Introducción al cálculo numérico y ciencia de datos

Clase 3

Mg. Víctor Viana

Dr. Diego Passarella

Tacuarembó – abril de 2023

EDUCACION PERMANENTE
Universidad de la República

SEDE TACUAREMBÓ
CENUR NORESTE
UNIVERSIDAD DE LA REPÚBLICA URUGUAY

Agenda

- Redes Neuronales tradicionales
- Redes Profundas - Deep Learning

Aprendizaje Profundo

El aprendizaje profundo, es un sub-campo del aprendizaje automático [1], que utiliza redes neuronales artificiales inspiradas en el funcionamiento del cerebro humano para su aprendizaje.

[1] Dunn, T. 2020. Aprendizaje profundo. Salem Press Encyclopedia of Science.

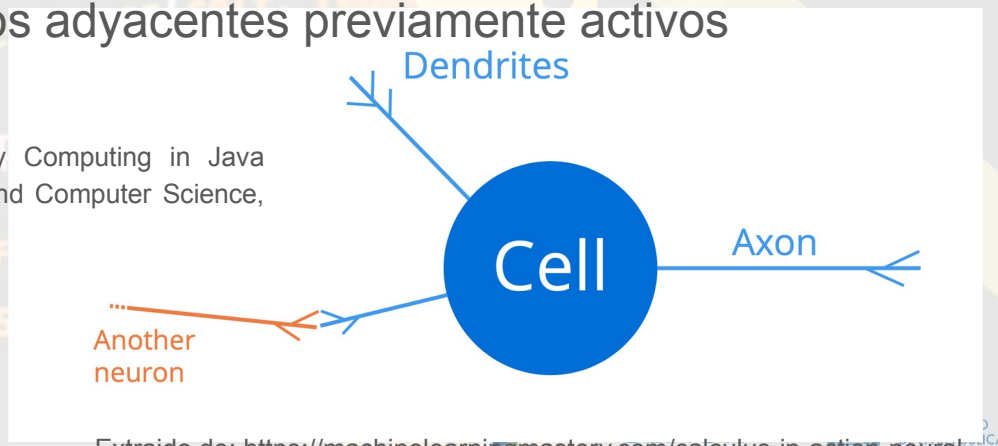
¿Qué es una red neural artificial?

- Una red neuronal artificial es un modelo de aprendizaje automático inspirado en la estructura y funcionamiento del cerebro humano.
- Consiste en un conjunto de nodos o unidades de procesamiento interconectados, organizados en capas.
- Cada unidad de procesamiento recibe una o varias entradas, las procesa y produce una salida.

¿Qué es una red neural artificial?(II)

Las redes neuronales artificiales [2] buscan emular el comportamiento de las neuronas en el cerebro humano, donde los grupos de nodos, o células, que se encuentran interconectados entre sí, producen una secuencia de activaciones, ya sea a través de estímulos del entorno -datos de entrada- o bien, de conexiones ponderadas de sus nodos adyacentes previamente activos

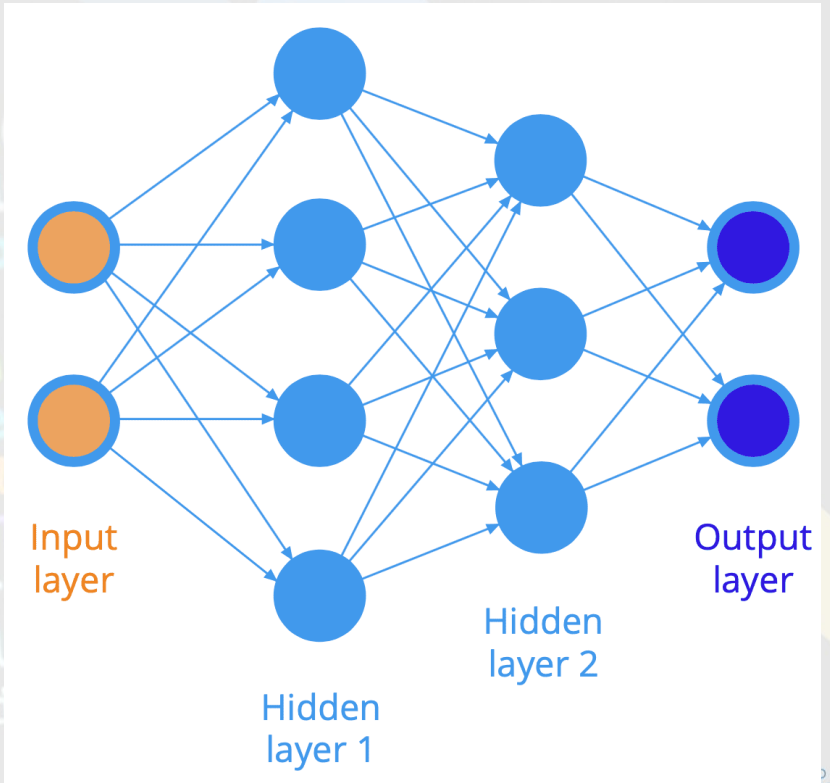
[2] Wang, SC. 2003. Artificial Neural Network. Interdisciplinary Computing in Java Programming. The Springer International Series in Engineering and Computer Science, vol 743. Springer, Boston, MA.



