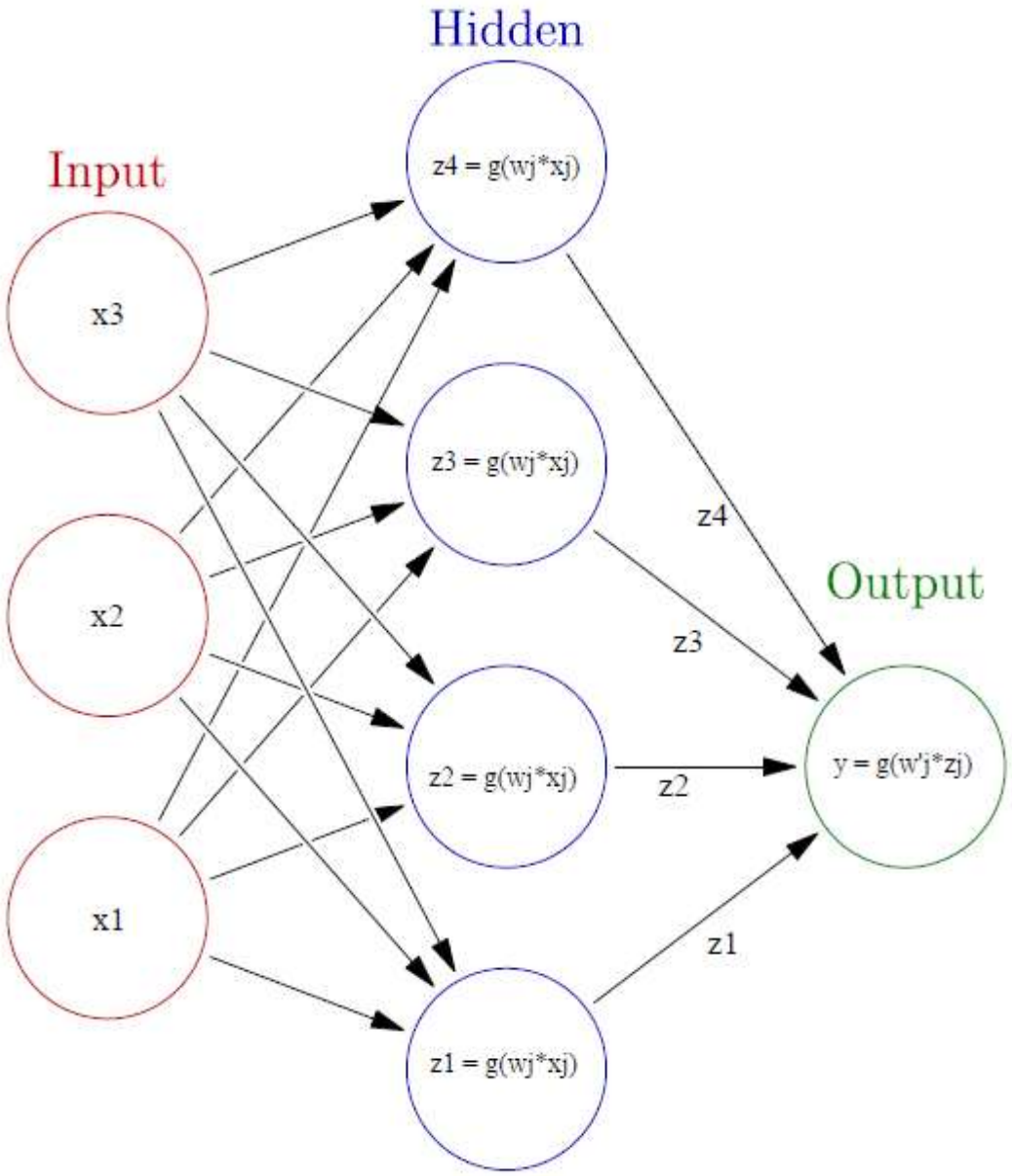
XAI.

- Una de las críticas más importantes sobre los modelos basados en redes Neuronales es su falta de interpretabilidad, y por consiguiente de transparencia.
- Se han propuesto estrategias para evaluar la importancia de las variables de entrada.

1. Garson, G.D., *Interpreting neural-network connection weights*. Artif. Intell. Expert 6, *47–51* (1991).
2. Ibrahim, OM. 2013. *A comparison of methods for assessing the relative importance of input variables in artificial neural networks*. Journal of Applied Sciences Research, 9(11): 5692-5700.
3. J.D. Olden et al. *An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data*. Ecol. Model. 178 389-397 (2004).
4. Olden, J.D., Jackson, D.A., *Illuminating the "black box": understanding variable contributions in artificial neural networks*. Ecol. Model. 154, 135–150 (2002).

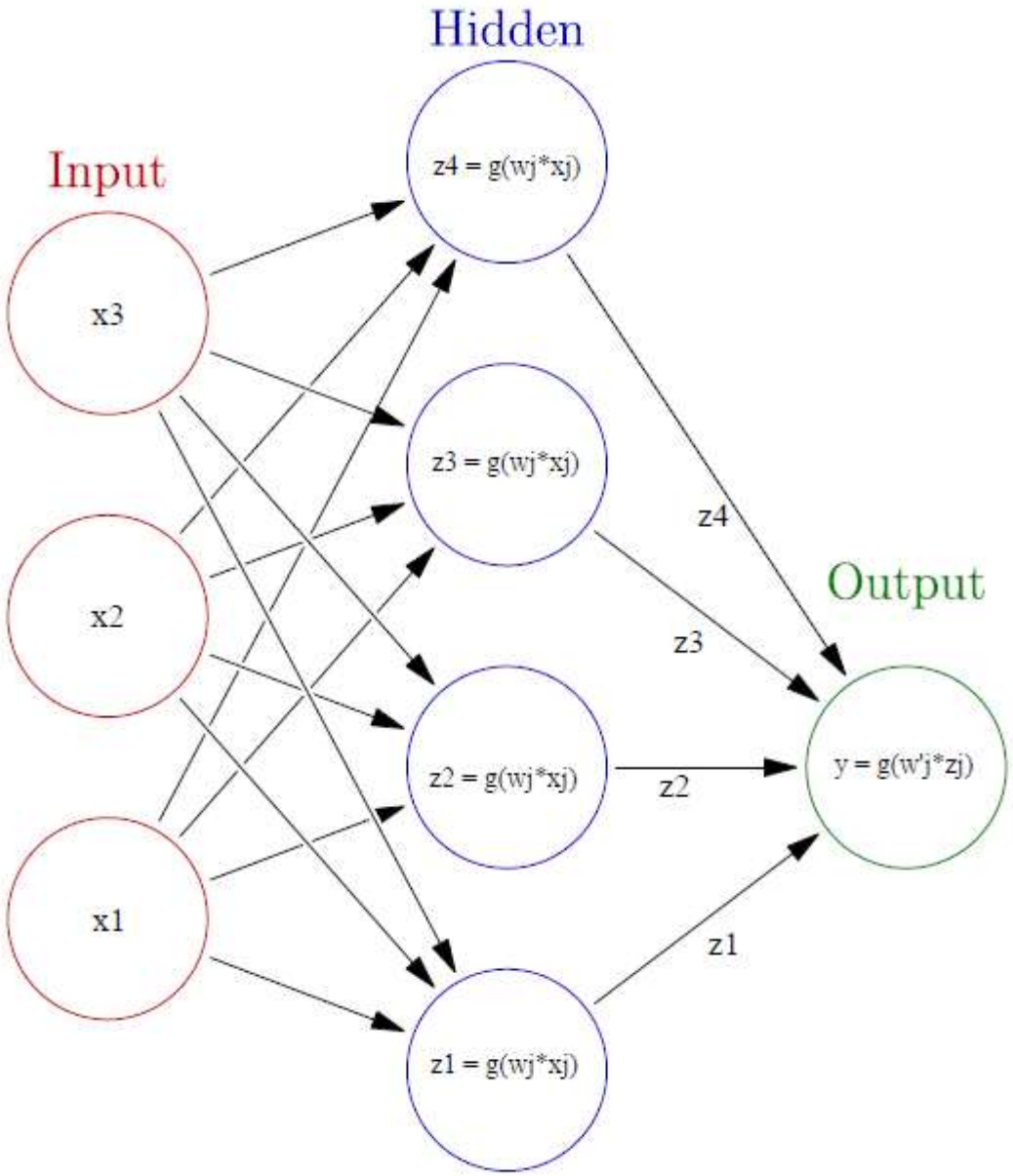
ALGORITMO DE GARSON.

- Propuesto por G.D. Garson en 1991.
- Limitado a redes neuronales de una sola capa oculta.
- Aunque se ha demostrado que una red neuronal de una capa son aproximadores universales a cualquier función continua.



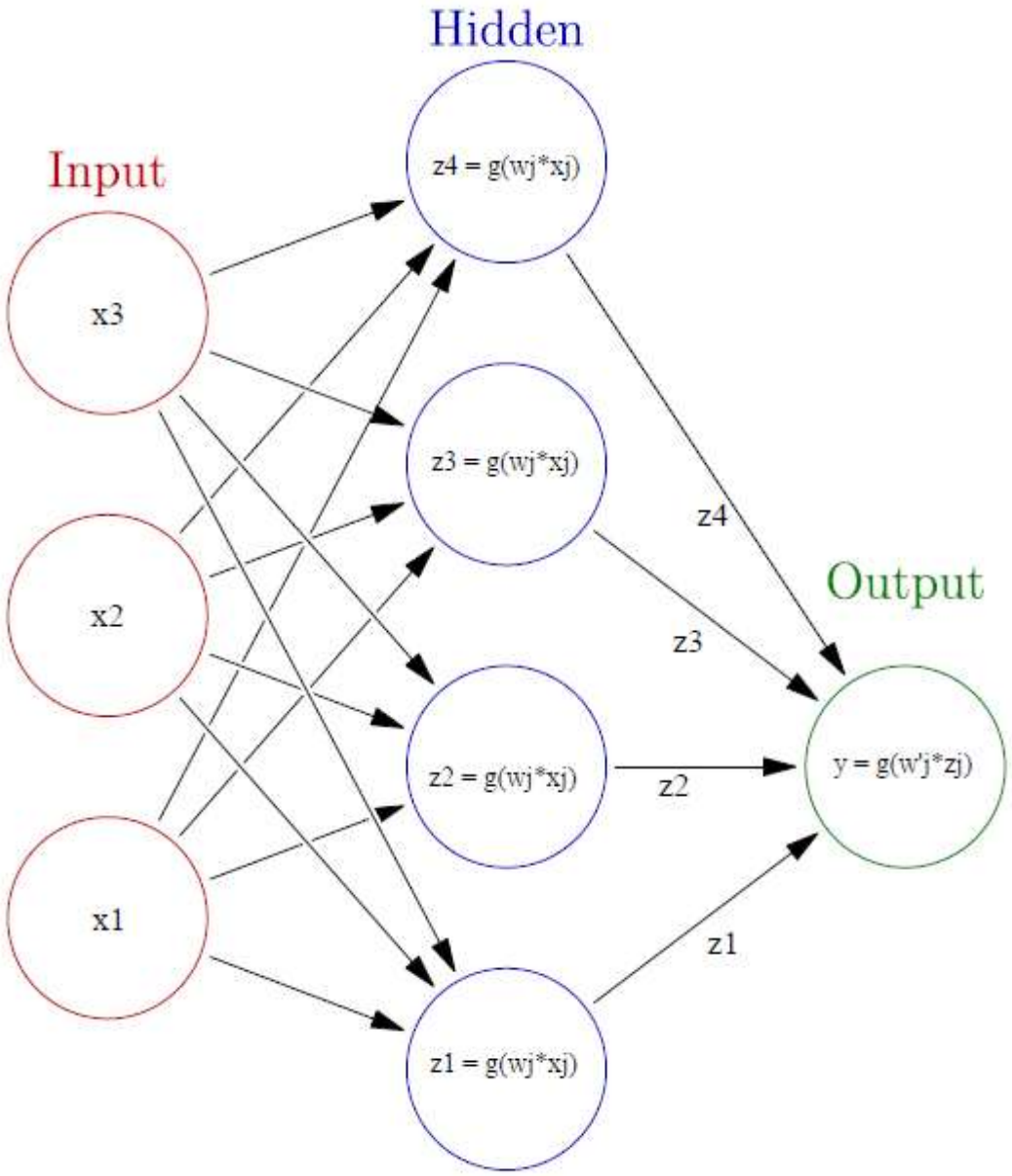
ALGORITMO DE GARSON.

- Intuitivamente, se puede entender que valores absolutos pequeños en las neuronas de la capa oculta para una variable o varias variables indica una escasa contribución de estas a la función de activación, y por lo tanto, al resultado final.
- De alguna manera, los pesos son las conexiones entre el problema (las variables de entrada) y la solución (la salida). Así la contribución relativa de las variables a la predicción depende de la magnitud y dirección (signo) de las conexiones (pesos).



ALGORITMO DE GARSON.

- Estos métodos requieren que la optimización de la red alcance un mínimo de alta calidad, ya que en caso contrario (mínimo local) podría generar un conjunto de pesos que produjesen una interpretación errónea de la importancia relativa de las variables.

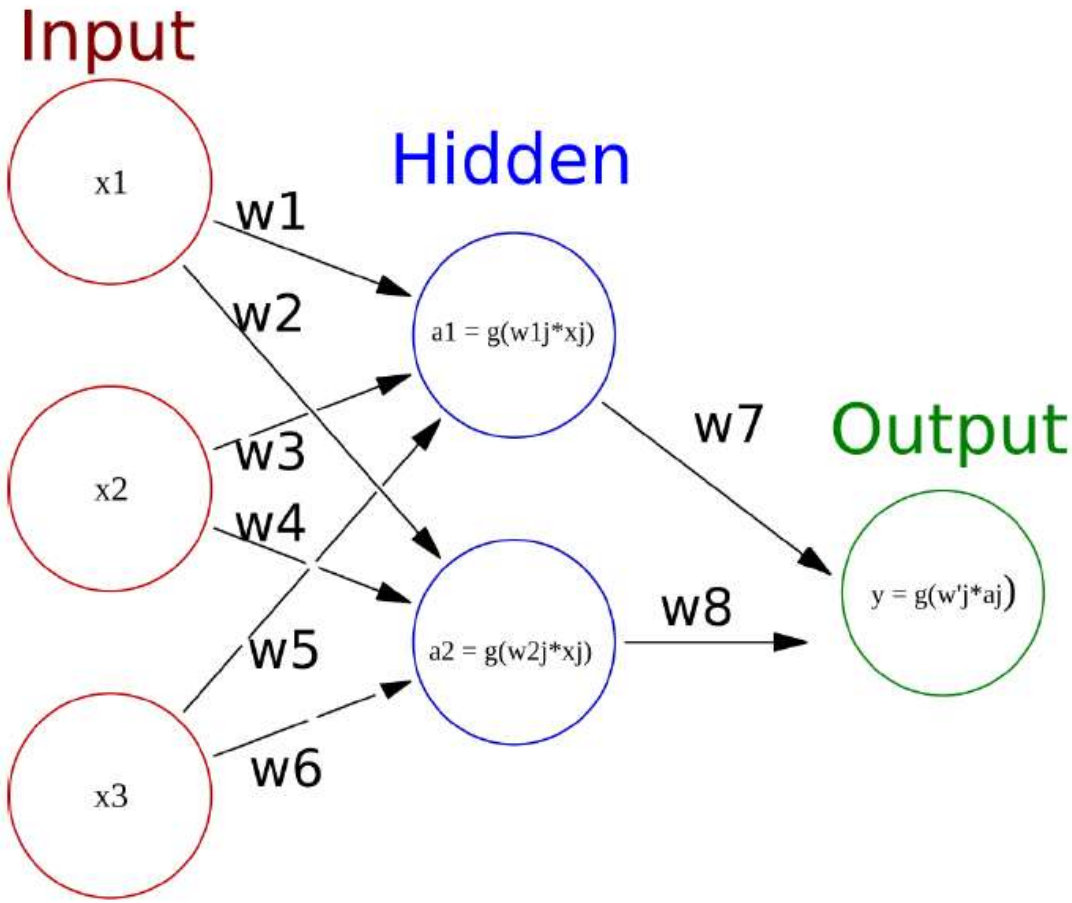


ALGORITMO DE GARSON.

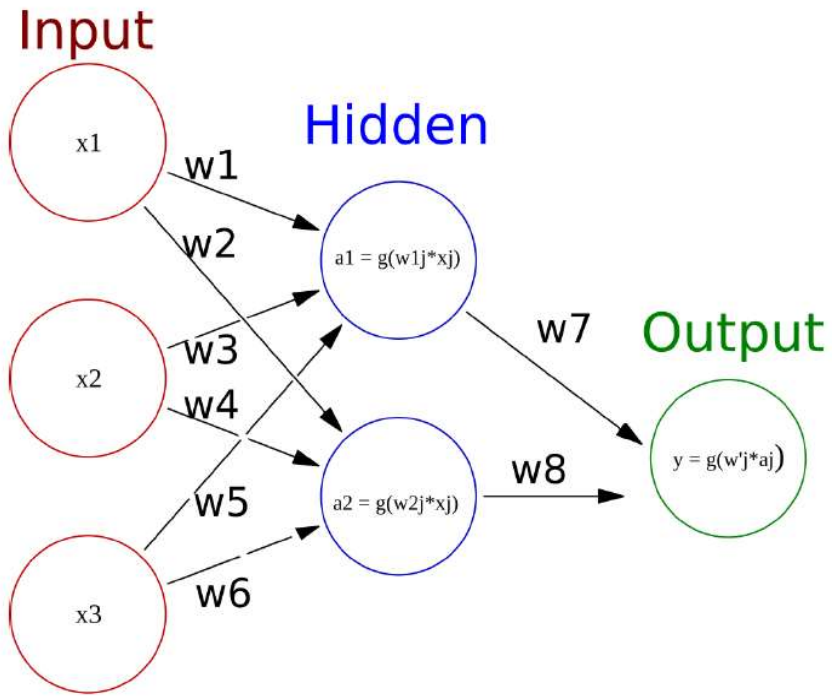
- La importancia relativa (*RI*) de la variable x_i se obtiene a través de la ecuación:

$$RI(x_i) = \sum_{y=1}^{n} \frac{|w_{xy} \cdot w_{yz}|}{\sum_{x=1}^{m} |w_{xy} \cdot w_{yz}|}$$

- siendo *n* el número de neuronas de la capa oculta, *m* el número de variables de entrada, w_{xy} los pesos de las neuronas de la capa oculta, y w_{yz} los pesos de las neuronas de la capa de salida.



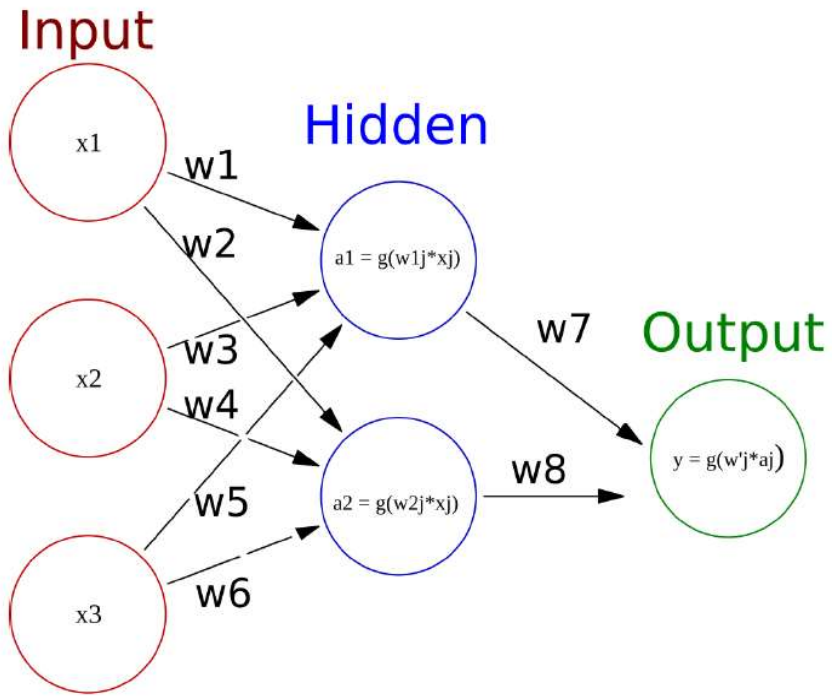
ALGORITMO DE GARSON.



- Así para la primera variable x_1 el resultado sería:

$$RI(x_1) = \frac{|w_1 \cdot w_7|}{|w_1 \cdot w_7| + |w_3 \cdot w_7| + |w_5 \cdot w_7|} + \frac{|w_2 \cdot w_8|}{|w_2 \cdot w_8| + |w_4 \cdot w_8| + |w_6 \cdot w_8|}$$

ALGORITMO DE GARSON.



- Para la variable x_2 el resultado sería:

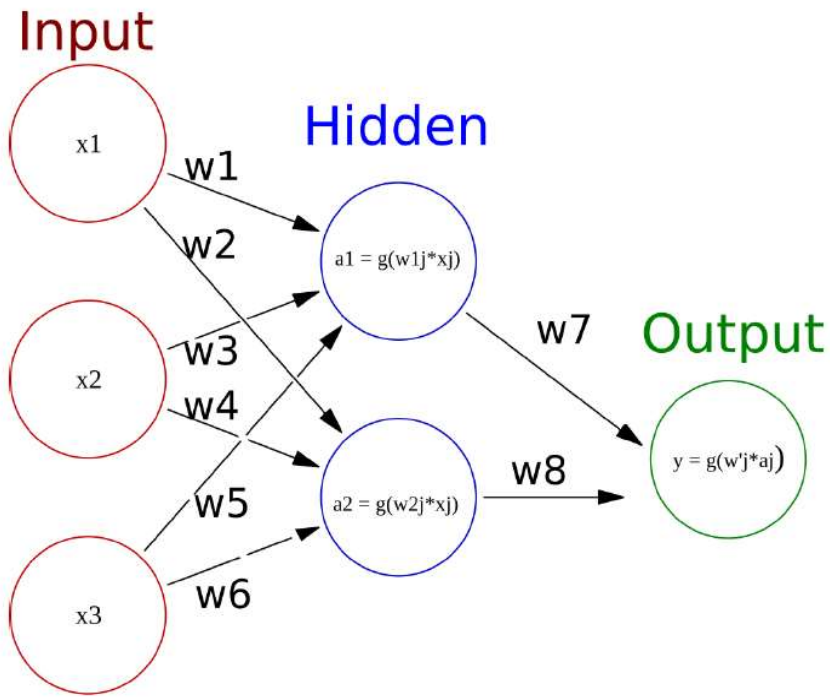
$$RI(x_2) = \frac{|w_3 \cdot w_7|}{|w_1 \cdot w_7| + |w_3 \cdot w_7| + |w_5 \cdot w_7|} + \frac{|w_4 \cdot w_8|}{|w_2 \cdot w_8| + |w_4 \cdot w_8| + |w_6 \cdot w_8|}$$

PESOS CONECTADOS.

- El algoritmo de pesos conectados (Olden) se muestra como una versión no normalizada del algoritmo de Garson:

$$RI(x_i) = \sum_{y=1}^{n} w_{xy} \cdot w_{yz}$$

$$RI(x_1) = w_1 \cdot w_7 + w_2 \cdot w_8$$



GARSON FRENTE A OLDEN

Garson:

- Valores altos de RI, implican variables importantes en el problema.
- Valores cercanos a cero, implican variables no importantes.

Olden.

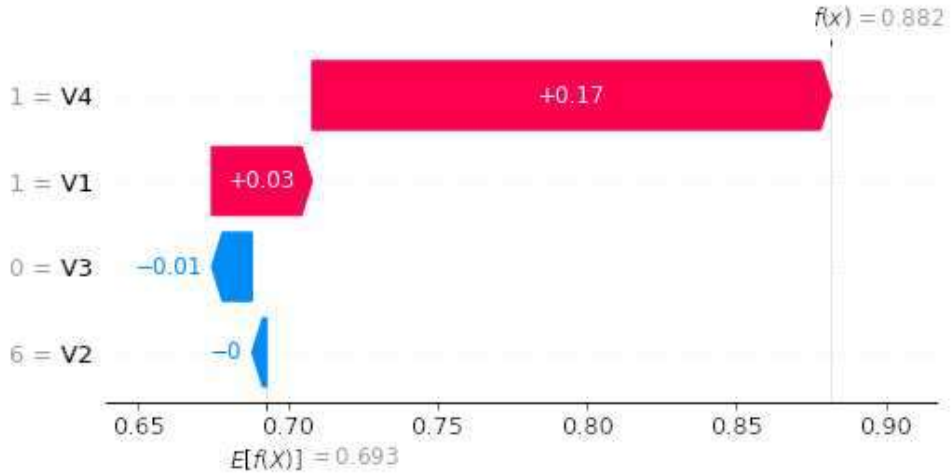
- Valores cercanos a cero, implican variables no importantes.
- En un problema de clasificación, valores altos de RI, implican variables importantes la clase con etiqueta +1; y valores muy pequeños implican variables importantes para la clase con etiqueta -1 ó 0.

GLOBAL FRENTE A INDIVIDUAL.

Global.

- Los algoritmos de Garson y Olden ofrecen explicaciones globales sobre la importancia de las variables.
- No permite interrogar sobre una instancia en particular.

Individual.



Publicación

- Lundberg, Scott, Su-In Lee. *A unified approach to interpreting model predictions*. (2017)

RESUMEN

- Se han visto algunos algoritmos sencillos que pueden proporcionar información acerca de las variables más importantes en redes neuronales densas.
- Se desmitifica la idea de las redes neuronales como modelos de cajas negras.

Código:

- Código ejemplo de red neuronal en problema de regresión con aplicación del/los algoritmo/s de Garson y Olden:

MLP Regressor Add2 XAI.ipynb.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

REDES NEURONALES CONVOLUCIONALES.

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDOS.

- Introducción
- CNN
- Resumen

REDES NEURONALES CONVOLUCIONALES

- Una de la debilidades de las redes completamente conectadas es su incapacidad para explotar la correlación espacial de la intensidad de los píxeles de las imágenes.
- El producto de los pesos w_{ij} y las variables independientes que es el argumento de la función no lineal $f(\sum_{i=0}^{D} w_{ji} x_i)$ ivo:

- Cualquier reorganización de las variables de entradas y sus pesos debe dar el mismo resultados, luego este resultado no depende si los píxeles son adyacentes o no.

REDES NEURONALES CONVOLUCIONALES

- Las redes neuronales convolutivas (CNN) son redes especializadas en el procesamiento de datos con topología de rejilla, por ejemplo imágenes (2D, 3D), sonidos (2D), series temporales (1D), y vídeos (4D).
- Como su nombre indica, las CNN emplean la operación de convolución en vez de la multiplicación de matrices en algunas de las capas de la red.

REDES NEURONALES CONVOLUCIONALES

- Las CNNs tienen un mejor rendimiento cuando los datos de entrada tienen alguna estructura intrínseca. Por ejemplo, en imágenes, los píxeles cercanos tienen a tener valores cercanos.
- Por el contrario, en datos tabulados y extraídos de bases de datos, los valores de las columnas cercanas no tienen relación más que el orden de extracción.

CNN

- En su forma más general, la operación de convolución, es una operación sobre dos funciones de argumentos reales (ecuación 1).

$$s(t) = \int x(a)w(t-a)da \tag{1}$$

- Cuando las variables son discretas, la operación de convolución puede escribirse como la ecuación 2. En esta ecuación, el tiempo *t* toma valores discretos y se asume

$$s(t) = (x * w)(t) = \sum_{a=\infty}^{\infty} x(a)w(t-a) \tag{2}$$

CNN

- En minería de datos es usual que la entrada sean objetos bidimensionales. Por esta razón, la ecuación 2 se reescribe como la ecuación 3. Gracias a la propiedad conmutativa de la convolución nos permite voltear los índices de la ecuación 3 produciendo la ecuación 5.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m,j-n) \tag{3}$$

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m,j-n)K(m,n) \tag{4}$$

EJEMPLO DE KERNEL CONVOLUTIVO DE TAMAÑO 3×3 CON ZANCADA 1

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n) \qquad (5)$$

1	5	1	1	3	3
1	4	2	3	2	1
2	5	3	2	1	0
7	7	1	4	4	5
1	5	2	2	0	2
2	1	4	2	1	0

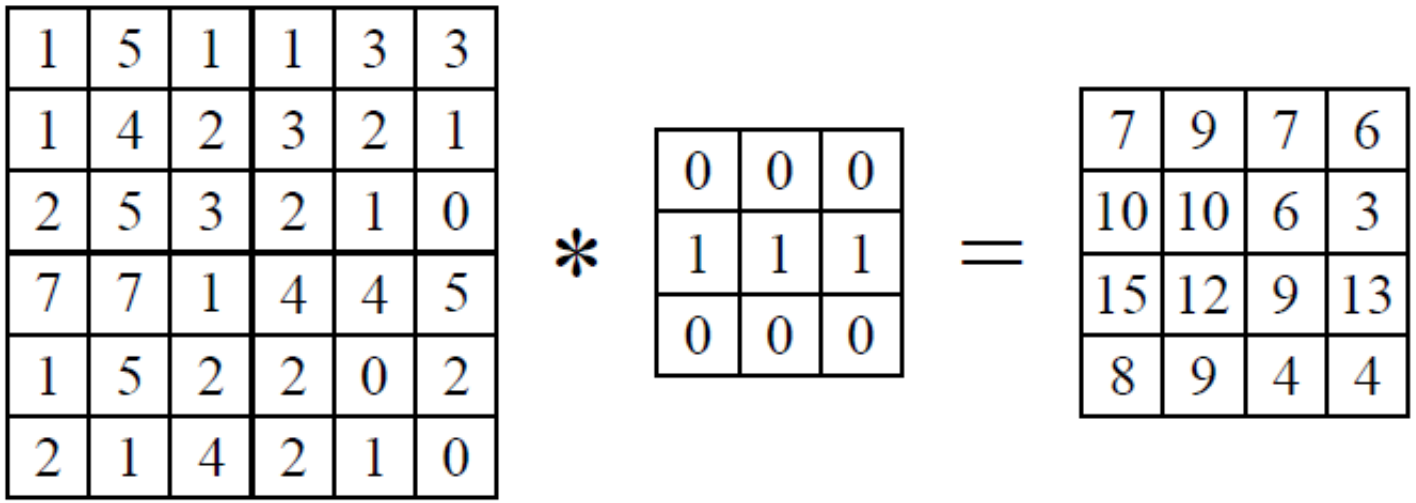
$*$

0	1	0
0	1	0
0	1	0

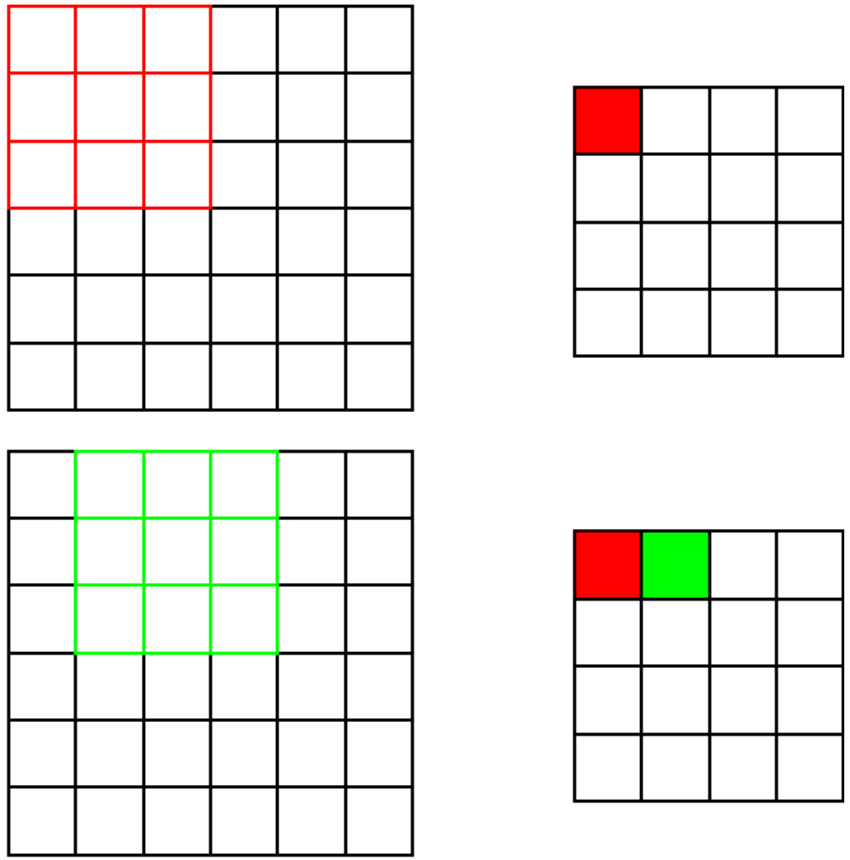
$=$

14	6	6	6
16	6	9	7
17	6	8	5
13	7	8	5

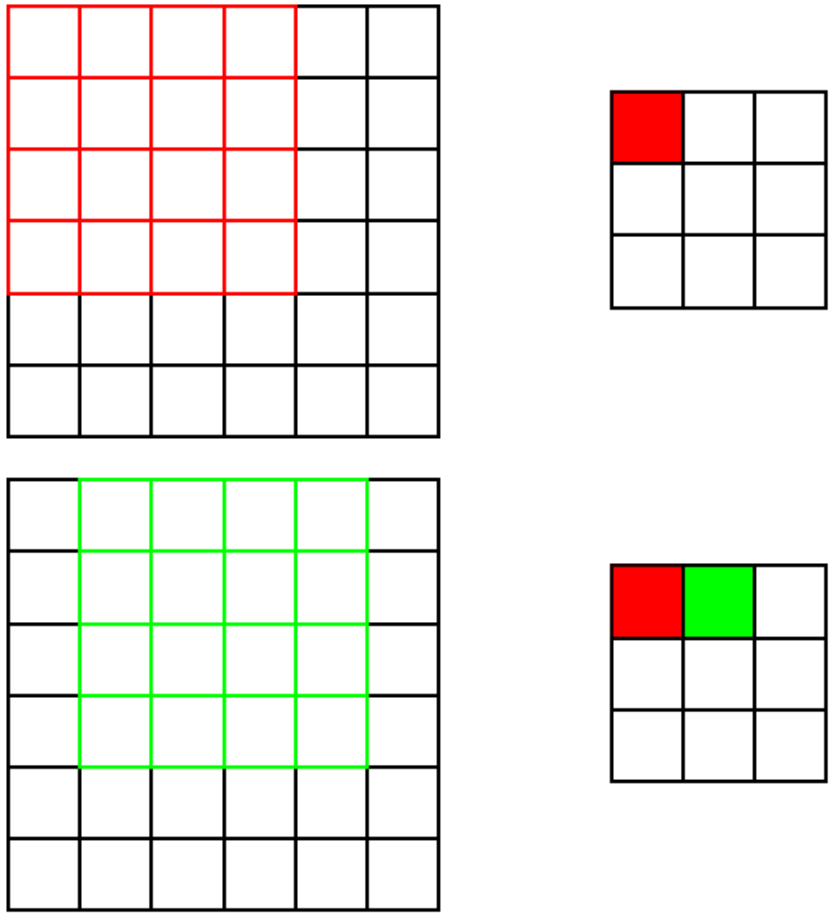
EJEMPLO DE KERNEL CONVOLUTIVO DE TAMAÑO 3 × 3 CON ZANCADA 1



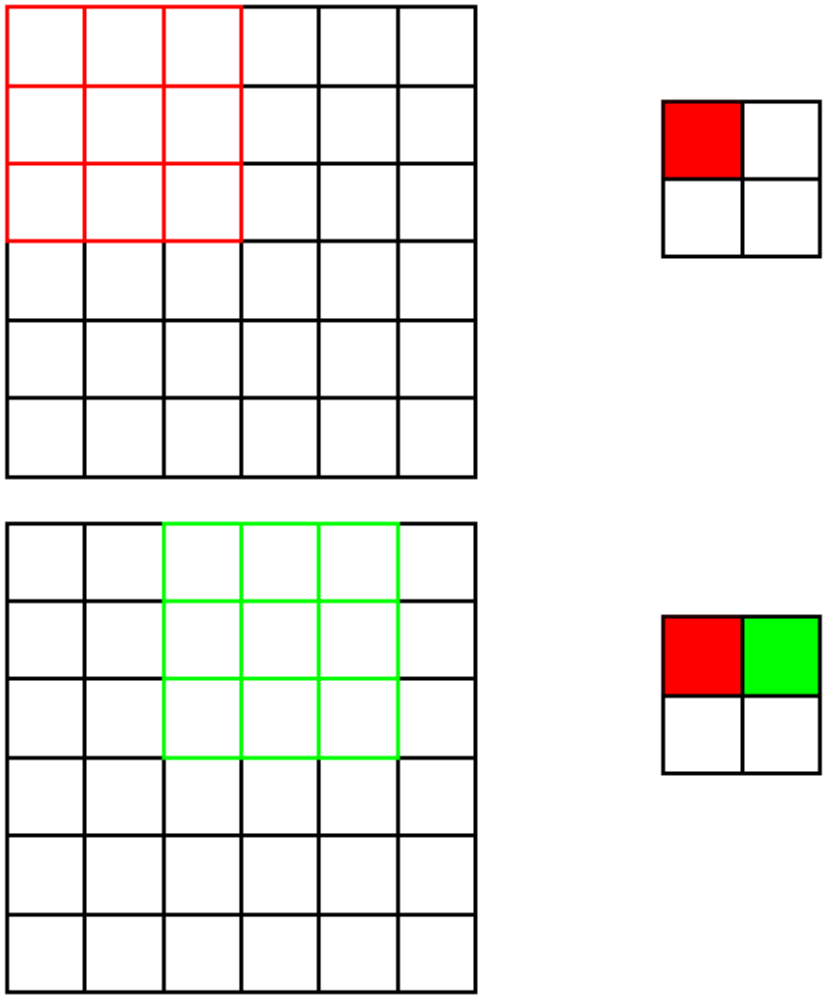
EJEMPLO DE KERNEL CONVOLUTIVO DE TAMAÑO 3×3 CON ZANCADA 1



EJEMPLO DE KERNEL CONVOLUTIVO DE TAMAÑO 4×4 CON ZANCADA 1

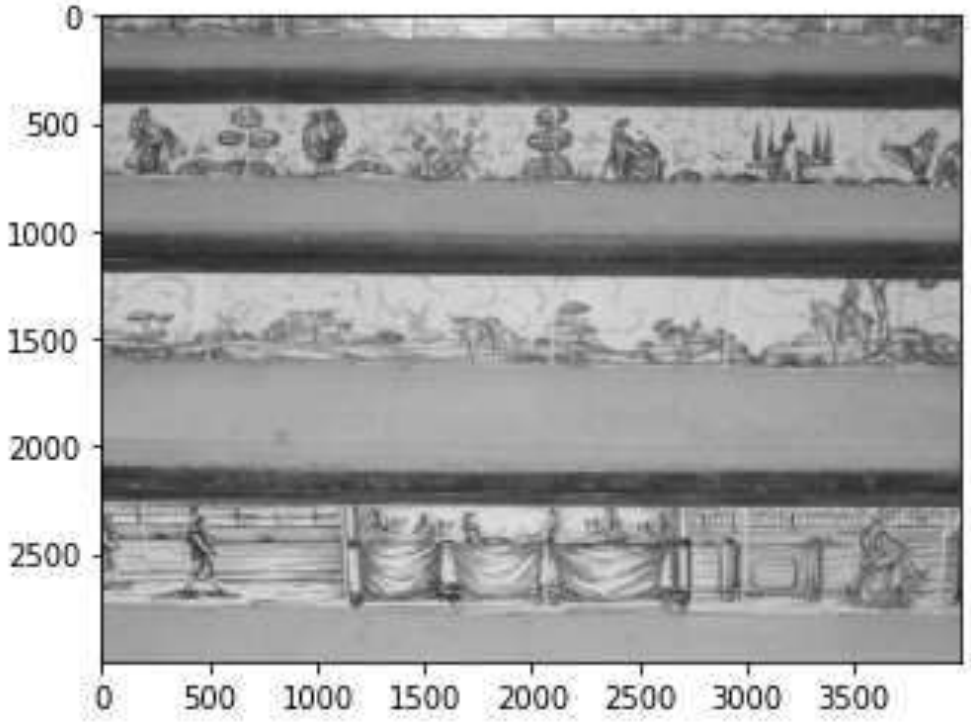


EJEMPLO DE KERNEL CONVOLUTIVO DE TAMAÑO 3×3 CON ZANCADA 2



FILTRO SOBEL

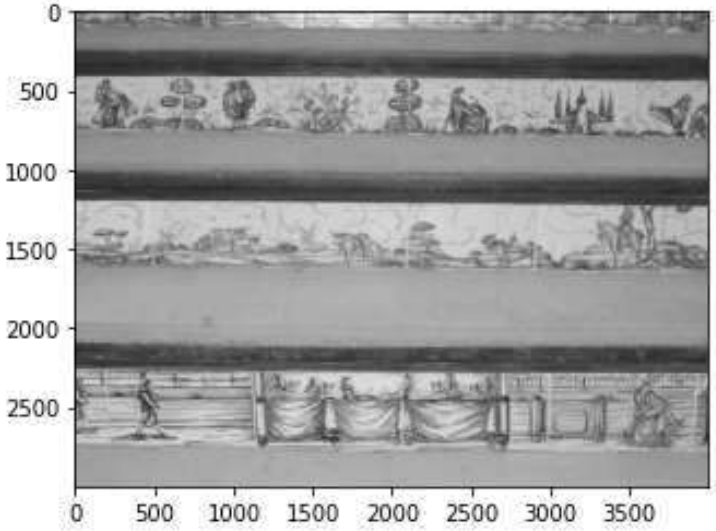
- Un ejemplo de filtro convolucional es el filtro de Sobel. Este permite destacar los líneas verticales (ecuación 7) o las líneas horizontales (ecuación 6).



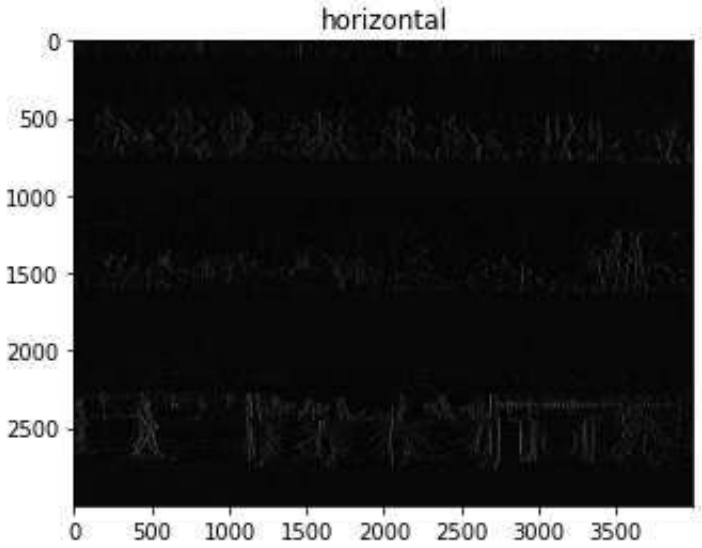
$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad (6)$$

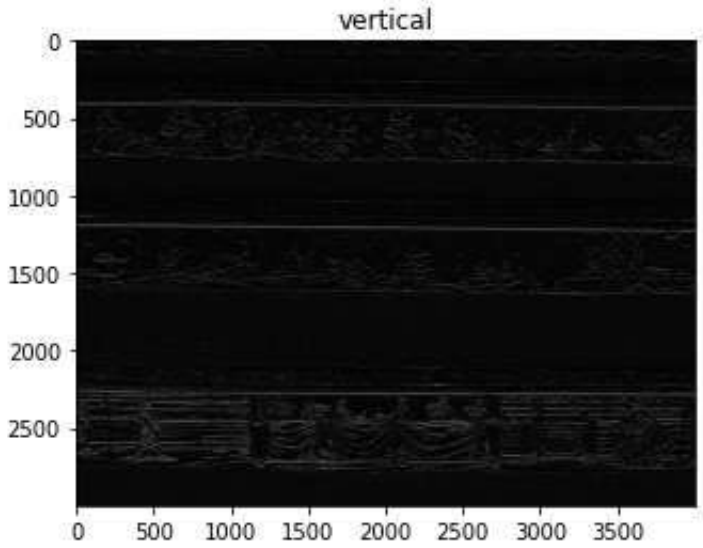
$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (7)$$

FILTRO SOBEL



$$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$





$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

REDES NEURONALES CONVOLUCIONALES

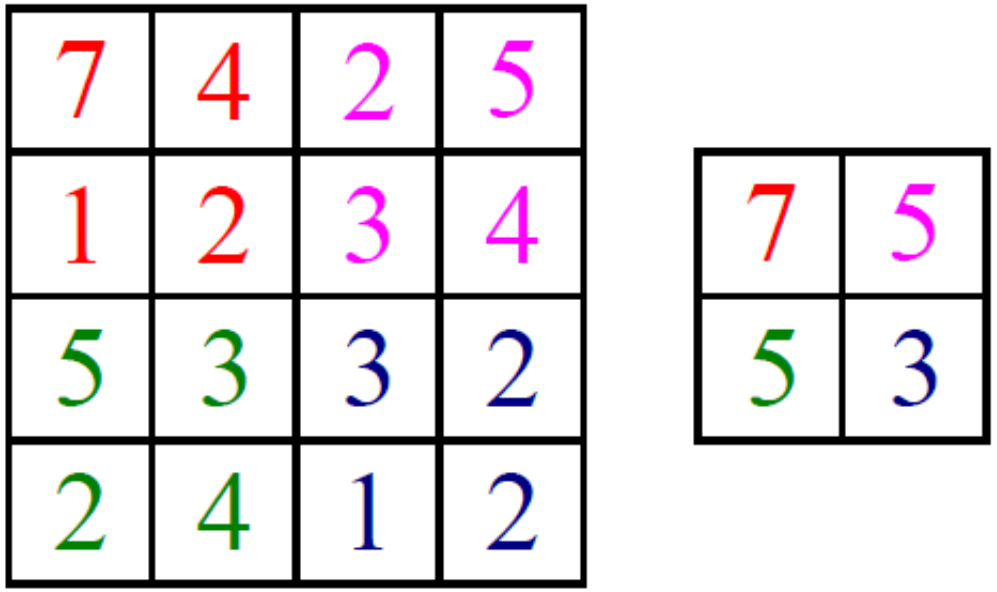
Kernels Pequeños

- Tienen un campo receptivo más pequeño.
- Extraen características locales y pueden no tener una vista general de los datos de entrada.
- Producen un volumen de características mayor.

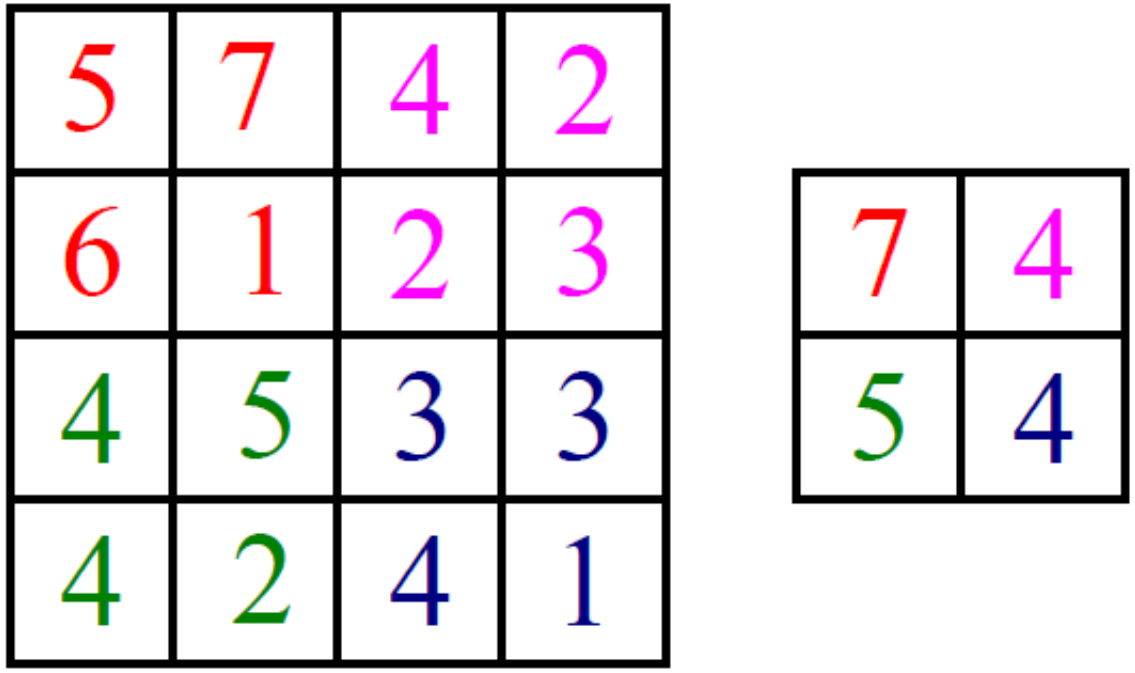
Kernels Grandes

- Capturan características más genéricas.
- Miran más píxeles a la vez y por lo tanto tienen un campo receptivo mayor.
- Se reduce la cantidad de información extraída.
- En el extremo de un tamaño de kernel igual al tamaño de la imagen, se asemeja a una red completamente conectada.
- Computacionalmente más intensivo