Extraido de: https://machinelearningmastery.com/calculus-in-action-neural-networks/

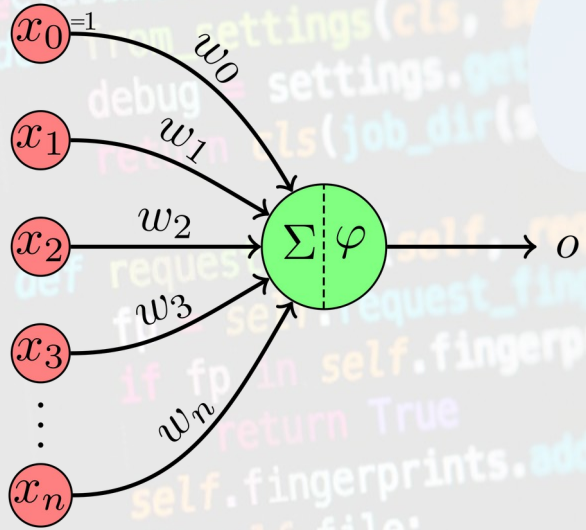
¿Qué es una red neural artificial?(III)

Este proceso de transformación y retransmisión de información entre las diferentes capas se da hasta llegar al final de la red, en donde se obtiene un resultado final que variará en función de la información brindada, el entrenamiento, y el objetivo para el cual ha sido creada la red.



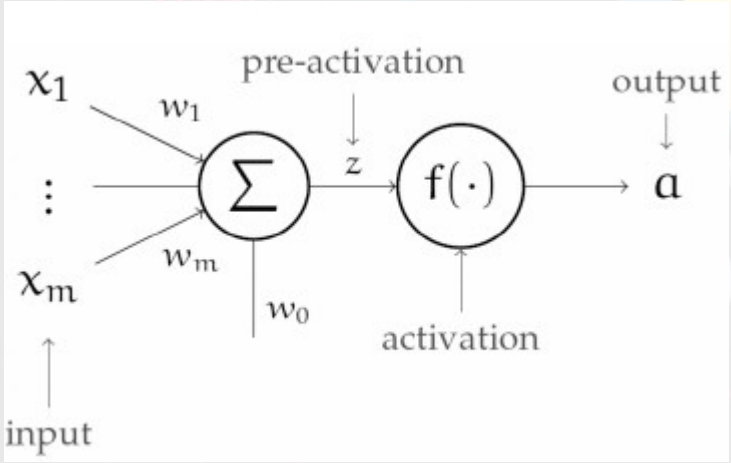
Extraido de: https://machinelearningmastery.com/calculus-in-action-neural-networks/