EJEMPLO DE MAXPOOLING DE TAMAÑO 2 × 2 Y ZANCADA 2



EJEMPLO DE MAXPOOLING DE TAMAÑO 2 × 2 TRAS UN DESPLAZAMIENTO



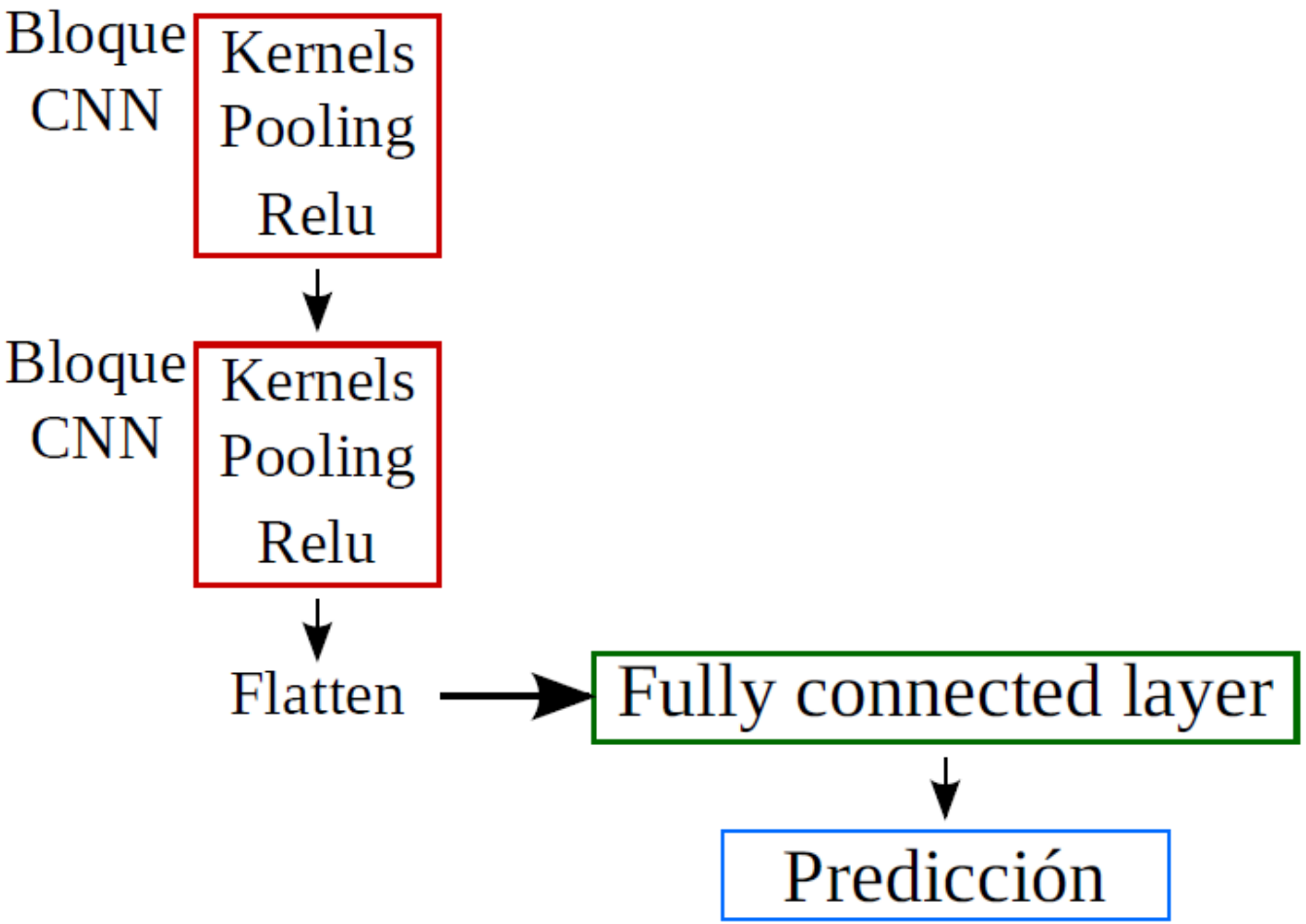
EJEMPLO DE RELU

-7	9	7	-6
10	-10	6	3
15	12	-9	13
8	9	-4	4

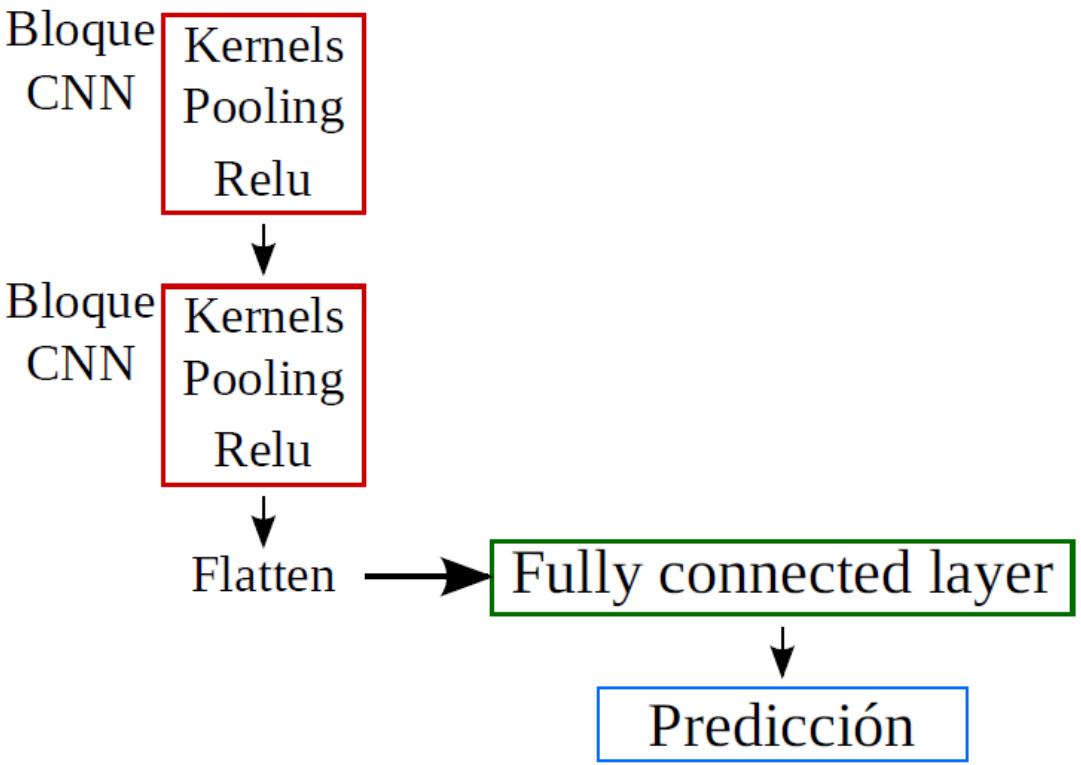
$$* \text{Relu} =$$

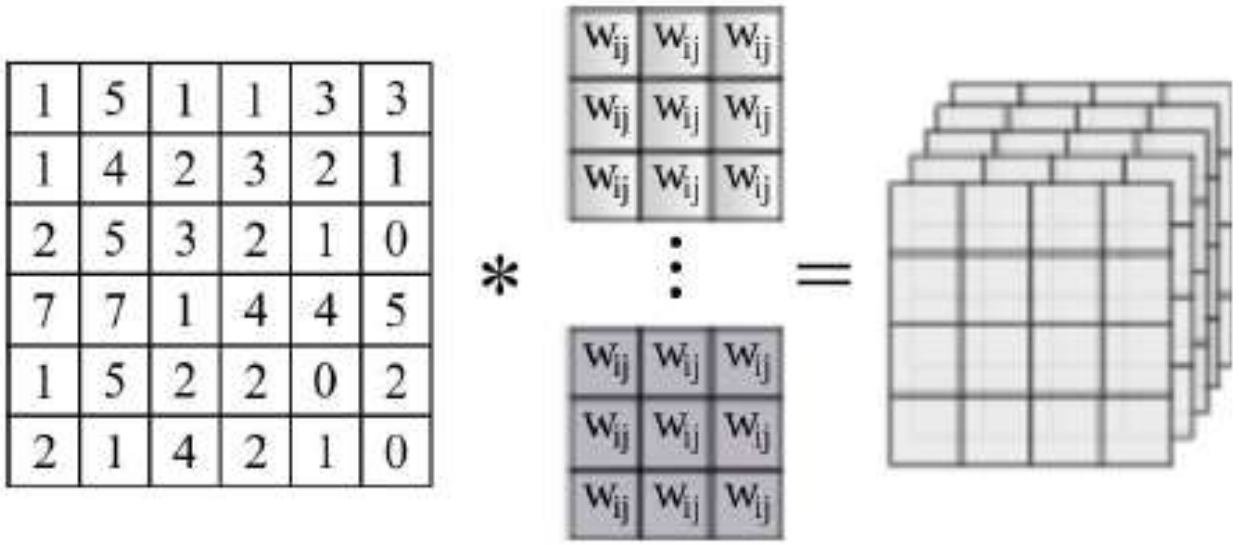
0	9	7	0
10	0	6	3
15	12	0	13
8	9	0	4

ESQUEMA.

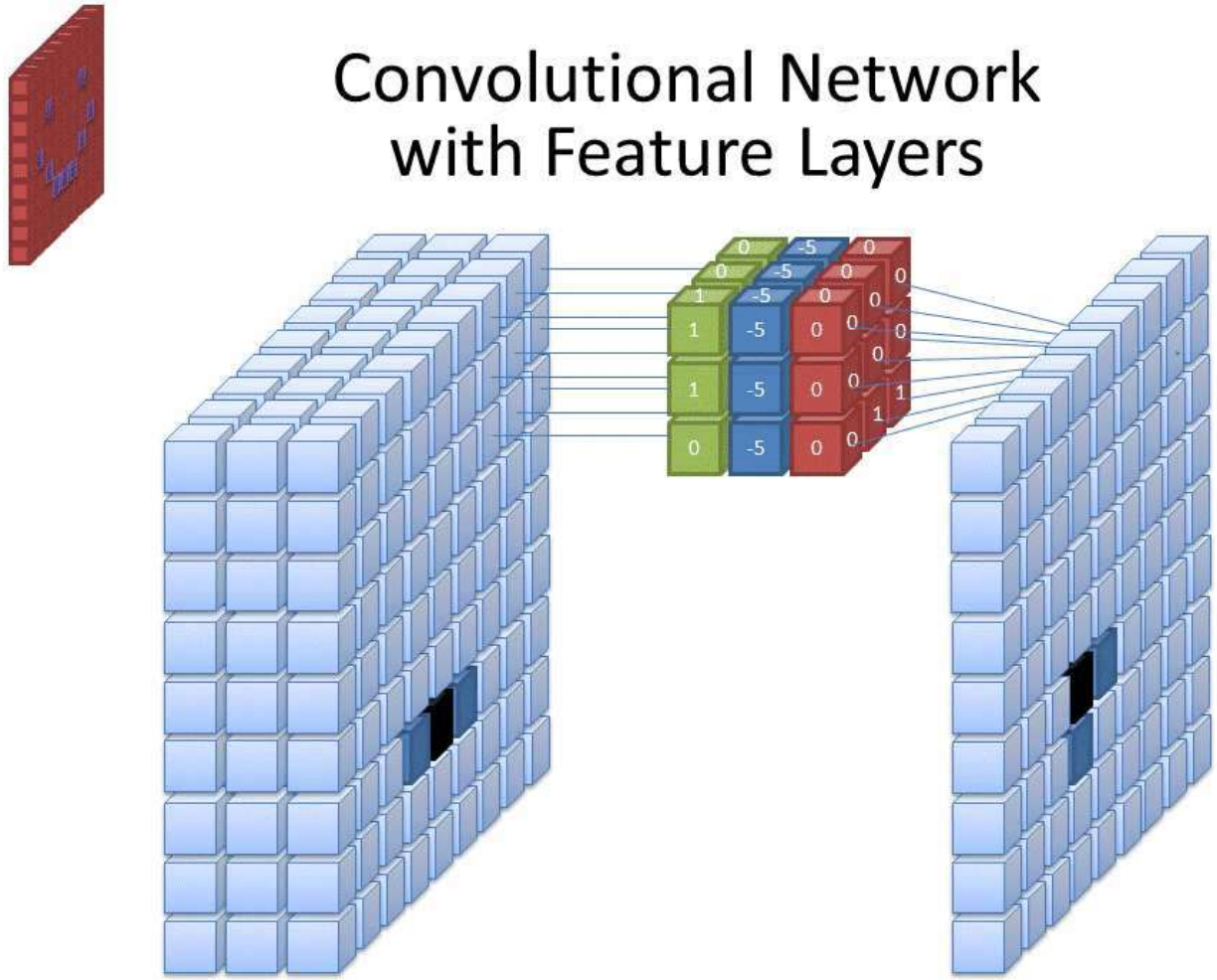

Bloque CNN
Kernels
Pooling
Relu
Bloque CNN
Kernels
Pooling
Relu
Flatten
Fully connected layer
Predicción

ESQUEMA Y APRENDIZAJE.

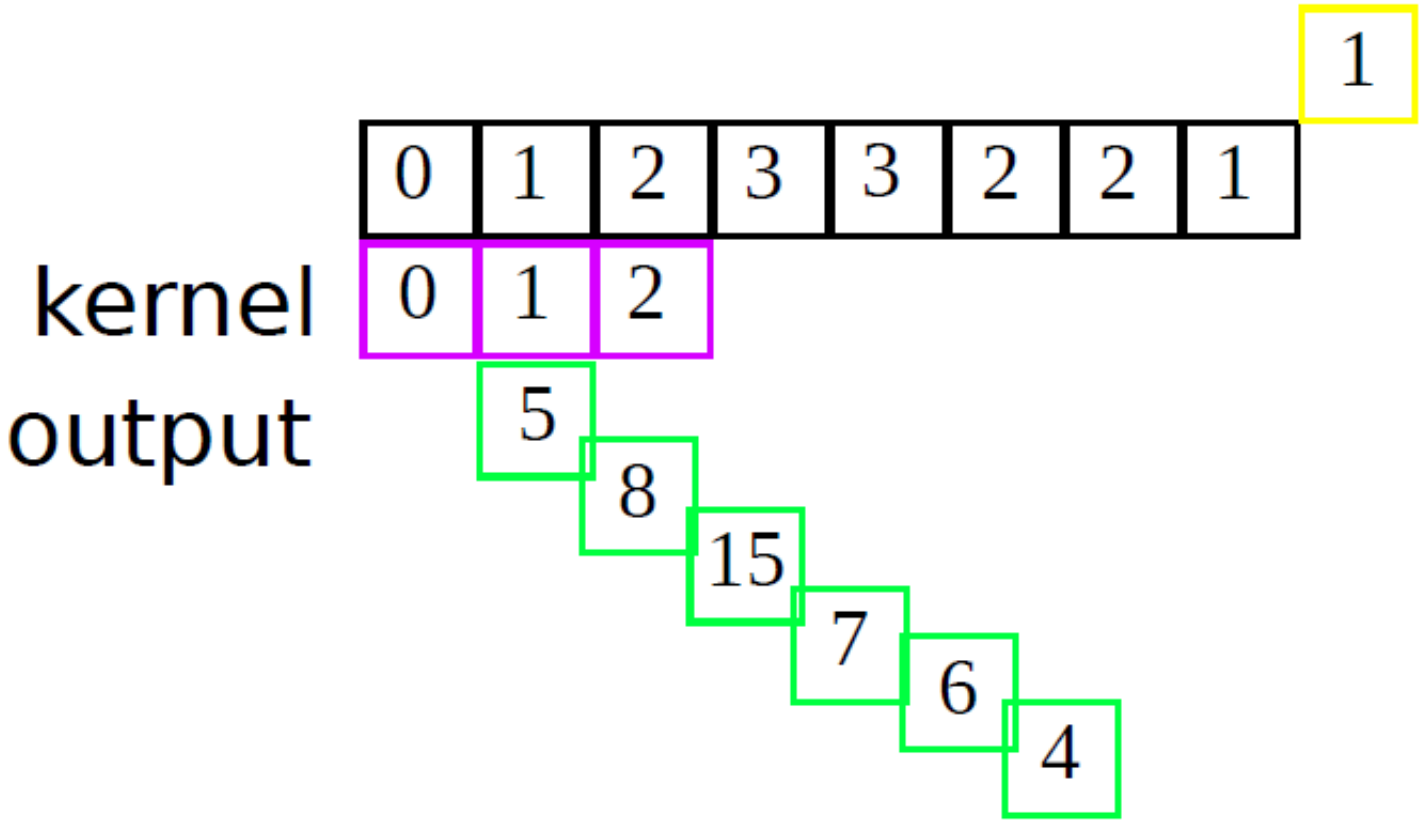




CNN EN IMAGEN COLOR.



CNN EN SERIES TEMPORALES



RESUMEN

- Fundamentos de las CNN.
- Las CNN son una poderosa herramienta para tratar datos con correlación espacial, por ejemplo imágenes, especialmente para clasificar imágenes.

Código:

- Código ejemplo de clasificación binaria:

Clasificador_CNN_PerrosGatos.ipynb.

- Código ejemplo de CNN para el pronóstico en series temporales:

CNN_TS_ContMadrid.ipynb.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

CRITERIOS DE CALIDAD EN CLASIFICACIÓN.

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDOS.

- Clasificación
- Resumen

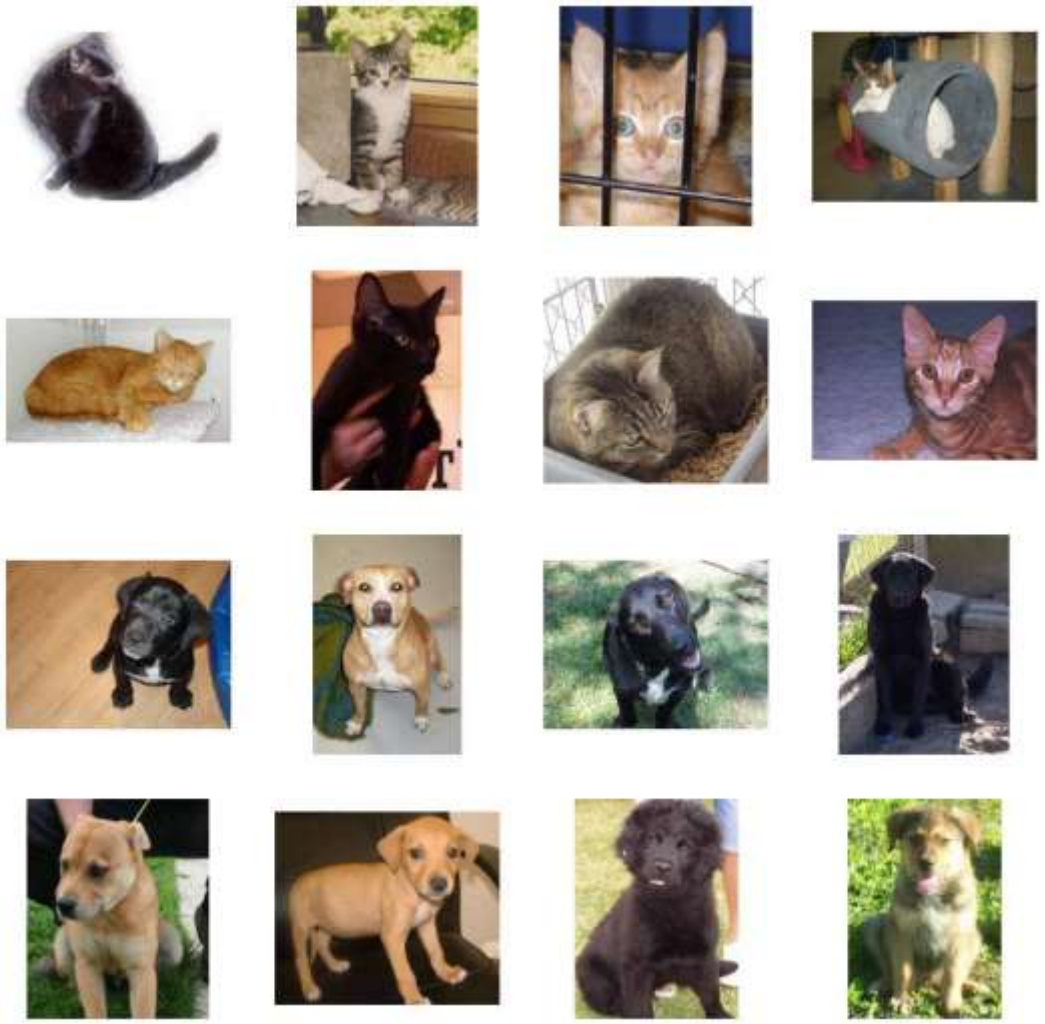
REDES NEURONALES CONVOLUCIONALES.

- Existen numerosos criterios de calidad para problemas de clasificación con redes neuronales.
- Son diferentes para problemas de regresión y para problemas de clasificación.
- Muchos son comunes a otros algoritmos de aprendizaje automático usados en problemas de clasificación.

PROPUESTA 60-20-20

- 60% de los datos para el conjunto de entrenamiento. Sirve para optimizar los parámetros de la red neuronal.
- 20% de los datos para el conjunto de test, para el ajuste de los parámetros del algoritmo. Sirve para optimizar los hiperparámetros de la red neuronal.
- 20% de los datos para el conjunto de validación. ¡Este es el valor del error generalizado!
- Hay autores que utilizan otra nomenclatura pero se mantiene el sentido de la división de los datos.

ESQUEMA



- ¡La elección del positivo-negativo es arbitraria en la mayoría de los casos!

MATRIZ DE CONFUSIÓN I

- La matriz de confusión resume cuatro casos :
  - Verdadero positivo: caso positivo etiquetado como positivo.
  - Verdadero negativo: caso negativo etiquetado como negativo.
  - Falso positivo: caso negativo etiquetado como positivo.
  - Falso negativo: caso positivo etiquetado como negativo.
- Partiendo de estos casos, se puede hacer una estadística más elaborada.

	Clasificado Verdadero	Clasificado Falso
Es Verdadero	verdaderos positivos (TP)	falsos negativos (FN)
Es Falso	falsos positivos (FP)	verdaderos negativos (TN)

MATRIZ DE CONFUSIÓN II

- Se denomina *sensitivity* o *recall* o *true positive rate* al cociente entre los verdaderos positivos y el total de positivos.
- También se denomina *recognition*.

	Clasificado Verdadero	Clasificado Falso	
Es Verdadero	TP	FN	$sensitivity = \frac{TP}{TP+FN}$
Es Falso	FP	TN	

MATRIZ DE CONFUSIÓN III

- Se denomina *specificity* o *true negative rate* al cociente entre los verdaderos negativos y el total de negativos.

	Clasificado Verdadero	Clasificado Falso	
Es Verdadero	TP	FN	
Es Falso	FP	TN	$specificity = \frac{TN}{TN+FP}$

MATRIZ DE CONFUSIÓN IV

- Se denomina *precision* o *positive predictive value* al cociente entre los verdaderos positivos y la suma de los verdaderos positivos y los falsos positivos.

	Clasificado Verdadero	Clasificado Falso	
Es Verdadero	TP	FN	
Es Falso	FP	TN	
	$precision = \frac{TP}{TP+FP}$		

MATRIZ DE CONFUSIÓN V

- Se denomina *negative predictive value* al cociente entre los negativos verdaderos y la suma de los negativos verdaderos y los falsos negativos.

	Clasificado Verdadero	Clasificado Falso	
Es Verdadero	TP	FN	
Es Falso	FP	TN	
		$NPV = \frac{TN}{TN+FN}$	

MATRIZ DE CONFUSIÓN VI.

- La matriz de confusión ideal para M positivos y N negativos es:

	Clasificado Verdadero	Clasificado Falso	
Es Verdadero	M	0	
Es Falso	0	N	

MATRIZ DE CONFUSIÓN VII.

- 1000 positivos en la muestra.
- 100 negativos en la muestra.

	Clasificado Verdadero	Clasificado Falso		
V	995	5	$\frac{995}{995+5} = 99.50\%$	*sensitivity*
F	20	80	$\frac{80}{80+20} = 80.00\%$	*specificity*
	$\frac{995}{995+20} = 98.03\%$ *precision*			

ACCURACY I

- Una medida adicional de la precisión es el concepto de *accuracy*:

$$accuracy = sensitivity \cdot \frac{pos}{pos + neg} + specificity \cdot \frac{neg}{pos + neg}$$

- Aplicado al caso anterior:

$$accuracy = 0.995 \cdot \frac{1000}{1100} + 0.800 \cdot \frac{100}{1100} = 0.977$$

MATTHEWS CORRELATION COEFFICIENT

- El coeficiente de correlación de Matthews (MCC) es aplicable a la matriz de confusión incluso con datos desbalanceados.
- MCC es aplicable a clasificaciones binarias.
- Devuelve un valor entre -1 y 1, siendo 1 el caso de predicción perfecta, -1 el totalmente incorrecta y 0 predicción aleatoria.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

MATRIZ DE CONFUSIÓN VII

Debilidades

- No existe un consenso sobre la división de un conjunto único de datos, qué porcentaje de los datos se asigna al conjunto de validación.
- Pierde utilidad con datos desbalanceados.
- Se vuelve confusa para muchas clases.

MATRIZ DE CONFUSIÓN VIII

- Para M clases, la matriz de confusión tendría M×M dimensiones.
- La diagonal serían los verdaderos, y el resto las clasificaciones erróneas.

RESUMEN

- Los criterios de calidad dependen de la naturaleza del problema, y son críticos para mostrar la calidad del trabajo realizado.
- Debe presentarse sobre el conjunto de datos de test o de validación.

Código:

- Esta lección no incluye un código específico de ejemplo.
- En el código:

Clasificador_CNN_PerrosGatos.ipynb

se incluyen los criterios específicos de los problemas de clasificación.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

ONE-HOT-ENCODING.

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDOS.

- One-Hot-Encoding.
- Resumen.

ONE-HOT-ENCODING.

- En los problemas de clasificación es recomendable transformar las etiquetas de las clases en valores numéricos, y estos a su vez en la representación *one-hot-encoding*.
- Las redes neuronales no pueden operar con etiquetas de texto: (rojo, verde, azul, amarillo, marrón), (mujer, hombre), (sano, neumonía, covid), (comprar acciones, vender acciones), (árbol, avión, batidora, botella, camión, coche, elefante, toro, tren, vaca), etc.

ONE-HOT-ENCODING.

- Las redes neuronales no pueden operar con etiquetas de texto, necesitan transformarlas en numéricas.

(comprar acciones, vender acciones),	(0,1)
(sano, neumonia, covid)	(0,1,2)
(rojo, verde, azul, amarillo, marrón)	(0,1,2,3,4)

ONE-HOT-ENCODING.

(rojo, verde, azul, amarillo, marrón)	(0,1,2,3,4)

- ¿Esto significa que marrón (4) es mejor que azul (2)?
- ¿Esto significa que la media de marrón (4) y azul (2) es amarillo (3)?
- La solución es *one-hot-encoding* (1-of-K).

ONE-HOT-ENCODING.

- En one-hot-encoding las clases se organizan como un vector de igual dimensionalidad que el número de clases, señalándose cada clase con un 1.
- La no existencia se señala con un 0 y la existencia con un 1.

Clase	numérica	one-hot-encoding
comprar acciones	(0)	[1,0]
vender acciones	(1)	[0,1]

ONE-HOT-ENCODING.

- Esta codificación requiere que en la capa de salida el número de neuronas sea igual al número de clases, y que la función de activación de esta capa sea *softmax*.
- La intuición detrás de que el número de neuronas coincida con el número de clases es que cada neurona se activará para cada una de las clases.

Clase	numérica	one-hot-encoding
sano	(0)	[1,0,0]
neumonia	(1)	[0,1,0]
covid	(2)	[0,0,1]

ONE-HOT-ENCODING.

- La función *softmax* es una generalización de la función logística.
- Esta función tiene como entrada un vector de números reales y los normaliza como una distribución de probabilidad, es decir, todos ellos están comprendidos en el rango 0-1 y suma la unidad:

$$y_k(\mathrm{x}, \mathrm{w}) = \frac{exp(a_k(\mathrm{x},\mathrm{w}))}{\sum_j exp(a_k(\mathrm{x},\mathrm{w}))}$$

- La función *softmax* aplica la función exponencial a cada elemento z_i del vector z, y normaliza los valores dividiendo por la suma de todas las exponenciales.

ONE-HOT-ENCODING.

- También requiere que la función de error sea la función de entropía cruzada correspondiente al número de clases.
- Entropía cruzada binaria:

$$E(\mathrm{w}) = -\sum_{n=1}^{N} \left(t_n \, ln(y_n) + (1 - t_n) \, ln(1 - y_n)\right)$$

- Entropía cruzada multiclase:

$$E(\mathrm{w}) = -\sum_{n=1}^{N} \sum_{k=1}^{K} \left(t_{nk} \, ln(y_{nk}) + (1 - t_{nk}) \, ln(1 - y_{nk})\right)$$

siendo N el número de ejemplos, K el número de clases, y_{nk} la etiqueta real del ejemplo n-ésimo y t_{nk} su pronóstico.

RESUMEN

- *One-hot-encoding* es recomendado para problemas de clasificación.
- La capa de salida de la red debe adaptarse al número de clases en el problema.

Código:

- Esta lección no incluye un código específico de ejemplo.
- En el código:

Clasificador_CNN_PerrosGatos.ipynb

se incluye la codificación *OneHotEncoding*.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

TRANSFERENCIA DE APRENDIZAJE.

Transfer Learning

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

TRANSFER LEARNING.

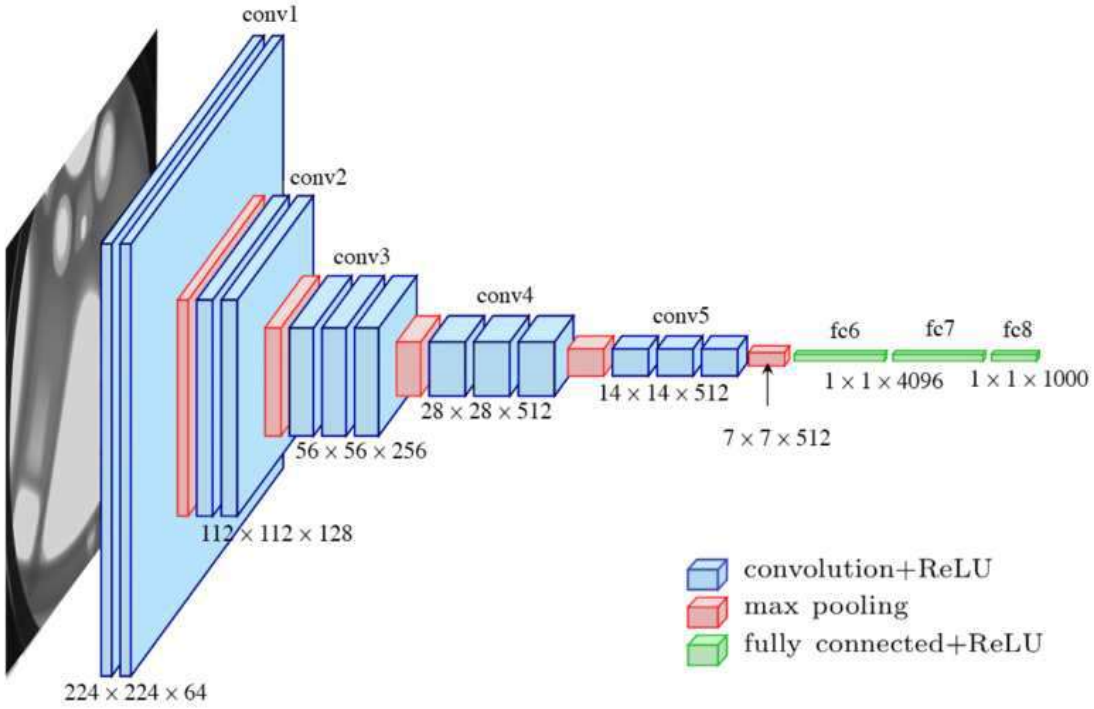
- En la comunidad de IA existen varias competiciones relevantes, por ejemplo en clasificación de fotografías.
- La competición ImageNet Large Scale Visual Recognition Challenge (ILSVRC) consta de 14 millones de imágenes que se expanden en 1000 categorías.
- Entrenar convolucionales modelos que obtengan un alto rendimiento estos desafíos es costoso y requiere un volumen de recursos no disponible para los estudiantes.

TRANSFER LEARNING.

- Los modelos entrenados ganadores de ILSVRC están libremente disponibles.
- ¿Cómo puede beneficiarnos una red neuronal entrenada sobre un catálogo de imágenes de barcos, camiones y otros objetos similares para clasificar imágenes de tejidos humanos sanos o enfermos?

TRANSFER LEARNING.

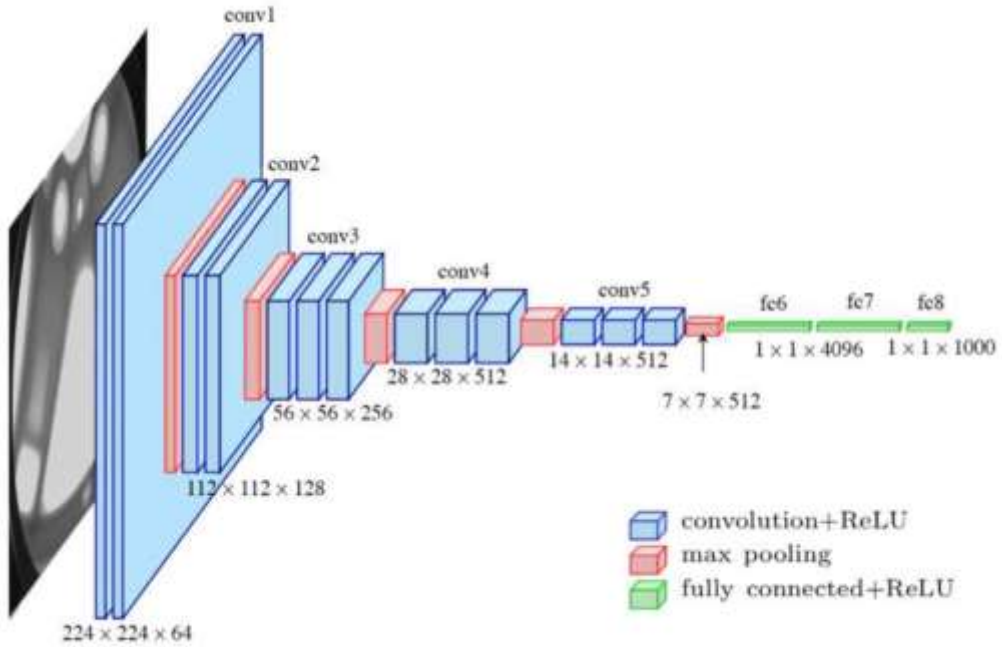
- ¿Cómo puede beneficiarnos una red neuronal entrenada sobre un catálogo de imágenes de barcos, camiones y otros objetos similares para clasificar imágenes de tejidos humanos sanos o enfermos?
- Las primeras capas de estos modelos reconocen formas básicas, y las capas superiores, formas específicas de las imágenes.
- Por lo tanto, se pueden congelar los parámetros de las capas iniciales, marcándolas como no entrenables, y solamente dejar como entrenables las capas superiores.
- Incluso se pueden añadir nuevas capas.



TRANSFER LEARNING.

- ¿Cómo puede beneficiarnos una red neuronal entrenada sobre un catálogo de imágenes de barcos, camiones y otros objetos similares para clasificar imágenes de tejidos humanos sanos o enfermos?

- Las primeras capas de estos modelos reconocen formas básicas, y las capas superiores, formas específicas del problema de clasificación.
- Por lo tanto, se pueden congelar los parámetros de las capas iniciales, marcándolas como no entrenables, y solamente dejar como entrenables las capas superiores.
- Incluso se pueden añadir nuevas capas.



BENEFICIOS

- Red de alta calidad a bajo coste: entrenamos miles de parámetros en una red de millones
- Reducción del tiempo de entrenamiento.
- Si el volumen de datos es pequeño, no gastamos datos en entrenar formas básica.
- Anadir nuevos datos a un modelo entrenado.

RESUMEN

- Aprovechamiento de la transferencia de aprendizaje para mejorar nuestros propios clasificadores.

- Código:
  - En el códigoTransferLearning VGG16 CatDog.ipynb se incluye un ejemplo de transferencia de aprendizaje

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

CLASS ACTIVACTION MAP.

Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDOS.

- CAM.
- Limitaciones.
- Resumen.

¿POR QUÉ SOY UN ELEFANTE?

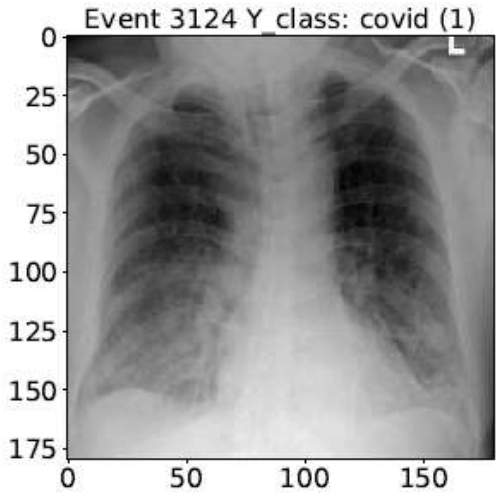
¿POR QUÉ SOY UN ELEFANTE?

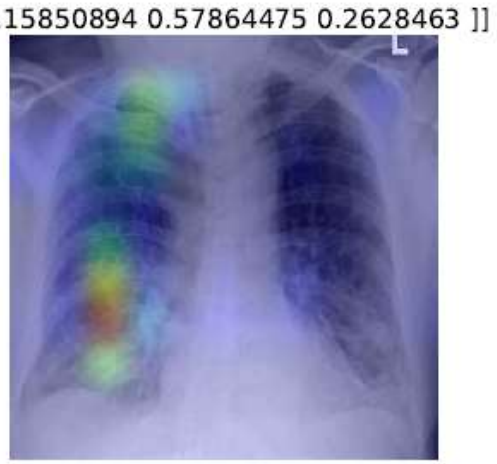
Publication

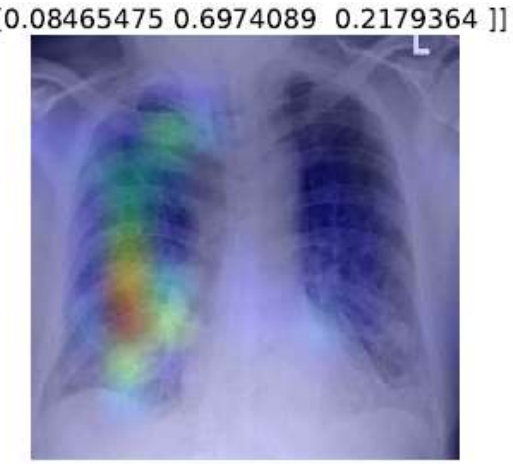
- Zhou et al., *Learning Deep Features for Discriminative Localization* (2015)

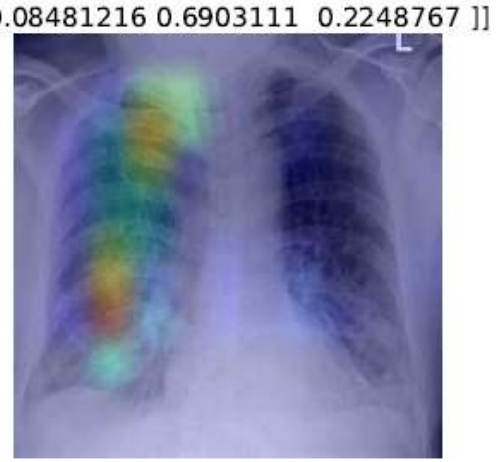
¿DÓNDE ACTIVA ESTA RADIOGRAFÍA LA ETIQUETA COVID-19?

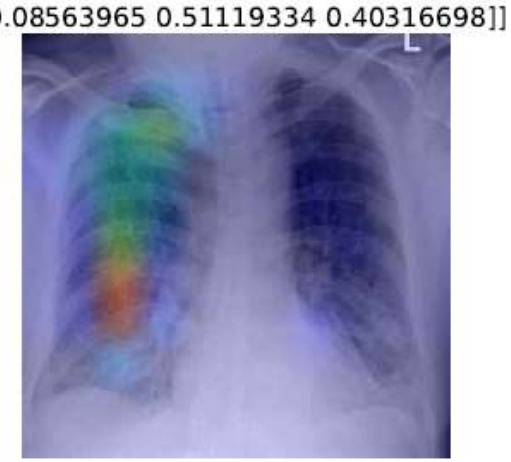
- ¿Qué área de la imagen activa más una etiqueta?











Publication

- I. Rodríguez-García et al., Uncertainty Propagation and Salient Features Maps in Deep Learning Architectures for Supporting Covid-19 Diagnosis. (2021)

¿POR QUÉ SOY UN ELEFANTE?

- ¿Para qué sirve saber por qué esta imagen se clasifica como elefante? Para evitar sesgos y ahorra dinero.

ENIA

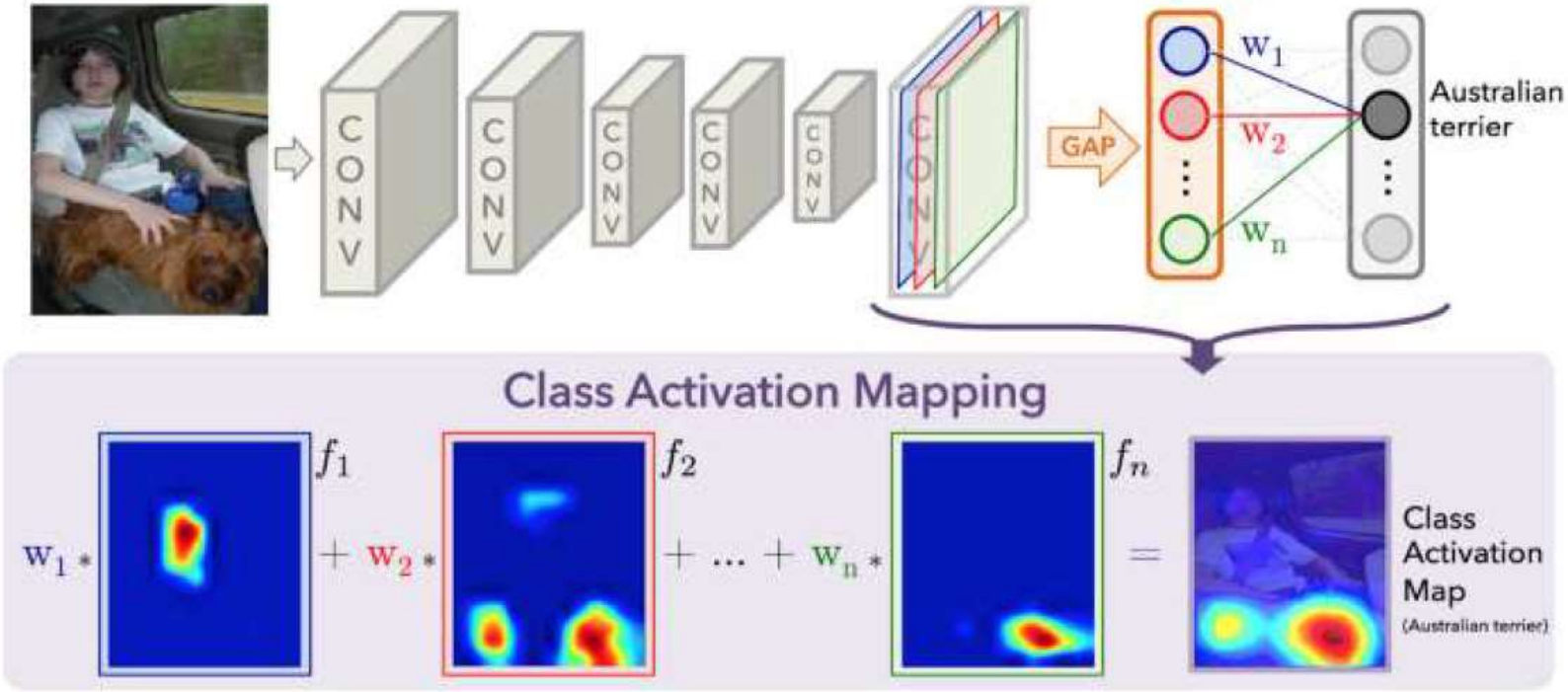
- La Estrategia Nacional de Inteligencia Artificial menciona que las decisiones basadas en IA deben ser transparente y explicables, y sin sesgos.

Publication

- Zech et al., *Confounding variables can degrade generalization performance of radiological deep learning models* (2018)

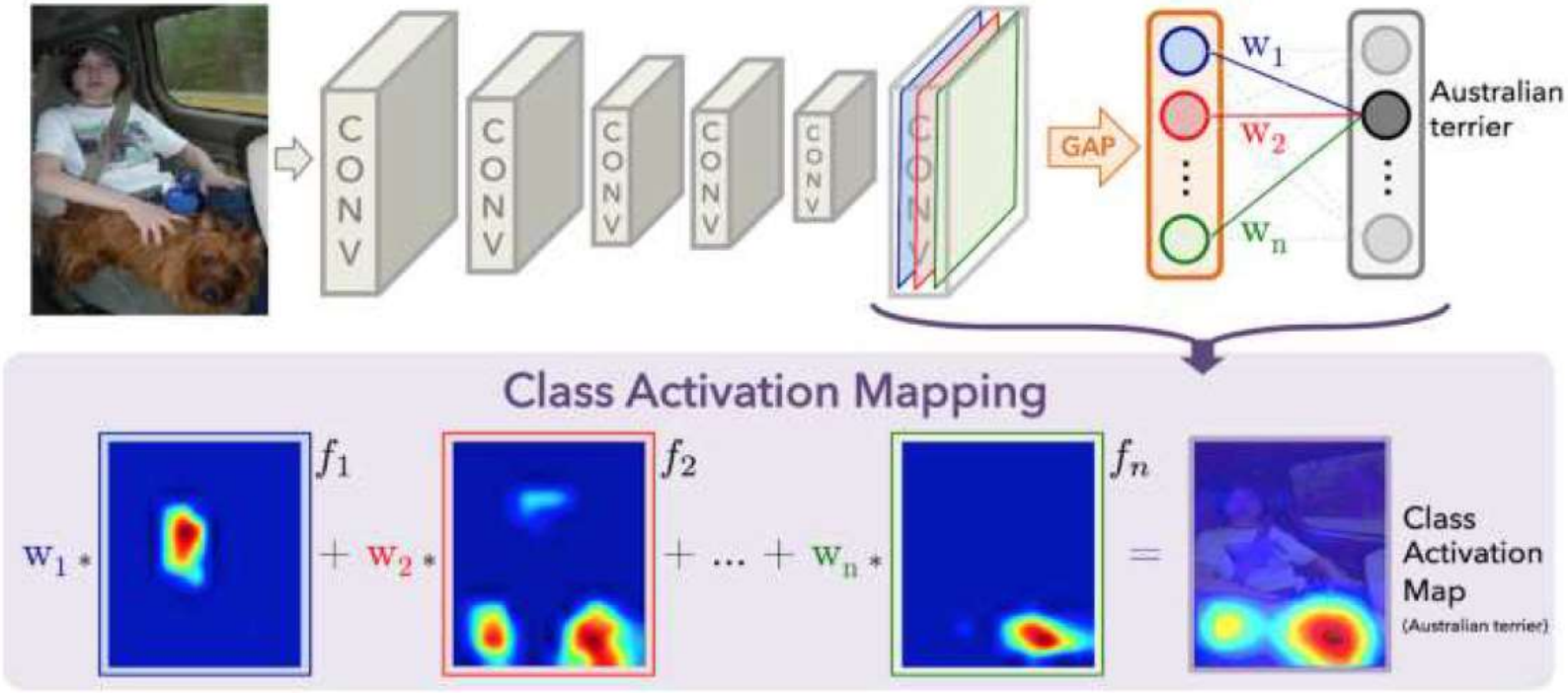
CAM

- *Class Activation Maps (CAM)* ayuda a ilustrar las partes de la imagen que tienen mayor influencia en el pronóstico de la clase.
- Aplicable a problema de clasificación con imágenes.
- Permite localizar objetos.
- Los resultados son presentados como un mapa de calor.

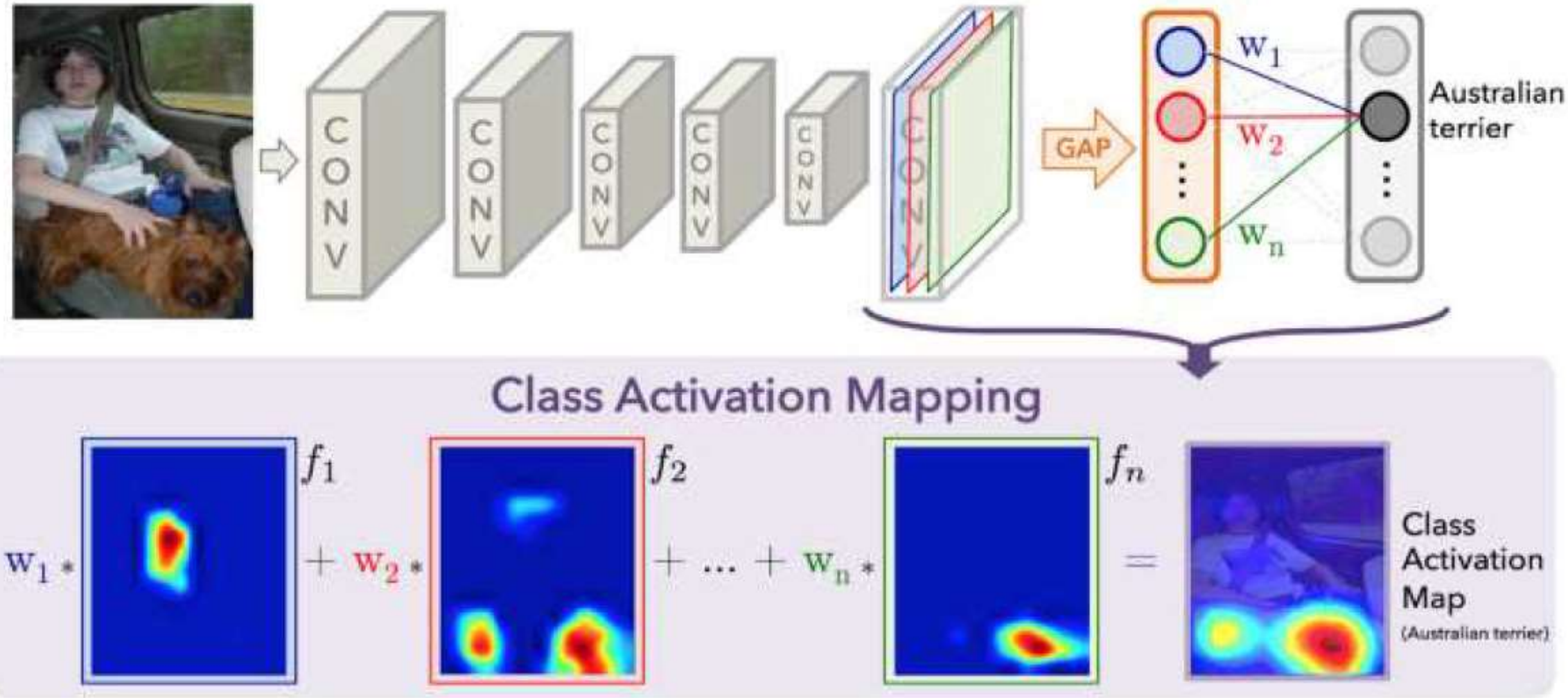


CAM

- Requiere una capa *GAP (Global Average Pooling)*. Esta es su mayor debilidad, puesto que penaliza fuertemente el rendimiento.
- Es obligatorio usar *one-hot-encoding* para las etiquetas de las clases.
- Ha sido la motivación para desarrollos posteriores con mejor rendimiento.

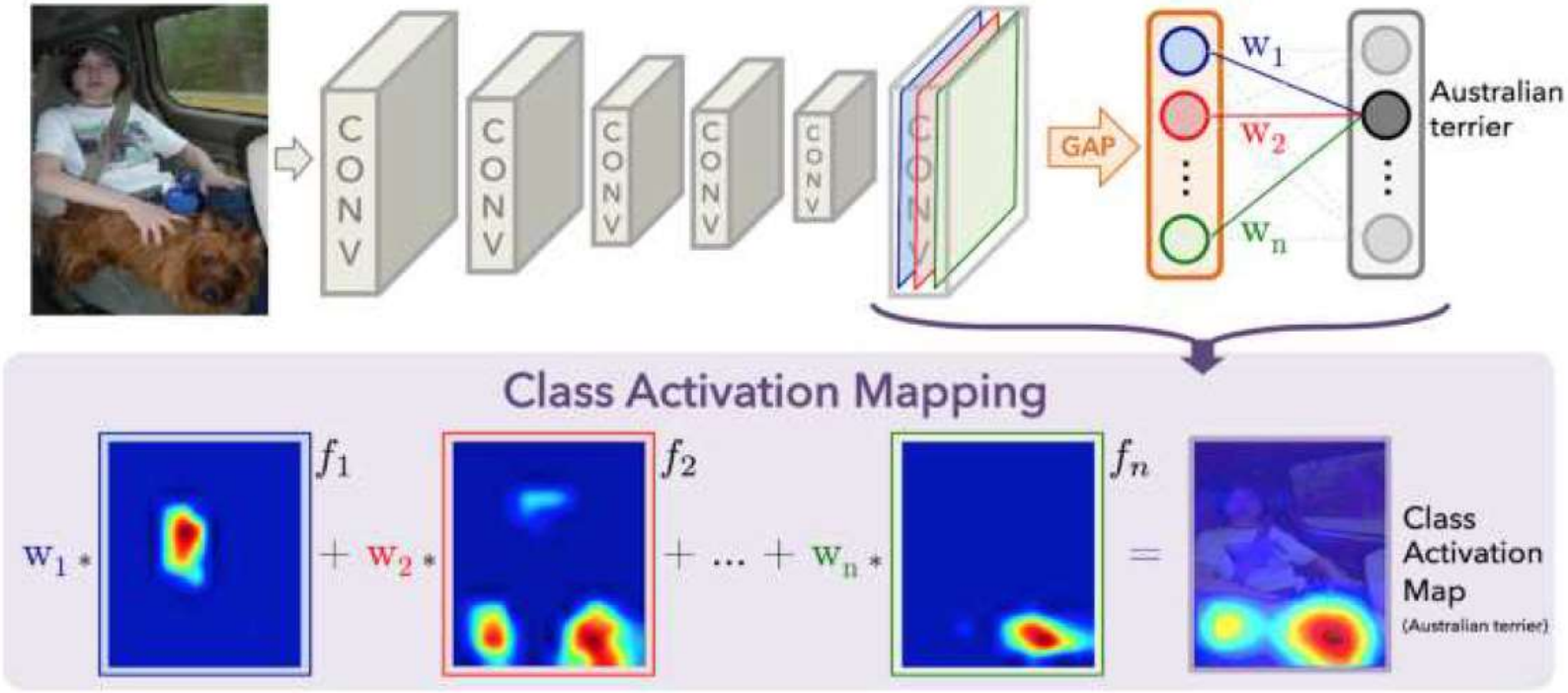


ESQUEMA


CONV
CONV
CONV
CONV
CONV
CONV
GAP
w_1
w_2
w_n
Australian terrier
Class Activation Mapping
$w_1 *$
f_1
$+ w_2 *$
f_2
$+ \ldots + w_n *$
f_n
$=$
Class Activation Map
(Australian terrier)

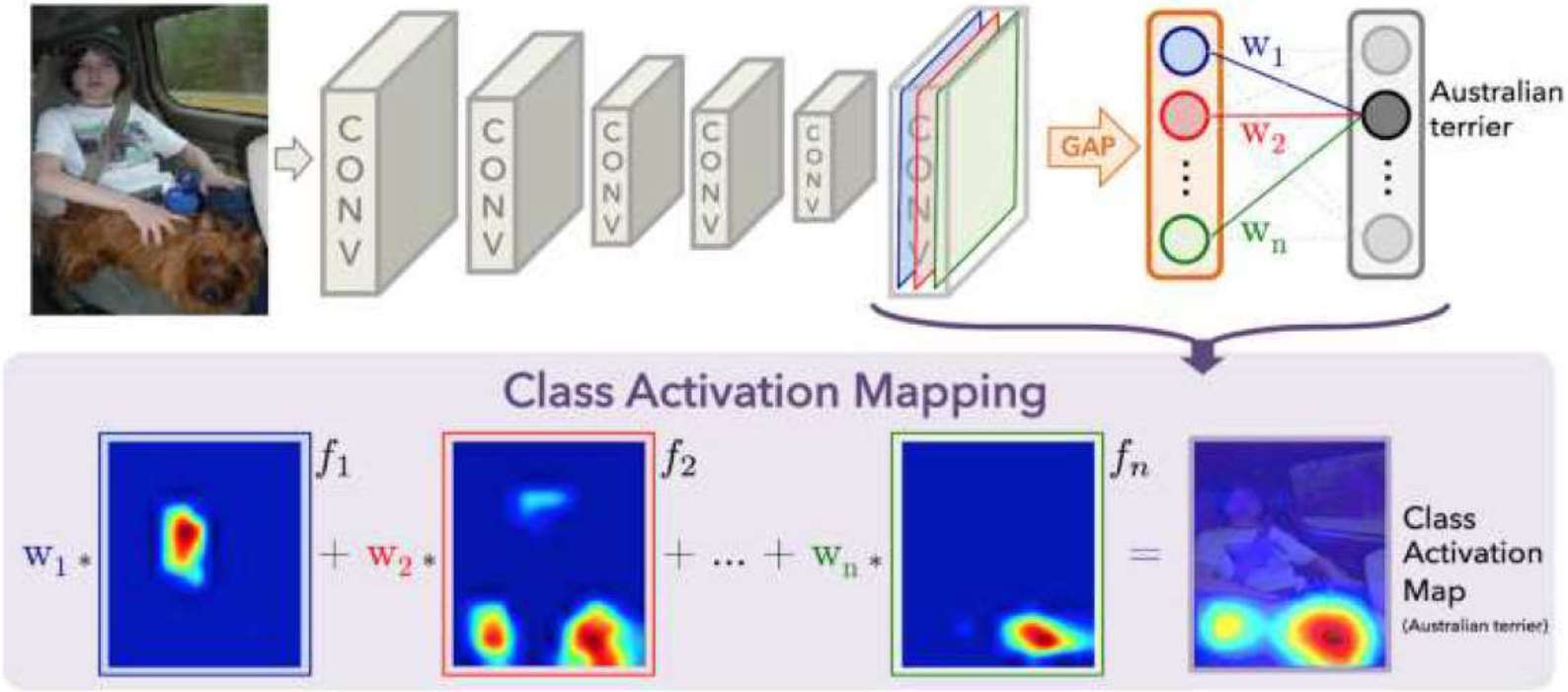
CAM

- CAM requiere de una arquitectura particular: tras la última capa convolucional, debe aparecer una capa *GAP (Global Average Pooling)* y una capa densa (capa de salida).
- La GAP convierte cada canal de la imagen un único valor.
- Este valor alimenta la última capa densa que emite el pronóstico.

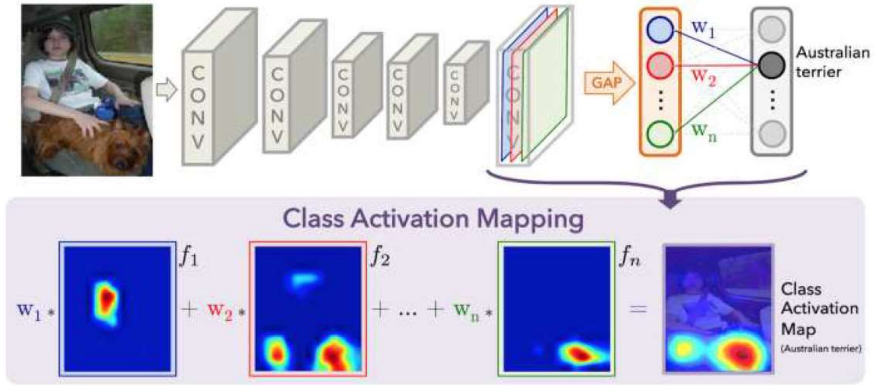


CAM

- Para obtener los mapas de calor, se multiplican los pesos por los mapas de características de la última capa.



CAM



- El mapa de características $A^k \in \mathbb{R}^{u \times v}$ de tamaño u × v.

$$y_c = \sum_k w_k^c \frac{1}{Z} \sum_{ij} A_{ij}^k \tag{1}$$

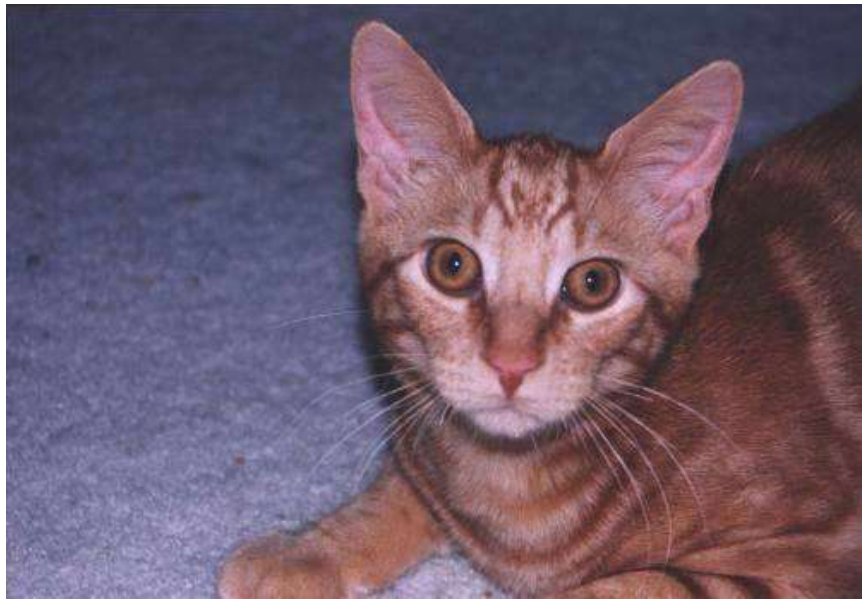
donde $\frac{1}{Z} \sum_i \sum_j A_{ij}^k$ es la acción de capa GAP.

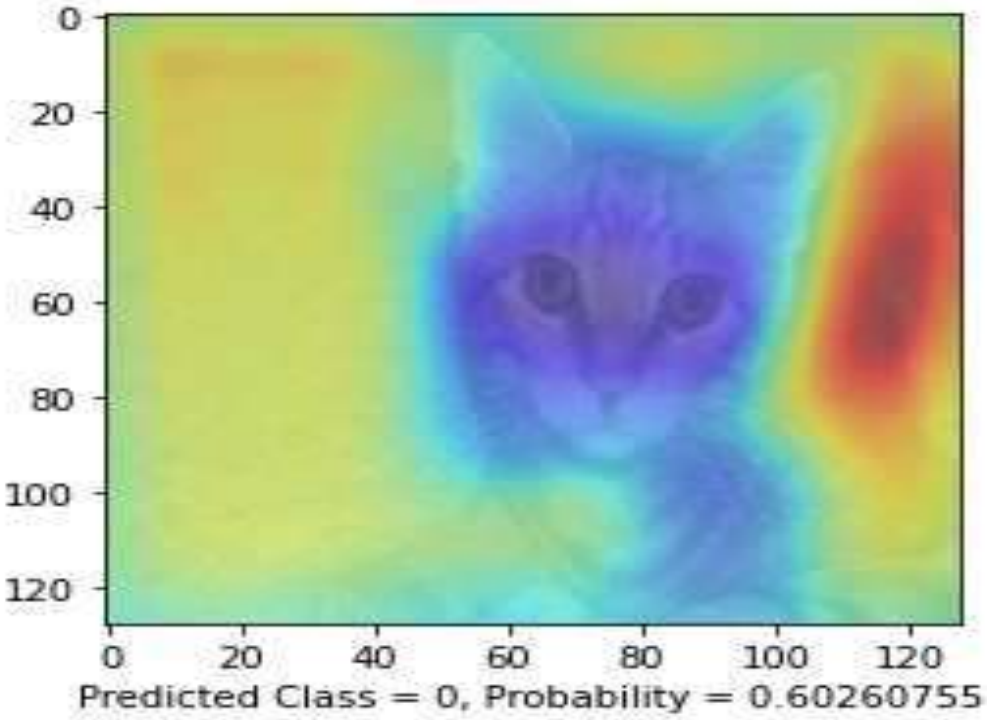
- El orden de los sumatorios puede intercambiarse:

$$y_c = \frac{1}{Z} \sum_{ij} \sum_k w_k^c A_{ij}^k \tag{2}$$

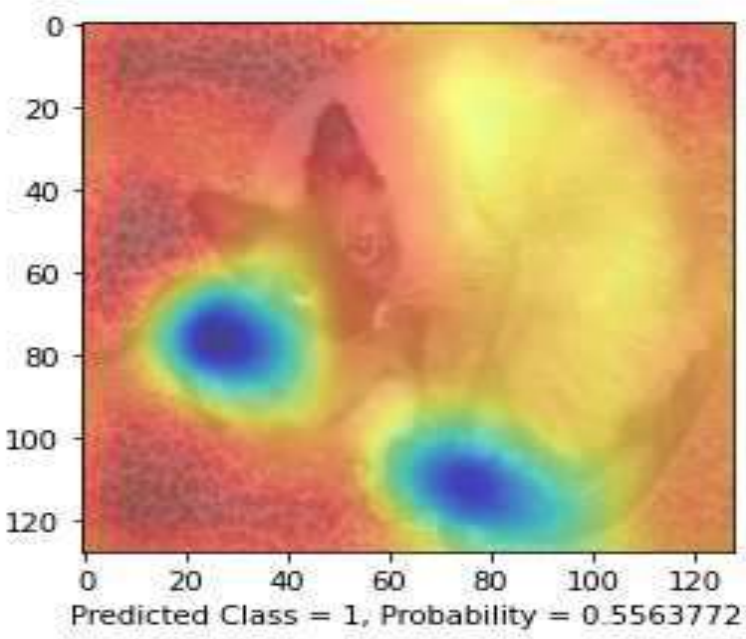
- De forma que para cada píxel, la importancia del mismo para la clase k viene dada por la expresión: $\sum_k w_k^c A_{ij}^k$

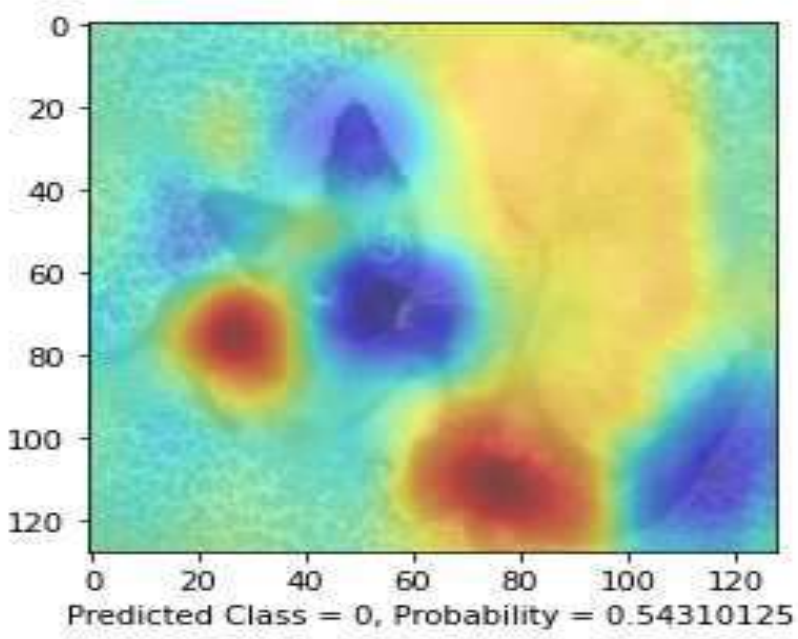
PERROS Y GATOS.



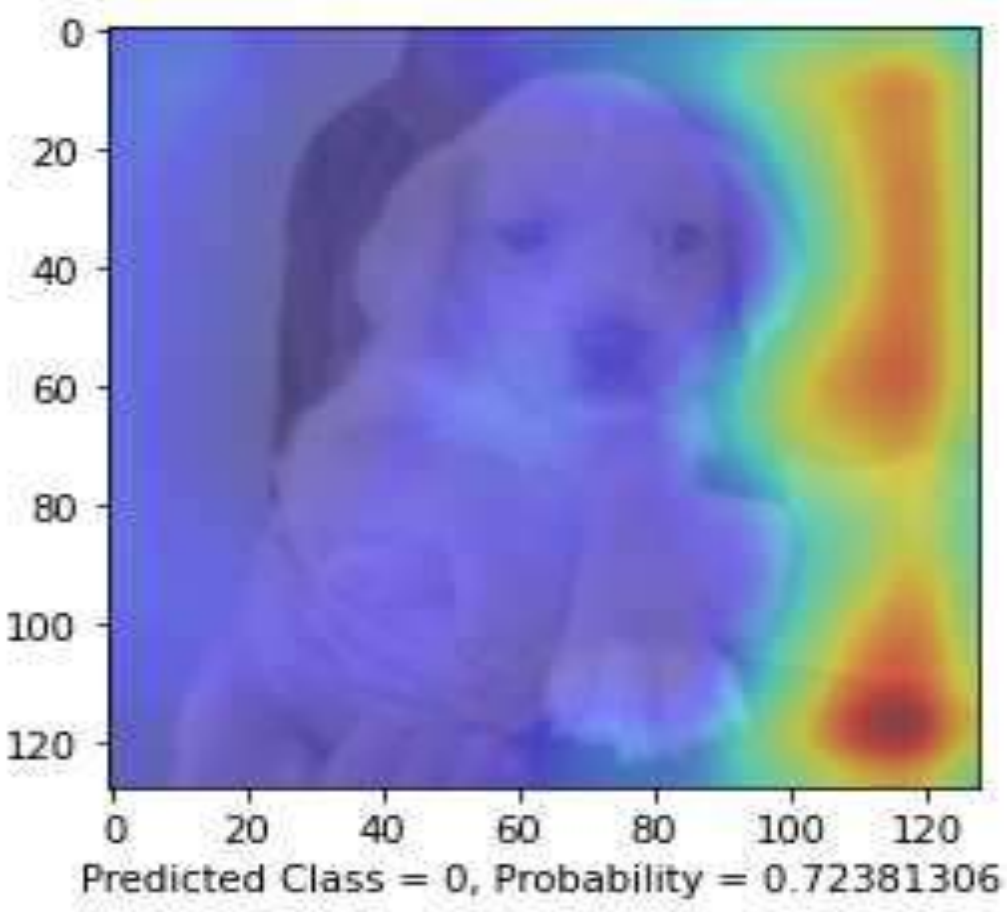


PERROS Y GATOS.

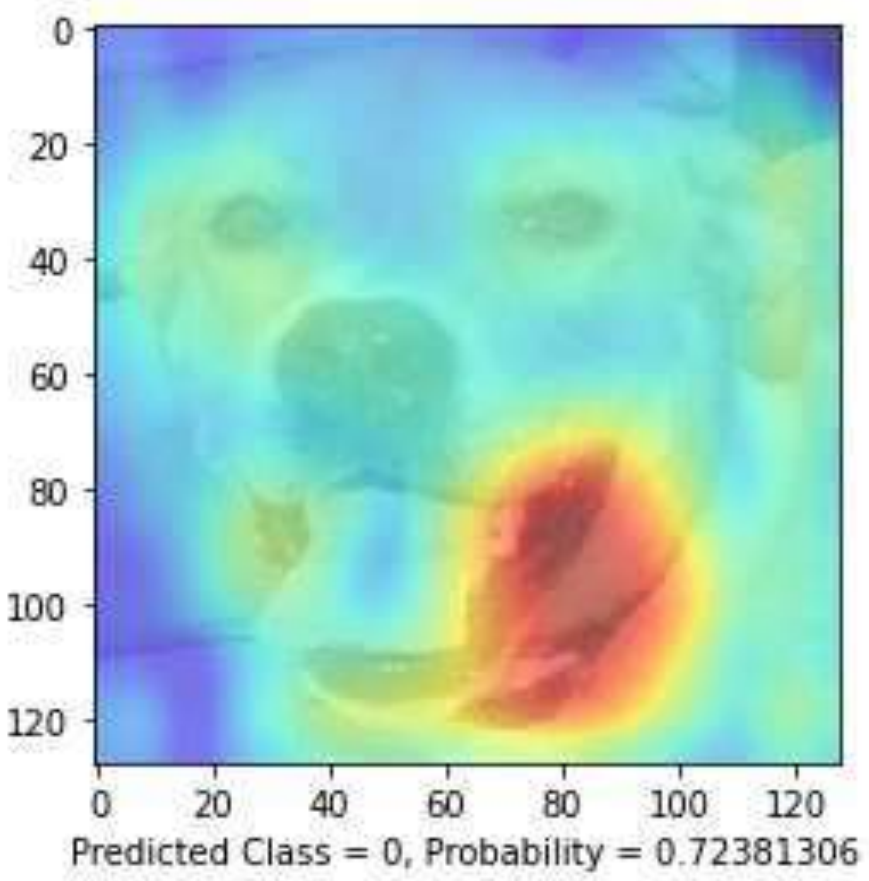




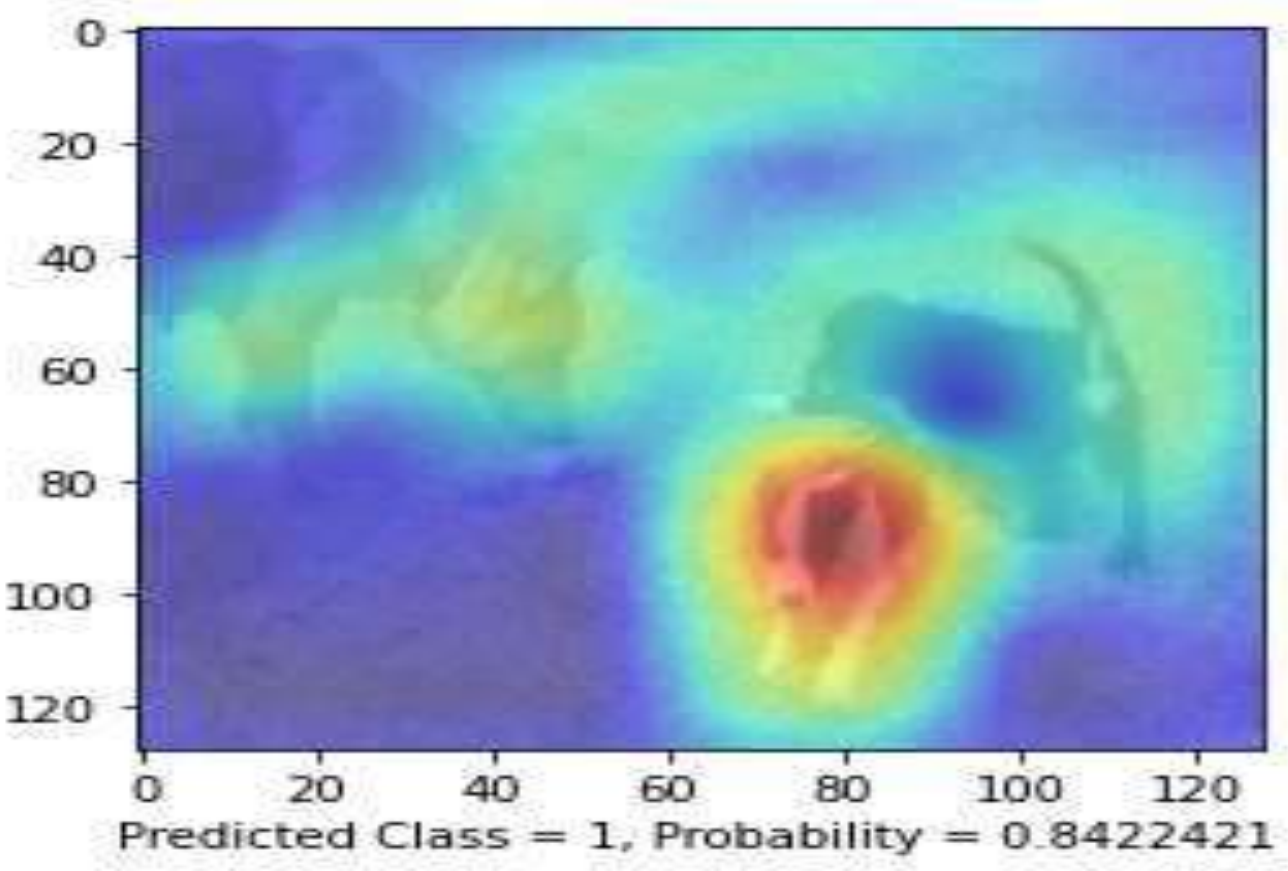
PERROS Y GATOS.



PERROS Y GATOS.



PERROS Y GATOS.

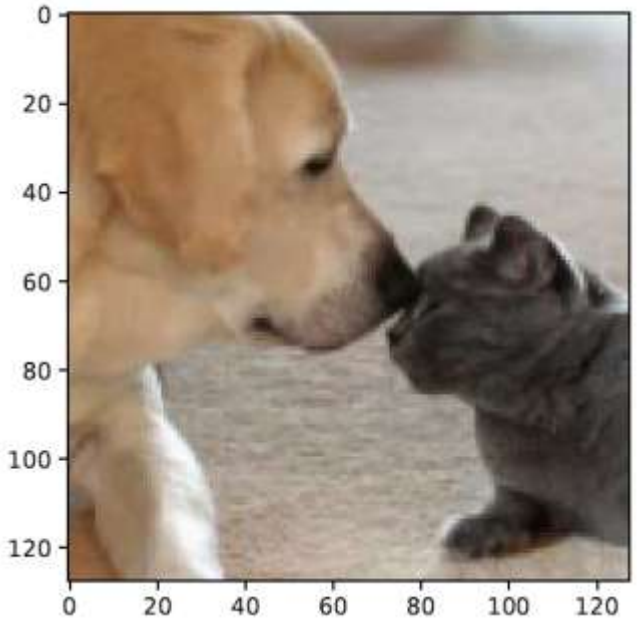


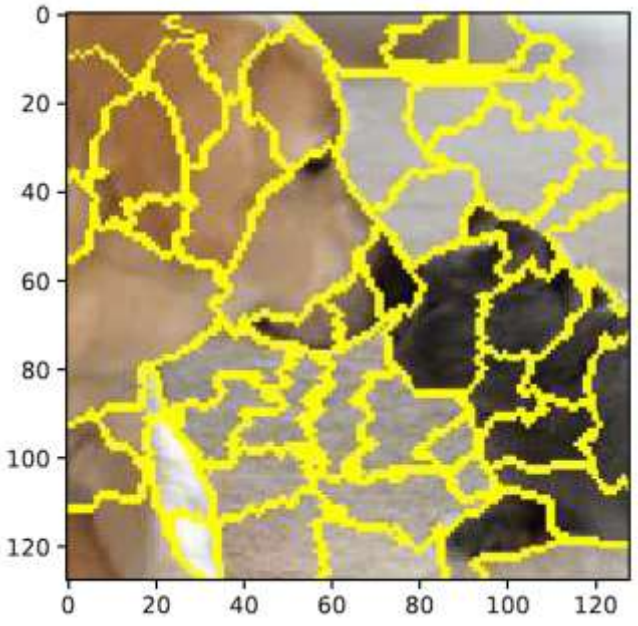
LIMITACIONES DE CAM

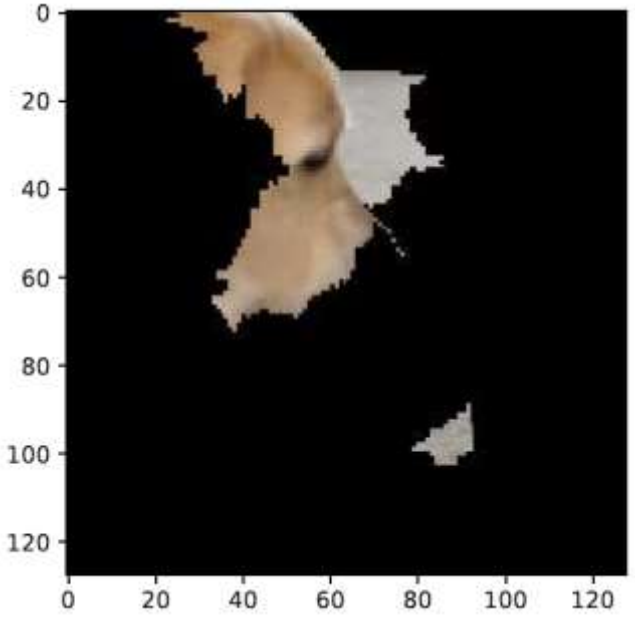
- Necesita forzosamente una capa *GAP (Global Average Pooling)*. Esto obliga a modificar una red que ya puede tener un alto rendimiento.
- Hay que insertar una sola capa densa tras la capa GAP.
- Al modificar la red, se requiere un nuevo entrenamiento con un cierto riesgo de pérdida de rendimiento.
- Solo se puede visualizar el mapa de calor de la última capa.
- El mapa de calor tiene el tamaño de la imagen de la última capa, y por lo tanto, tiene una resolución baja.

LIMITACIONES DE CAM

- Limitaciones en ciertas imágenes donde aparecen elementos de diferentes etiquetas.
- Sería deseable poder segmentar la imagen y ordenar los segmentos por su impacto en el pronóstico de la etiqueta.







Publication

- Ribeiro, Singh and Guestrin. *Why Should I Trust You?: Explaining the Predictions of Any Classifier* (2016)

RESUMEN

- La técnica CAM permite visualizar las características esenciales en la clasificación de imágenes.
- La CAM es la base de otras técnicas más elaboradas que solventan las limitaciones de esta técnica.

Códigos:

- En el código *CAM_ClassActivationMaps_CatDog.ipynb* se muestra un ejemplo del uso de CAM en la clasificación de perros y gatos.
- En el código *CAM_ClassActivationMapswithMaxPooling_4MNIST.ipynb* se muestra un ejemplo de uso conjunto de CAM aplicado al reconocimiento de dígitos escritos a mano.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

AUTOCODIFICADORES.

Identificación de datos anómalos.

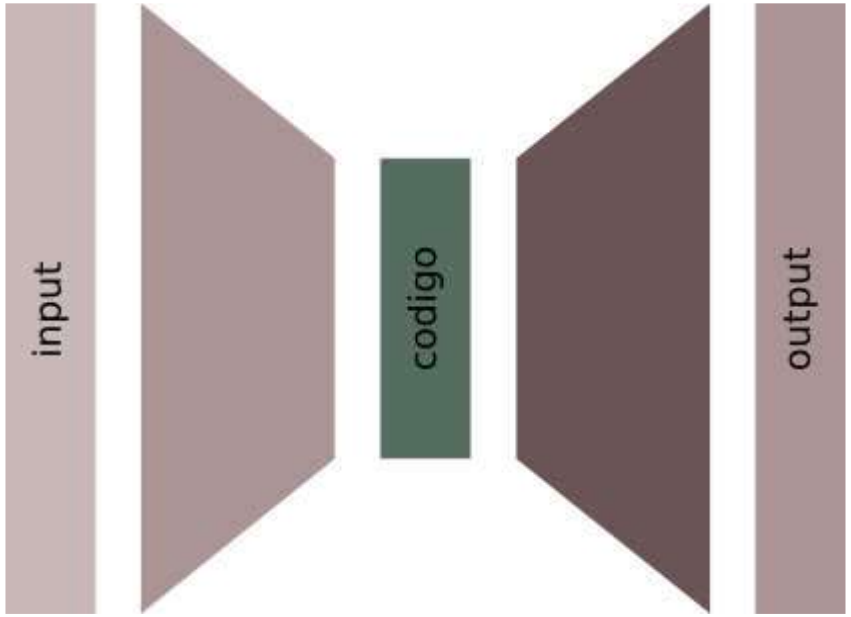
Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

CONTENIDOS.

- AE
- DBSCAN
- AE+DBSCAN
- Resumen

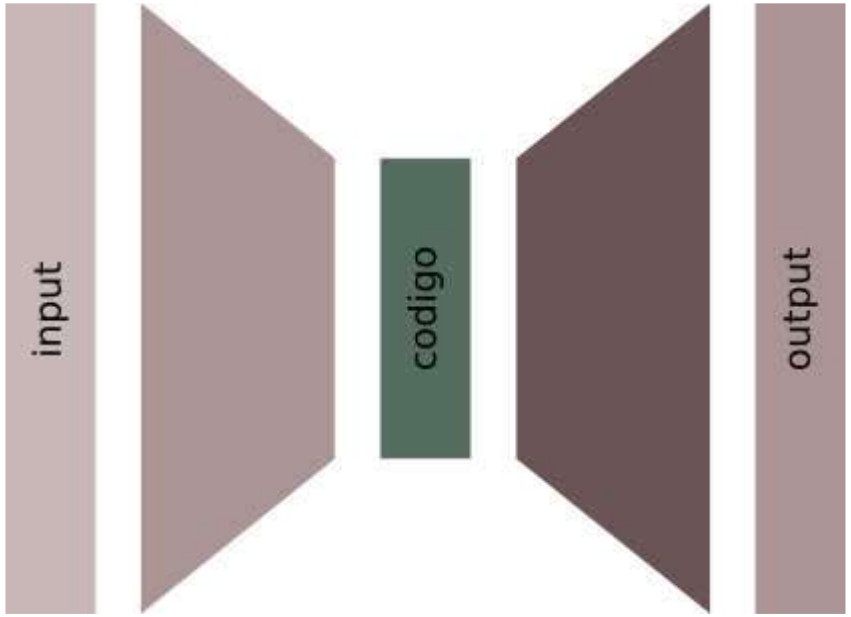
AE.

- Los autocodificadores (AE) son redes neuronales profundas compuestas de dos bloques diferenciadores, el codificador y el decodificador.
- El objetivo de esta redes es la reproducción con exactitud de los datos de entrada, evitando el aprendizaje de la función identidad.



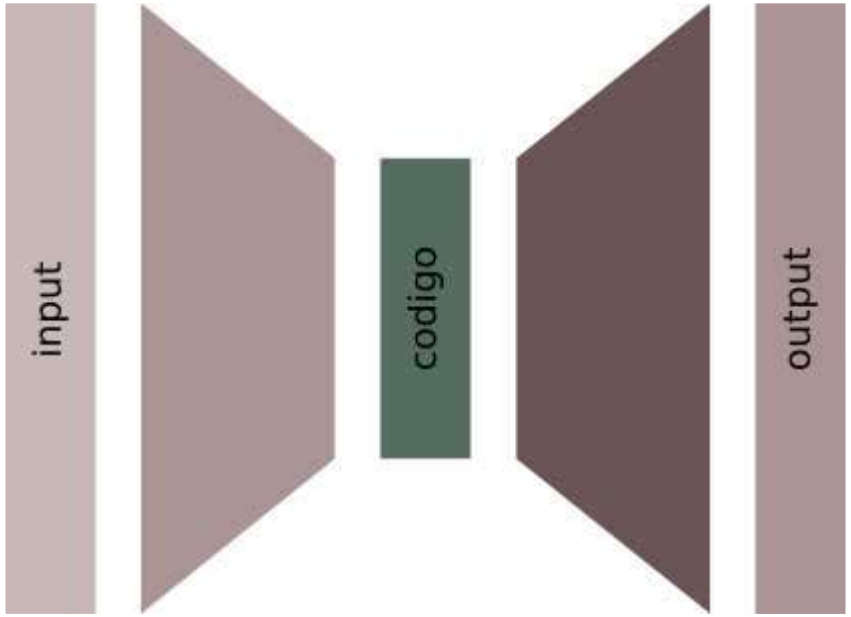
AE.

- Los AE se utilizan para realizar representaciones comprimidas de los datos.
- Como consecuencia se produce una reducción de la dimensionalidad, siendo su salida una representación o reconstrucción de los datos de entrada.



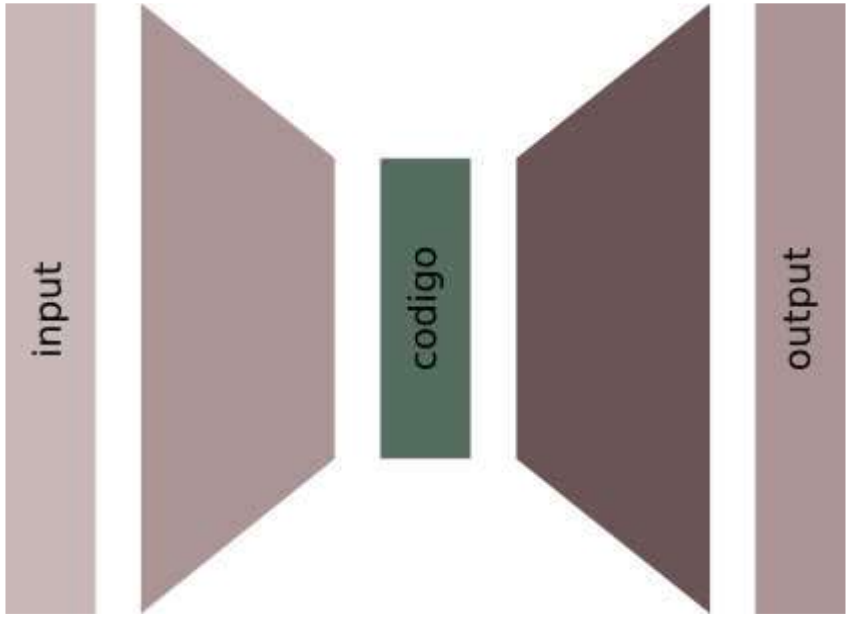
AE.

- También se pueden utilizar para la extracción de características relevantes de los datos y su posterior uso en algoritmos de aprendizaje no supervisado.



AE.

- AE se puede hacer con capas densas o con bloques convolucionales.
- La capa interna del autocodificador se denomina código, *code*.
- La red de un autocodificador está compuesta de dos partes, por una lado una función codificadora $f(\cdot)$, por otro lado una función decodificadora que realiza la reconstrucción $g(\cdot)$, de forma que $g(f(\mathbf{x}))=\mathbf{x}$.



AE.

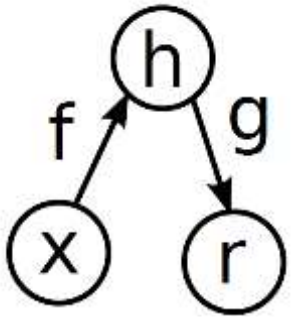
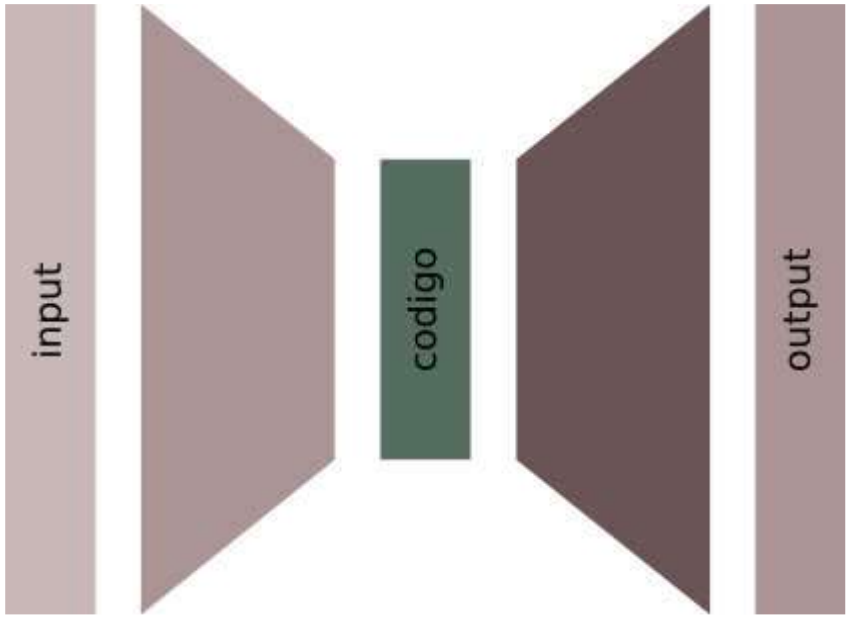
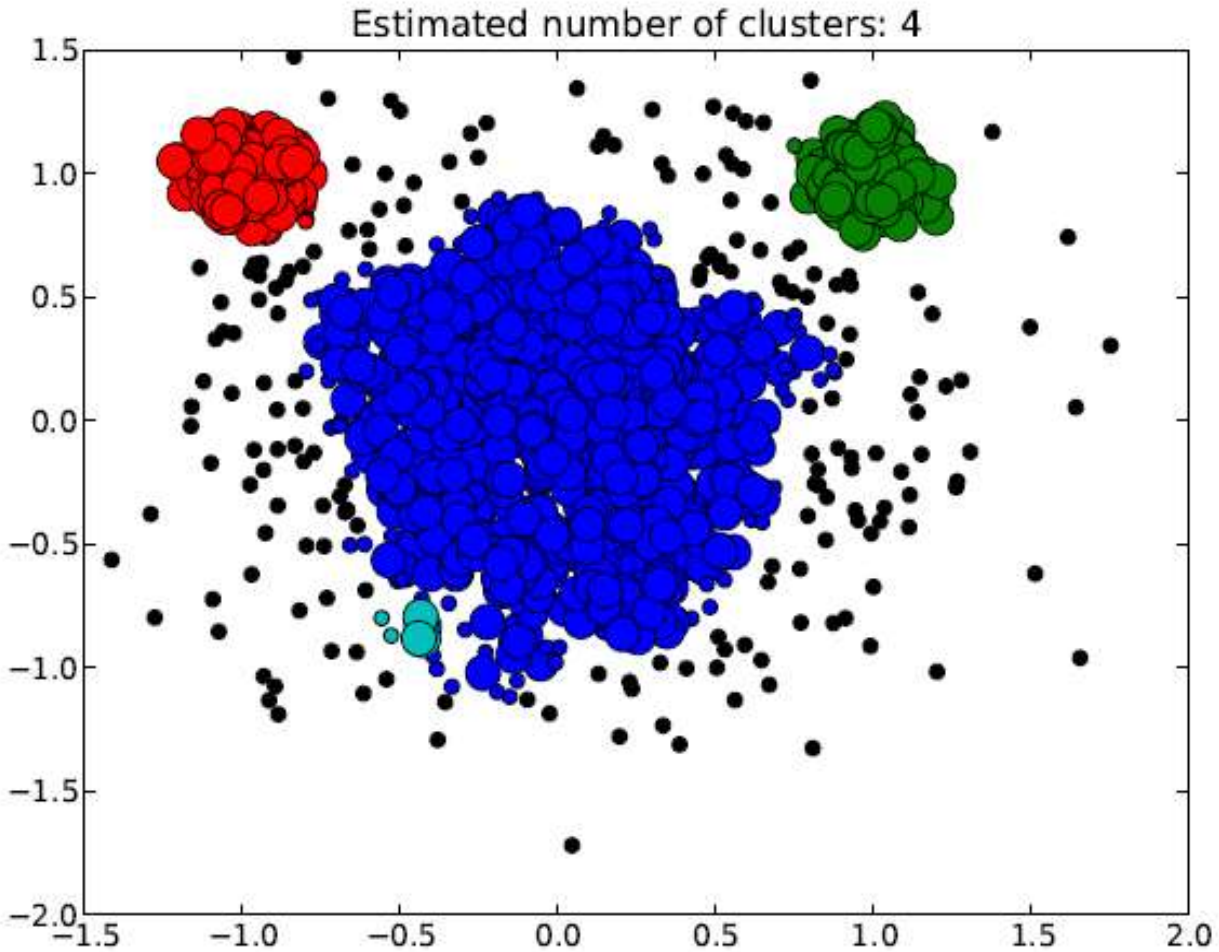


Figura.

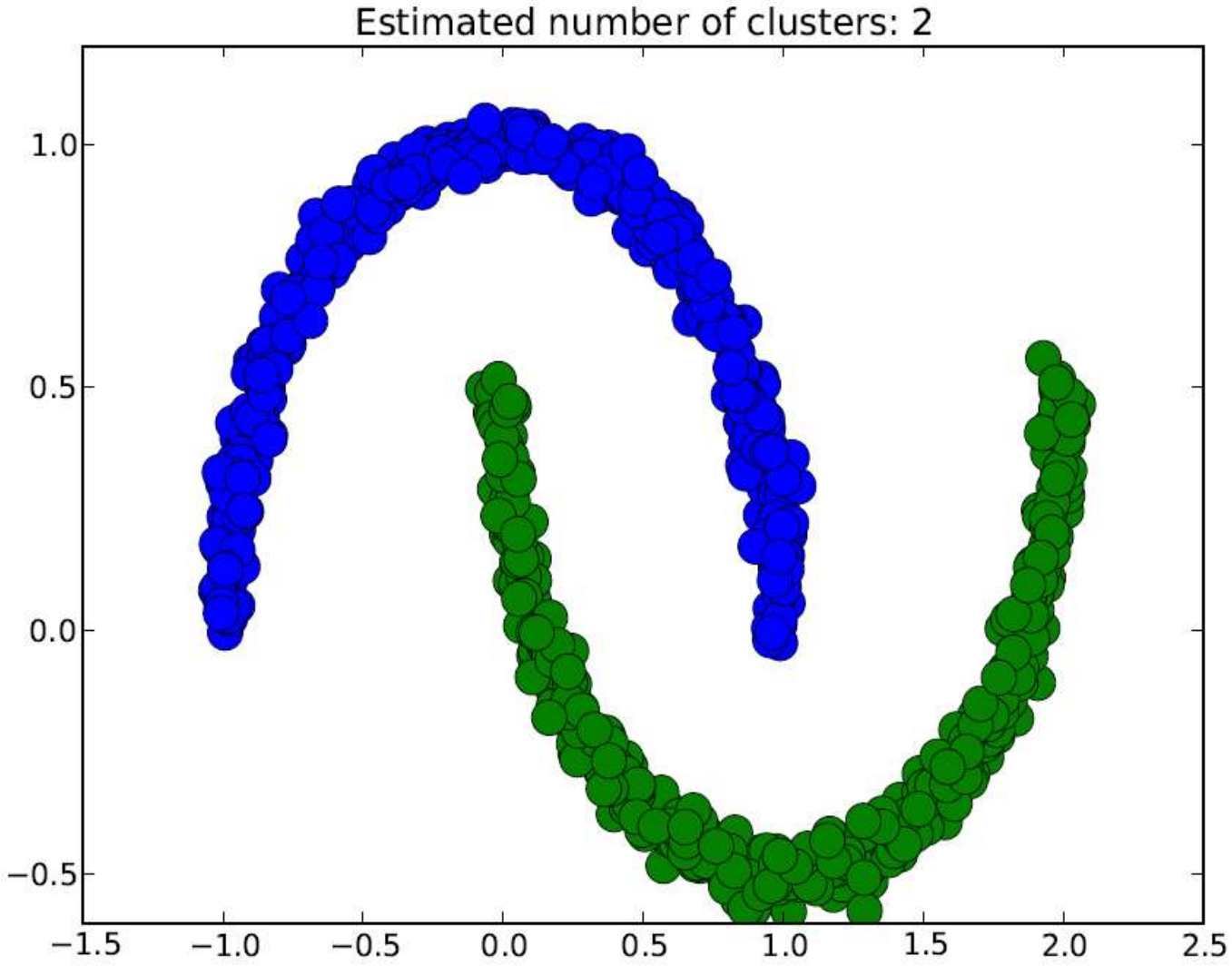
- Estructura general de un autocodificador, mapeando la entrada ***x*** a la salida (llamada reconstrucción) ***r*** a través una representación interna o código ***h***. El autocodificador tiene dos componentes: el codificador ***f*** , que mapea ***x*** a ***h***, y el decodificador ***g***, que mapea ***h*** a ***r***.



CLUSTERING.


Estimated number of clusters: 4
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

CLUSTERING.


Estimated number of clusters: 2
1.0
0.5
0.0
−0.5
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
2.5

MÉTODOS BASADOS EN DENSIDAD-DISTANCIA

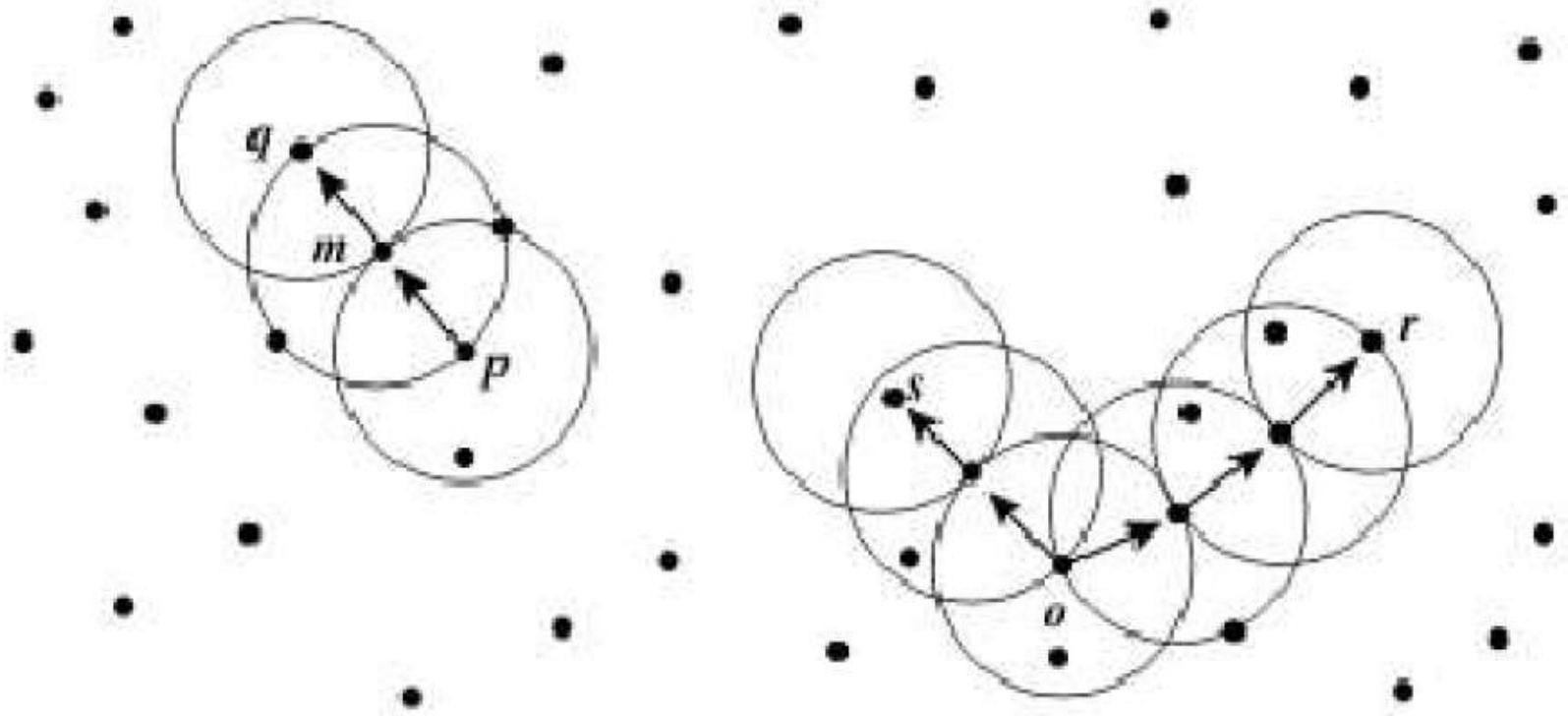
- DBSCAN es un método de *clustering* basado en densidad.
- La idea es hacer crecer un clúster siempre cuando la densidad en el entorno del objeto exceda de un umbral.
- Este tipo de método permite la detección de clústeres de forma arbitraria, sirviendo además para filtrar datos ruidosos.

DBSCAN.

- DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) es un algoritmo basado en métodos de densidad.
- DBSCAN hace crecer regiones con suficiente alta densidad en grupos y descubre grupos con forma arbitraria.
- Estos grupos están separados por regiones de baja densidad de objetos (ruido).

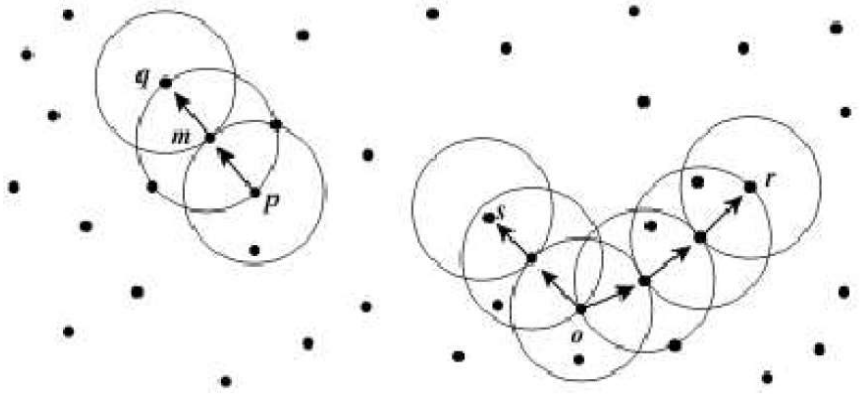
DBSCAN.

- Ejemplo con MinPts=3 y ∈ como el radio de los círculos.



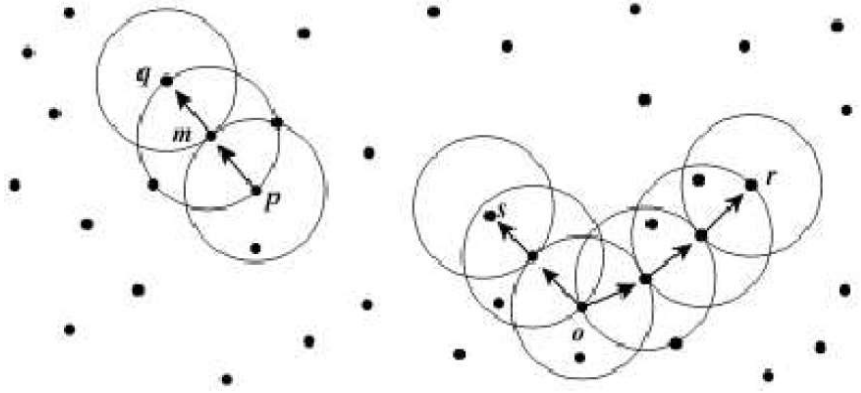
DBSCAN.

- Los parámetros esenciales del algoritmo son el radio ∈ y el número de puntos mínimos *MinPts*.
- Los puntos: 'm', 'p', 'o' son puntos nucleares (*core*) porque están en un ∈-vecindario y contienen el número mínimo de puntos.
- Puntos fronterizos (*border point*): Son los puntos que tienen menos de *MinPts* vecinos dentro de su vecindario de radio ∈, pero están en la vecindad de un punto nuclear. Por ejemplo 'q' y 'r'.
- Puntos ruidosos (*noise point*): Son aquellos puntos que no caen en ninguna de las dos categorías anteriores. Los que están fuera de los círculos y, por lo tanto fuera, de los grupos.

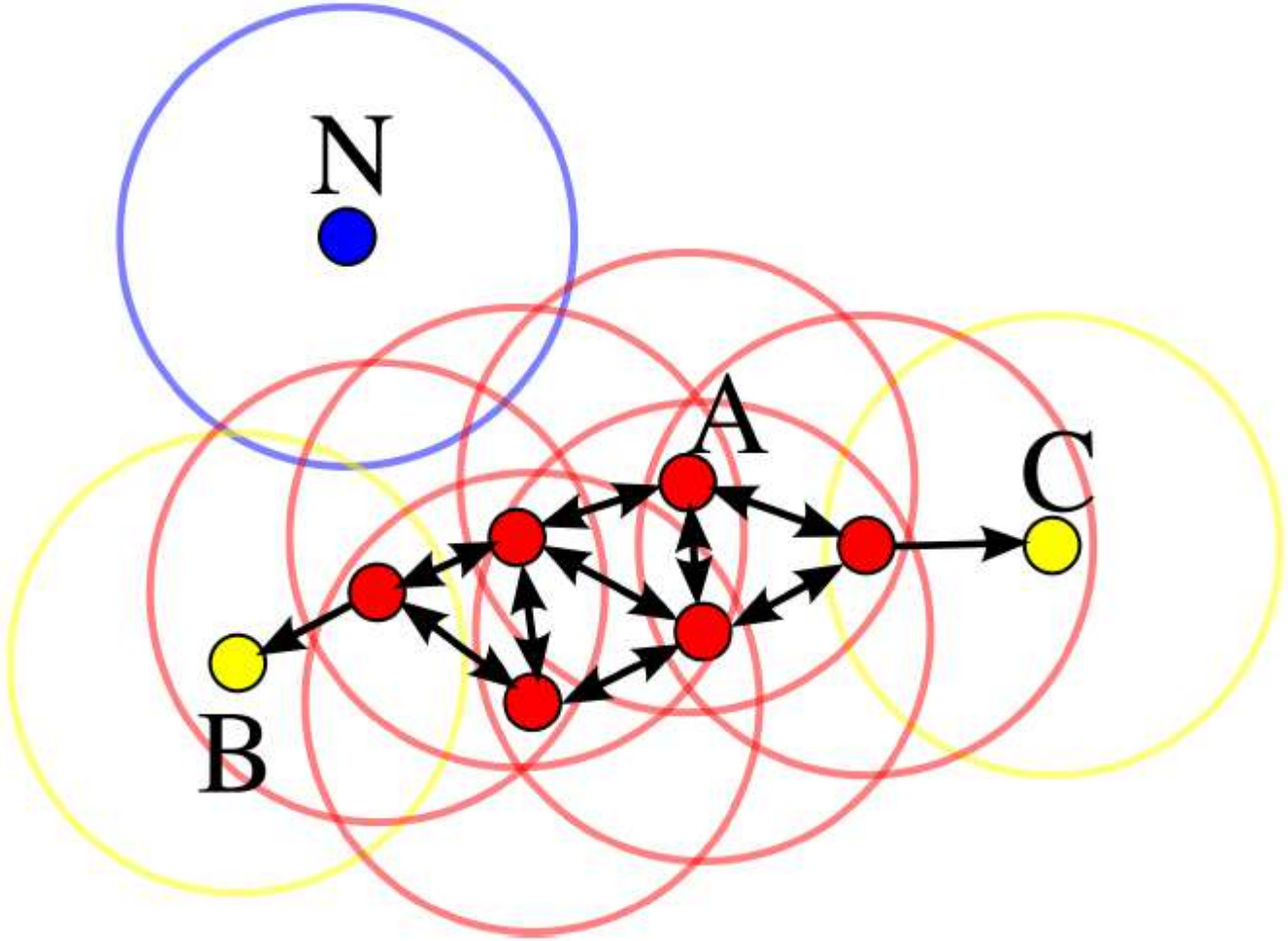


DBSCAN VI.

- 'q' es directamente densidad-alcanzable desde 'm', y 'm' lo es desde 'p' (y 'p' desde 'm').
- 'q' es indirectamente densidad-alcanzable desde 'p' porque 'q' lo es directamente desde 'm' y 'm' lo es desde 'p'. Sin embargo, 'p' no es alcanzable desde 'q' porque 'q' no es *core*.
- De igual forma, 'r' y 's' son alcanzables desde 'o', y 'o' es alcanzable desde 'r'.



DBSCAN.



DBSCAN.

¿Cómo funciona DBSCAN?

- DBSCAN busca clústeres comprobando en el $\in$-vecindario de cada punto.
- Si en el vecindario de un punto 'p' hay más de *MinPts*, un nuevo clúster con 'p' como núcleo es creado.
- DBSCAN iterativamente recolecta los puntos que son directamente alcanzables desde estos objetos núcleo.
- El proceso termina cuando no se pueden añadir nuevos puntos a ningún clúster.

DBSCAN.

Fortalezas.

- Encuentra clústeres no separables linealmente.
- No necesita asumir un número fijo de clústeres.
- No depende de las condiciones de inicio.

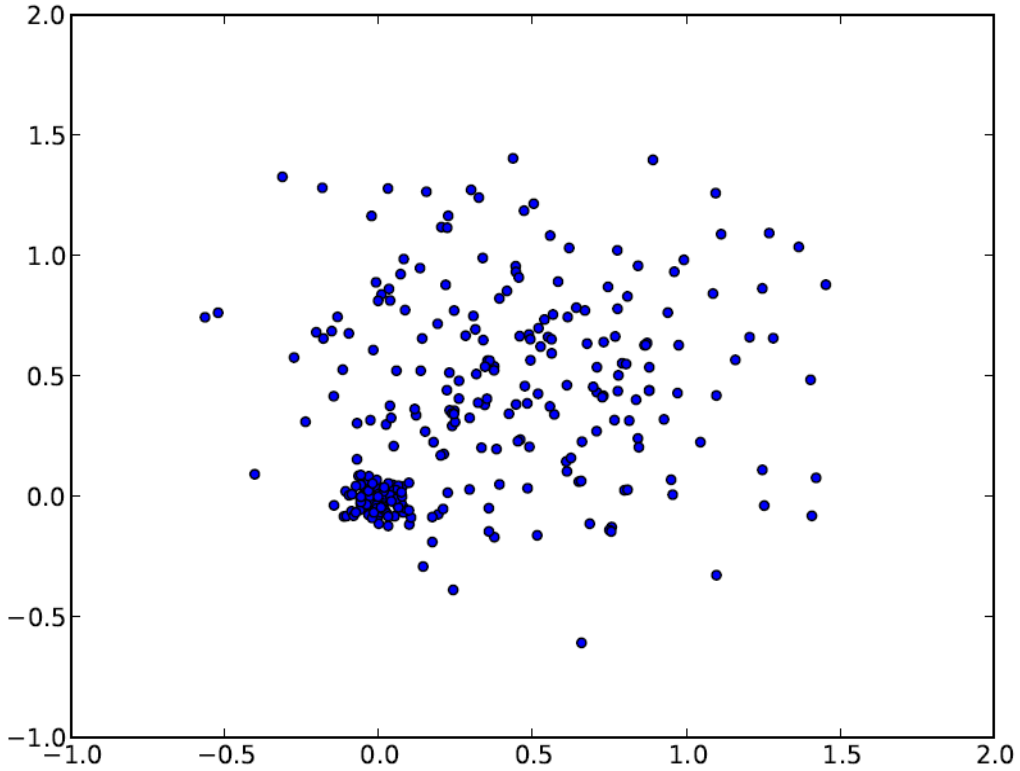
DBSCAN.

Debilidades.

- Asume densidades similares en todos los clústeres, lo que conlleva dificultades para separar clústeres donde las densidades son muy diferentes.

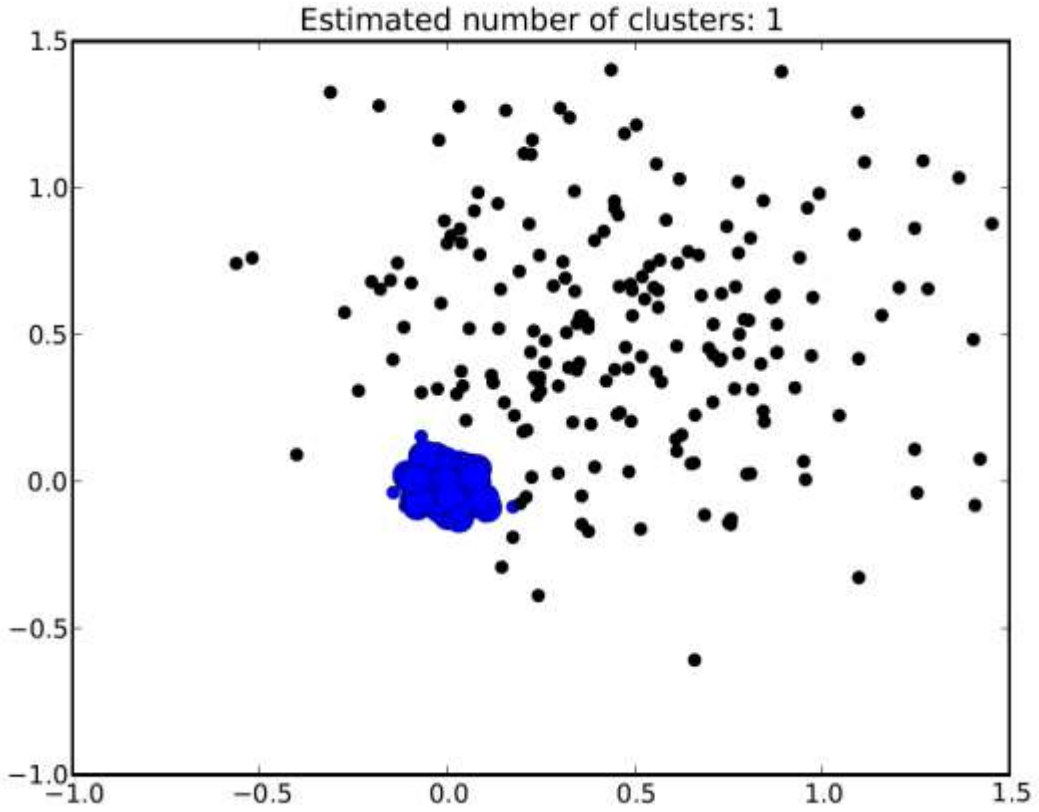
DBSCAN NO ES MÁGICO

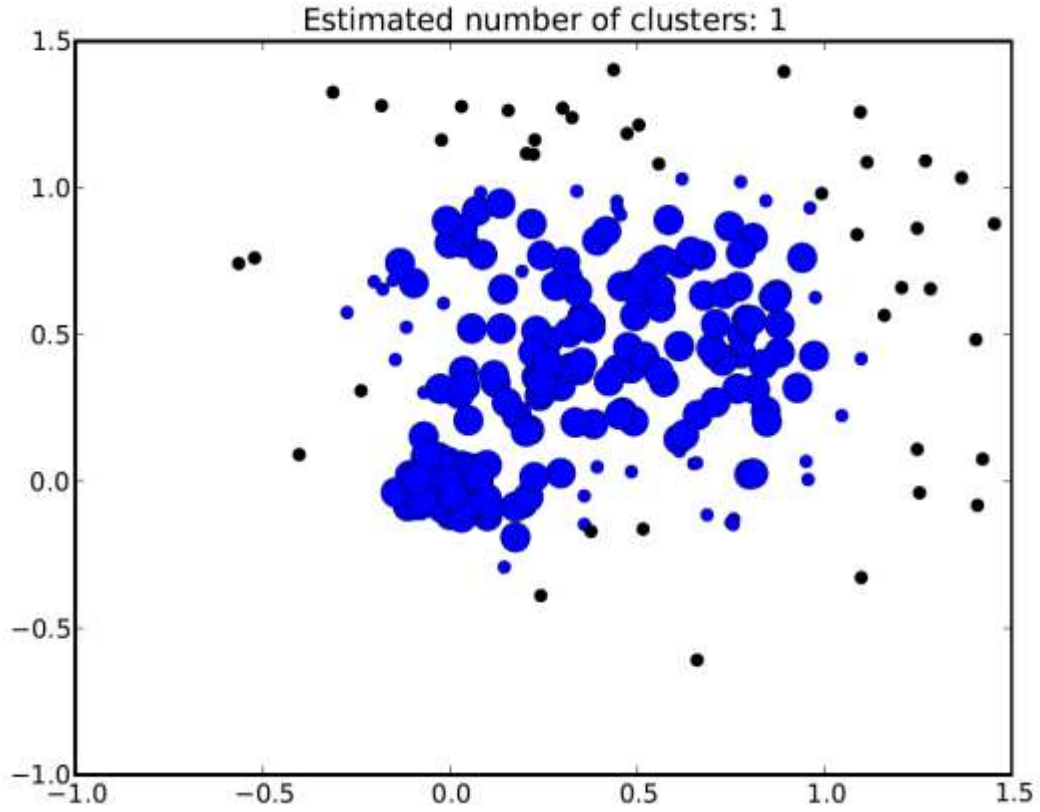
- ¿Qué pasa si tenemos un clúster disperso y uno compacto uno cercano al otro?



DBSCAN NO ES MÁGICO

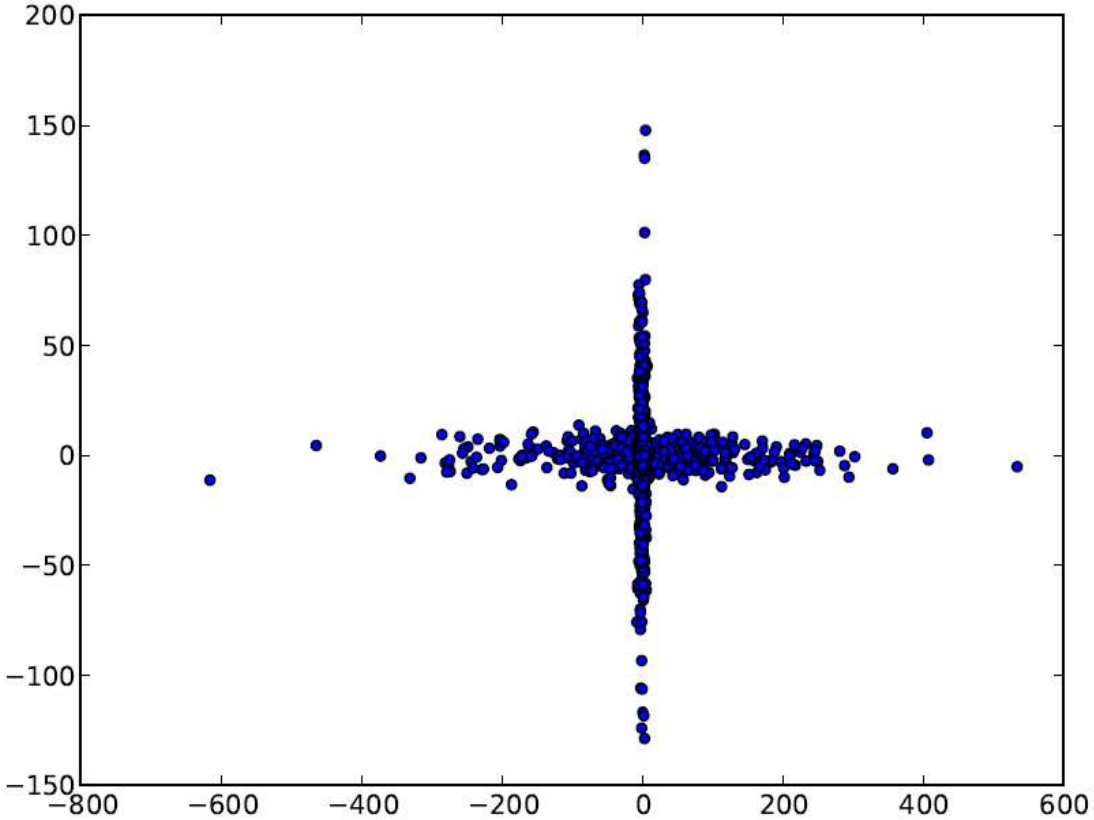
- ¿Qué pasa si tenemos un clúster disperso y uno compacto uno cercano al

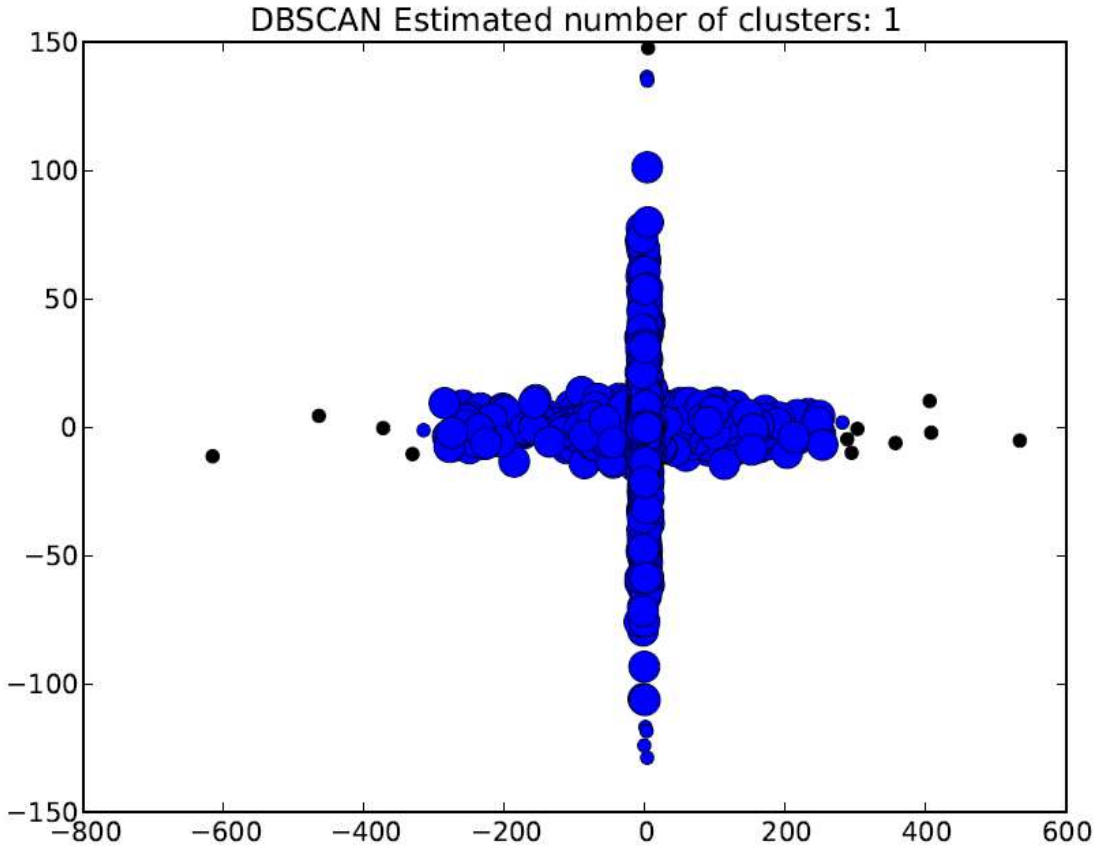




DBSCAN NO ES MÁGICO

- ¿Qué pasa si tenemos un clúster disperso y uno compacto uno cercano al

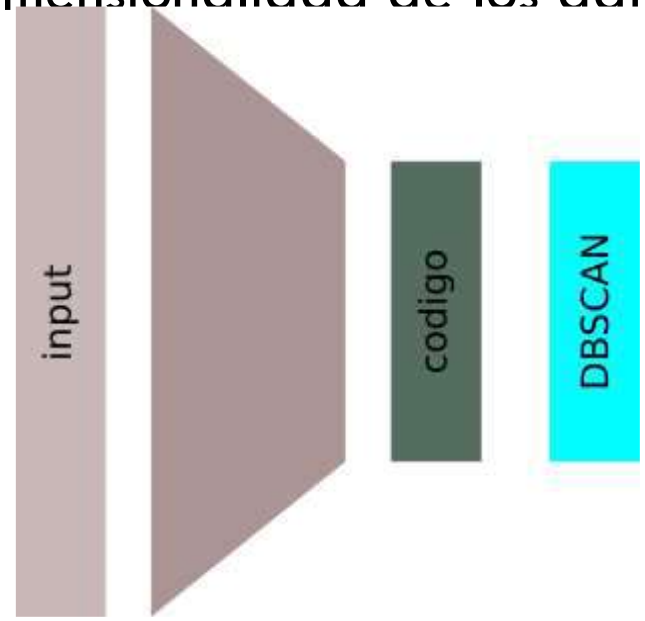




AE+DBSCAN

AE+DBSCAN

- La representación compacta de los datos en el AE es usada como entrada de DBSCAN.
- Permite reducir la dimensionalidad de los datos



RESUMEN

- Ejemplo de uso de redes neuronales para reducir la dimensionalidad de datos.
- Hibridación de redes neuronales y algoritmos de aprendizaje no supervisado para la detección de datos anómalos.

Código:

- Código ejemplo de autocodificador basado en CNN acoplado a DBSCAN sobre mapas de O_3 en Madrid: *DBSCAN+AE.ipynb*.
- Código ejemplo de DBSCAN sin AE para comparar la diferencia en los resultados.

Redes Neuronales Aplicadas a Problemas Científico-Técnicos

REDES NEURONALES RECURRENTES.

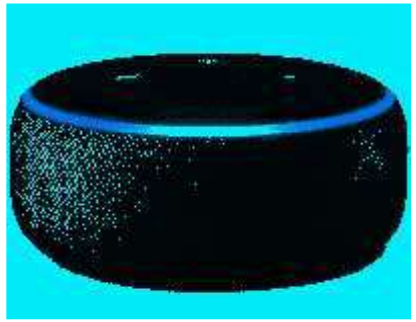
Miguel Cárdenas-Montes
CIEMAT
miguel.cardenas@ciemat.es

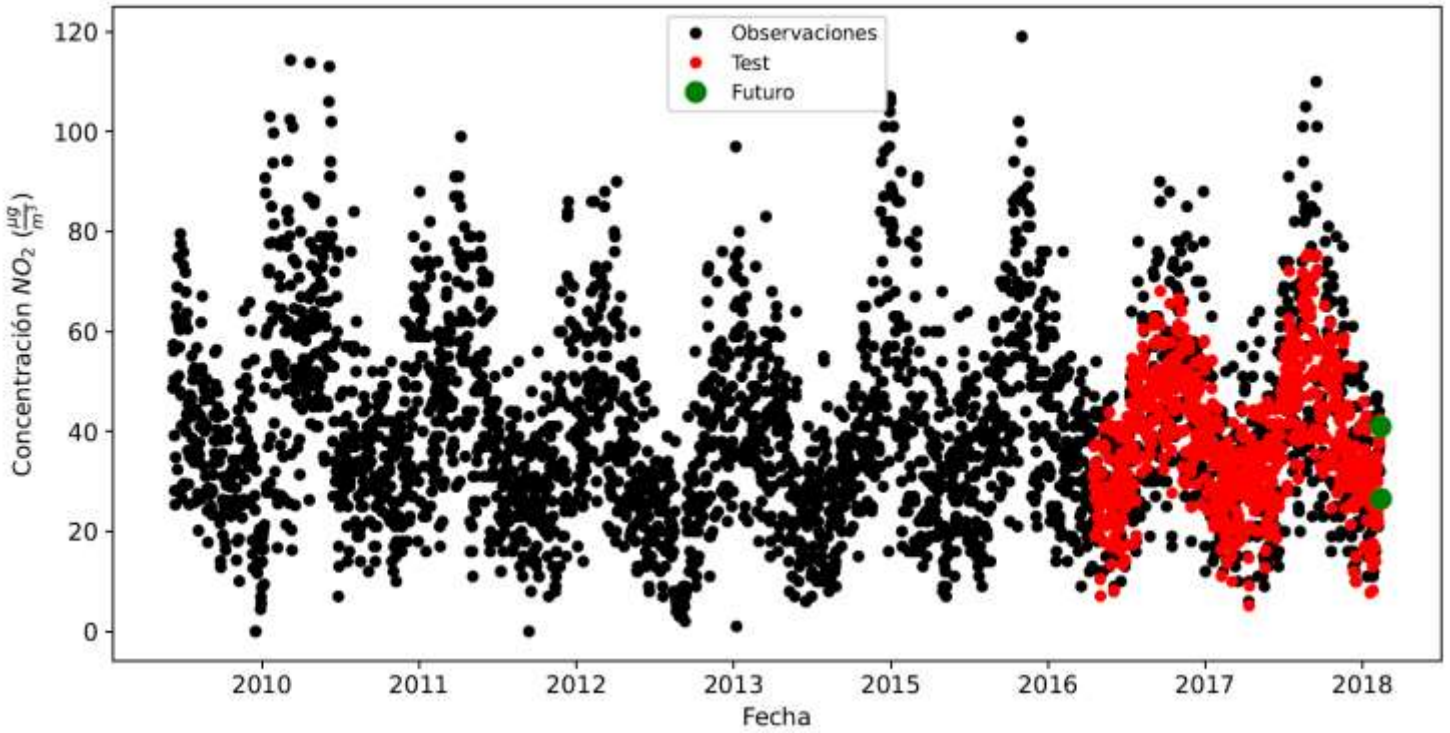
CONTENIDOS.

- Introducción
- RNN
- LSTM
- Resumen

REDES NEURONALES RECURRENTES.

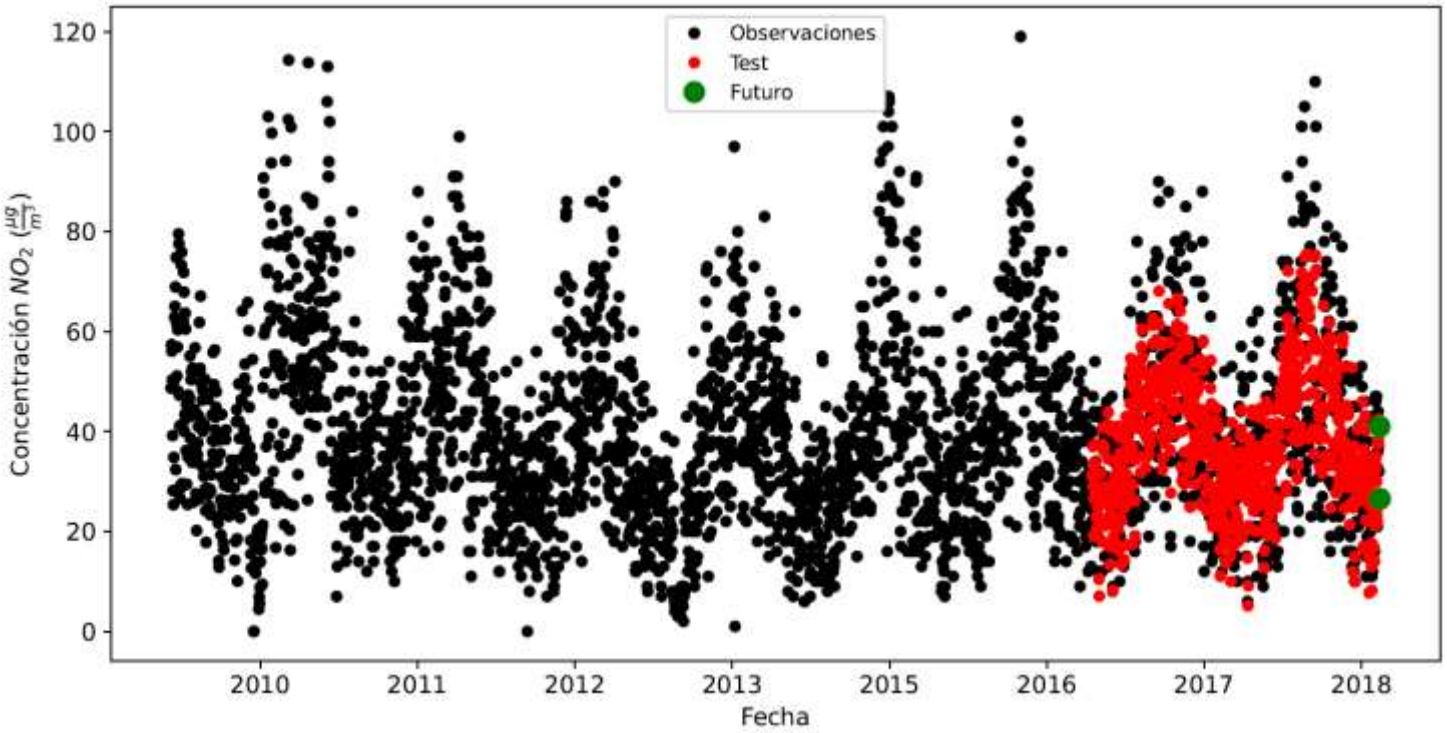
- Las redes neuronales recurrentes (RNN) son redes especializadas en el procesamiento de datos basados en secuencias temporales: series temporales numéricas (series temporales) y texto (Procesamiento de Lenguaje Natural, PLN).
- Tanto las series temporales com el PLN tienen gran importancia en la ciencia y la industria.

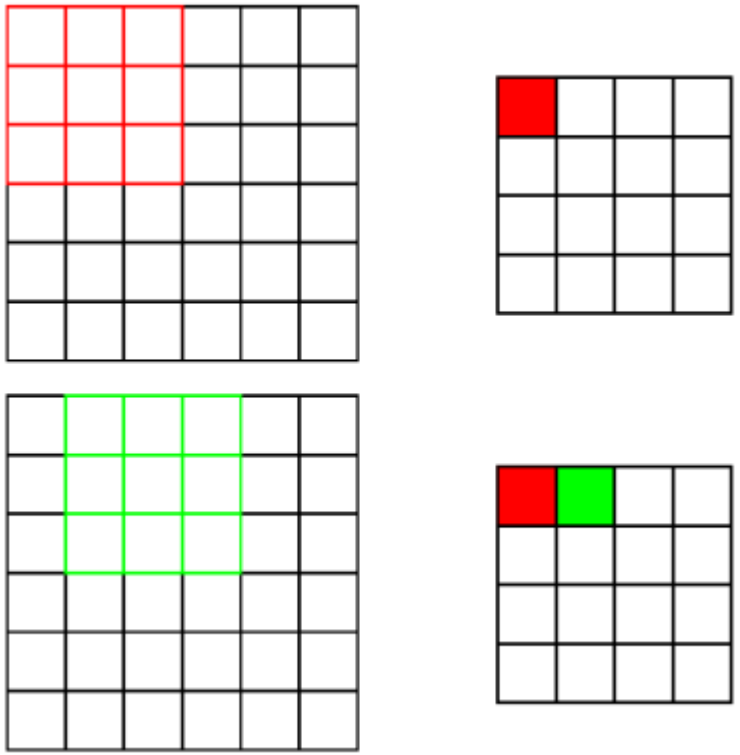




REDES NEURONALES RECURRENTES.

- Si las CNN explotan las correlaciones espaciales de los datos, las RNN explotan las correlaciones temporales.





REDES NEURONALES RECURRENTES.

- MLP no es capaz de explotar las correlaciones temporales de los datos de entrada.
- Cualquier permutación de variables cada ejemplo de datos de entrada con sus correspondientes pesos debe dar el mismo

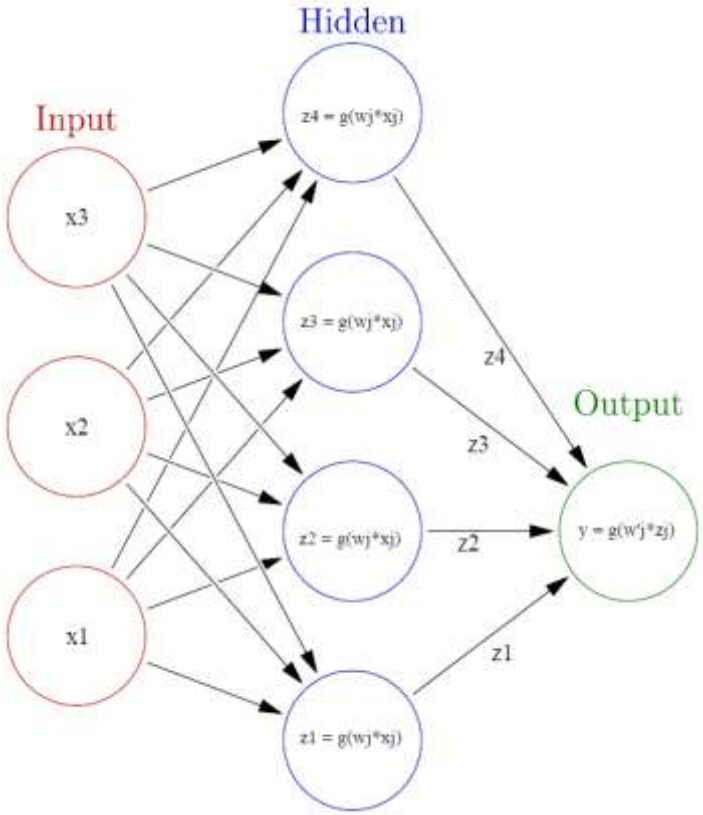
$$\hat{y}(x, w) = f\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

0,1,2,3,3,2,2,1,0,1,0,1,2,2,3

0,1,2,3 3

1,2,3,3 2

2,3,3,2 2



REDES NEURONALES RECURRENTES.

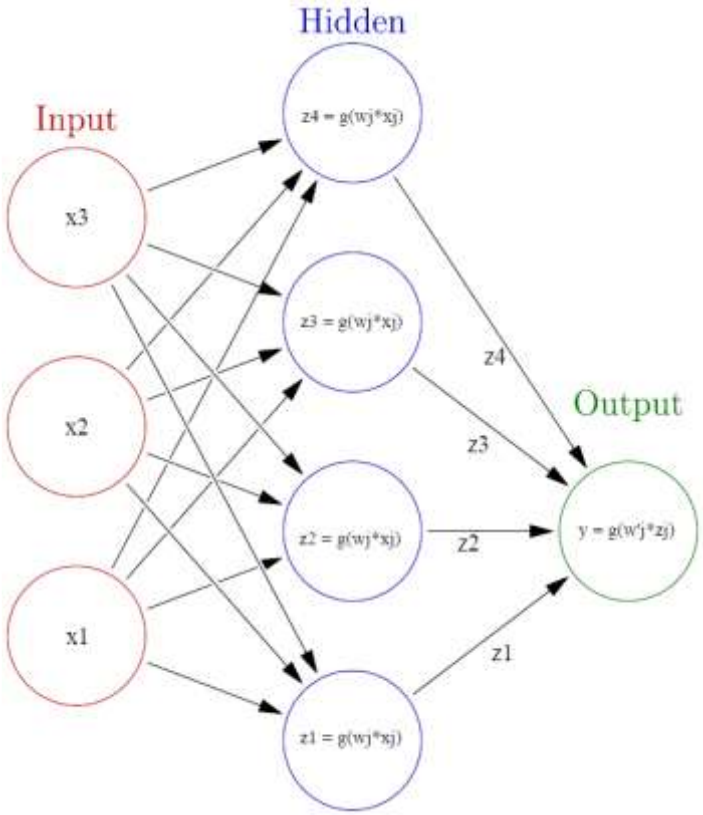
- MLP no es capaz de manejar entradas y salidas de diferente longitud, como por ejemplo en un traductor.

0,1,2,3,3,2,2,1,0,1,0,1,2,2,3

0,1,2,3 3

1,2,3,3 2

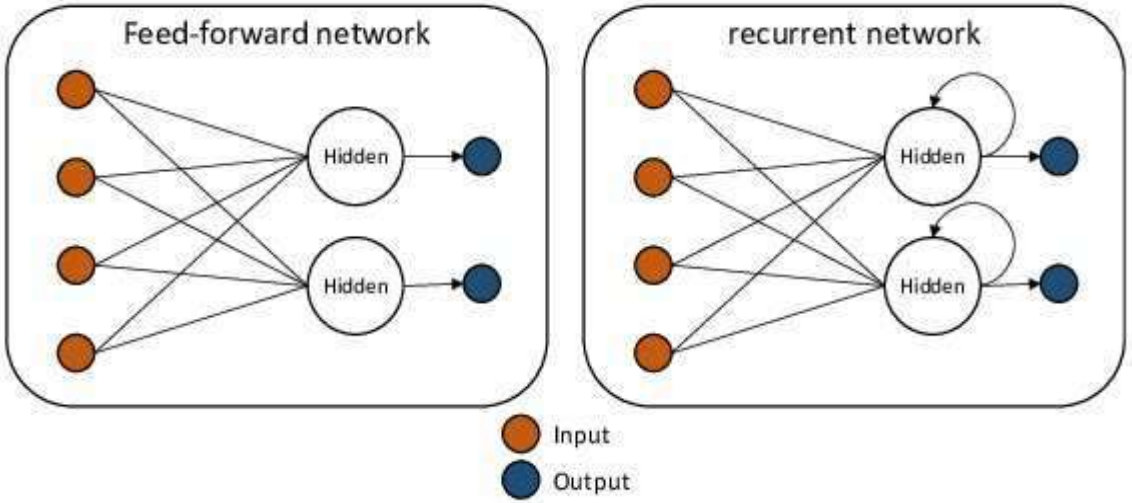
2,3,3,2 2



$$\hat{y}(x, w) = f\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

REDES NEURONALES RECURRENTES.

- La RNN ya no es una red "*feed forward*" sino que permite retroalimentarse.



REDES NEURONALES RECURRENTES.

- **MPL**

$$\hat{y}(x, w) = f\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right)$$

- **RNN.**

$$a^{(t)} = b + Wh^{t-1} + Ux^t \quad (1)$$

$$h^{(t)} = tanh(a^{(t)}) \quad (2)$$

$$o^{(t)} = c + Vh^t \quad (3)$$

$$y^{(t)} = f(o^t) \quad (4)$$

REDES NEURONALES RECURRENTES.

- donde b y c son vectores de términos independientes, y U, V, y W son respectivamente las matrices de pesos de conexión entre la capa de entrada y la capa oculta, entre la capa oculta y la salida, y entre la capa oculta.

- **RNN.**

$$a^{(t)} = b + Wh^{t-1} + Ux^t \quad (1)$$

$$h^{(t)} = tanh(a^{(t)}) \quad (2)$$

$$o^{(t)} = c + Vh^t \quad (3)$$

$$y^{(t)} = f(o^t) \quad (4)$$

REDES NEURONALES RECURRENTES.

- **RNN compacto**

$$h_t = f_h(b + Wh^{(t-1)} + UX^{(t)}) \quad (5)$$

$$\hat{y}_t = f_y(c + Vh^{(t)}) \quad (6)$$

- **RNN.**

$$a^{(t)} = b + Wh^{t-1} + Ux^t \quad (1)$$

$$h^{(t)} = tanh(a^{(t)}) \quad (2)$$

$$o^{(t)} = c + Vh^t \quad (3)$$

$$y^{(t)} = f(o^t) \quad (4)$$

REDES NEURONALES RECURRENTES.

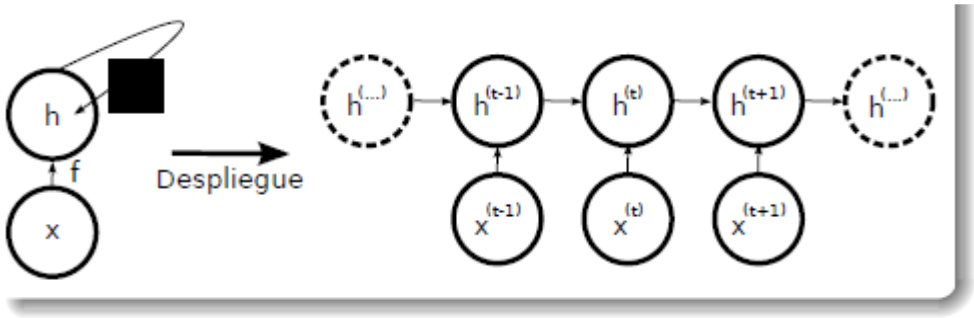
- RNN introduce lazos retroalimentados permitiendo que estos lazos sean condicionados por el contexto.
- Estos lazos permiten que la información aprendida por la red pase de un paso de tiempo de la red al siguiente.

RNN compacto

$$h_t = f_h(\mathrm{b} + \mathrm{W}h^{(t-1)} + \mathrm{U}X^{(t)}) \quad (7)$$

$$\hat{y}_t = f_y(\mathrm{c} + \mathrm{V}h^{(t)}) \quad (8)$$

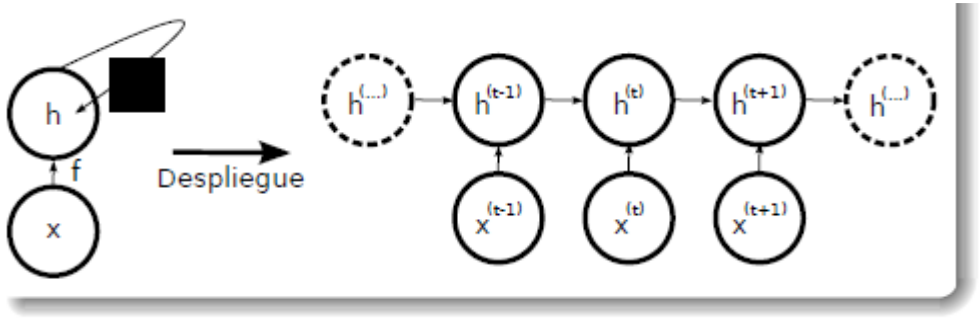
RNN visualización



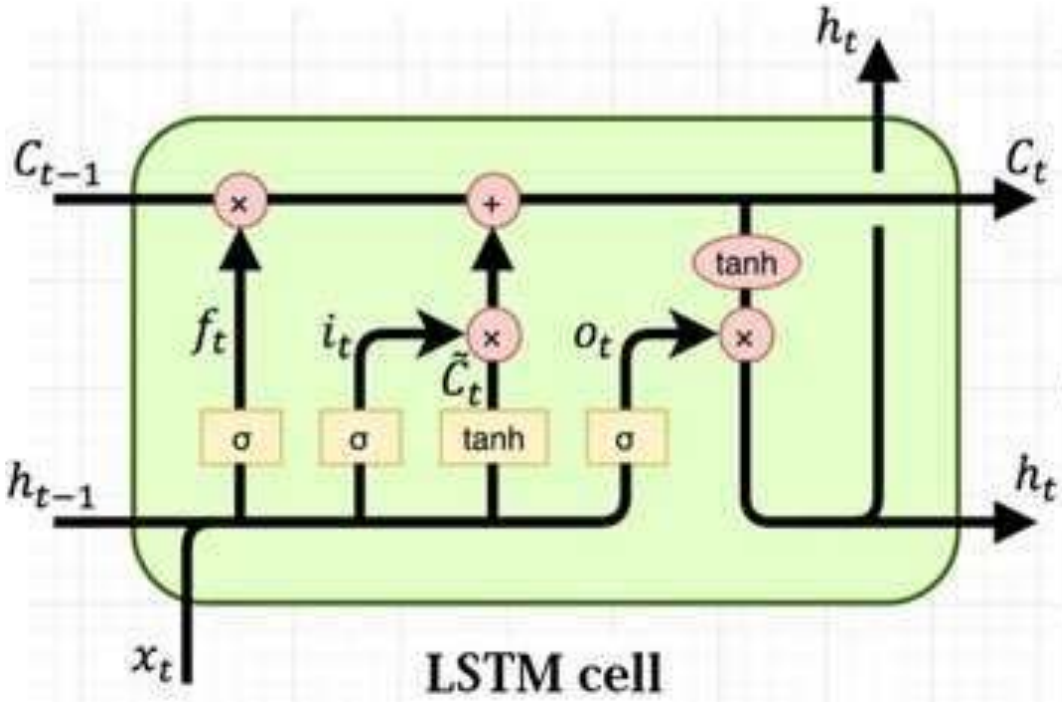
REDES NEURONALES RECURRENTES.

- El pronóstico de la siguiente palabra en un texto puede requerir de información reciente (contexto reciente) o de información de un contexto más lejano.
- Las RNN, en teoría, deben ser capaces de manejar estas dependencias de largo periodo. En realidad esta aproximación no fue frutífera más allá de experimentos muy pequeños.

RNN visualización



LSTM



LSTM cell

$$i_t = \sigma\left(x_t U^i + h_{t-1} W^i\right)$$

$$f_t = \sigma\left(x_t U^f + h_{t-1} W^f\right)$$

$$o_t = \sigma\left(x_t U^o + h_{t-1} W^o\right)$$

$$\tilde{C}_t = \tanh\left(x_t U^g + h_{t-1} W^g\right)$$

$$C_t = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t\right)$$

$$h_t = \tanh(C_t) * o_t$$

LSTM

- *Long Short-Term Memory* (LSTM) es un tipo de RNN.
- LSTM introduce múltiples lazos retroalimentados en el esquema RNN, tanto para aprender como para olvidar información, permitiendo que estos lazos sean condicionados por el contexto.
- LSTM han demostrado tener un gran rendimiento en múltiples problemas.

$$\begin{aligned} h_i &= \sigma(W_{hh}\, h_{i-1} + W_{hx}\, x_i) \\ \hat{y}_i &= W_{yh}\, h_i \end{aligned} \qquad (9)$$

donde x_i es el vector de entrada, h_i es el estado oculto, W_{hx} es la matriz de pesos que conecta la entrada con el estado oculto, W_{hh} es matriz de pesos que conecta las matrices de estados ocultos, σ es la función no lineal, W_{yh} es la matriz que conecta el estado oculto con la salida, e $\hat{y}$ es la predicción.

RESUMEN

- Se ha mostrado como funcionan las redes neuronales recurrentes, y sus diferencias con otras arquitecturas de aprendizaje profundo.
- Se mostrado en detalle la celda LSTM.

Código:

- Código ejemplo de red neuronal recurrente con celda LSTM en problema de pronóstico en series temporales:

RNN_LSTM_TS_ContMadrid.ipynb.