Redes Neuronales Tradicionales



Redes Neuronales Tradicionales

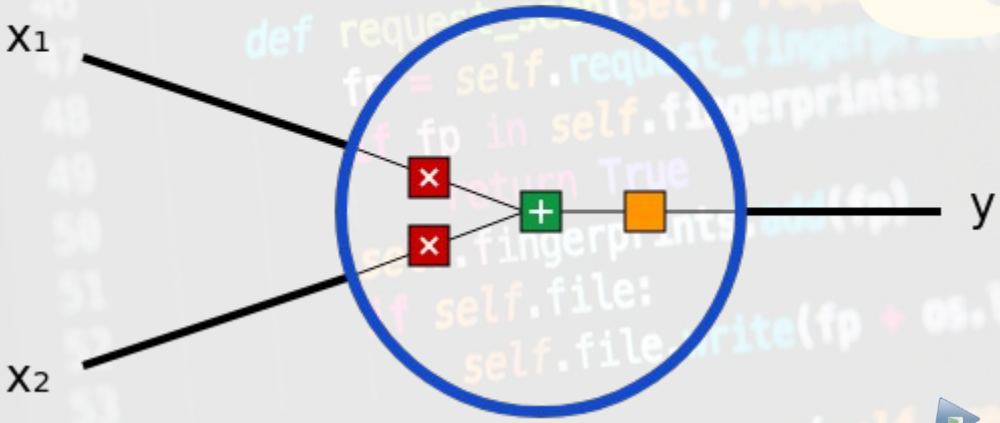
El nodo es una función no lineal de un vector de entrada $x_1, x_2, \ldots x_m$ a un valor único de salida **a**, parametrizado por un vector de pesos $w_1, w_2, \ldots w_m$, un sesgo ***z*** y una función de activación ***f*** que se encarga de introducir la no linealidad.



Extraido de Wang, SC. 2003. Artificial Neural Network. Interdisciplinary Computing in Java Programming. The Springer International Series in Engineering and Computer Science, vol 743. Springer, Boston, MA.

Ejemplo básico

Una neurona recibe datos de entrada, los procesa matemáticamente y produce un resultado. Este es el aspecto de una neurona de 2 entradas:



Ejemplo básico (II)

1) Cada entrada se multiplica por un peso:

$$x1 \rightarrow x1 * w1$$

$$x2 \rightarrow x2 * w2$$

2) A continuación, se suman todas las entradas ponderadas con un sesgo **b**:

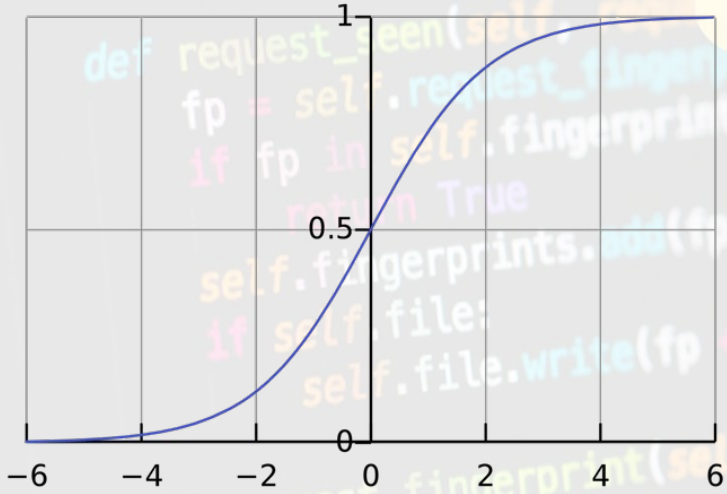
$$(x1 * w1)+(x2 * w2)+b$$

3) Finalmente, la suma se pasa por una función de activación **f**:

$$y=f(x1 * w1+x2 * w2+b)$$

Ejemplo básico (III)

La función de activación se utiliza para convertir una entrada no limitada en una salida que tenga una forma agradable y predecible. Una función de activación muy utilizada es la función sigmoide:



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

Ejemplo básico (III)

Supongamos que tenemos una neurona de 2 entradas que utiliza la función de activación sigmoidea y tiene los siguientes parámetros:

$$W=[0,1]$$

$$b=4$$

Ahora, vamos a dar a la neurona una entrada de x=[2,3]. Usaremos el producto punto para escribir las cosas de forma más concisa:

$$(w \cdot x)+b=((w1 * x1)+(w2 * x2))+b$$

$$=0 * 2+1 * 3+4 = 7$$

$$y=f(w \cdot x+b)=f(7)=0.999y=f(w \cdot x+b)=f(7) = 0.999$$

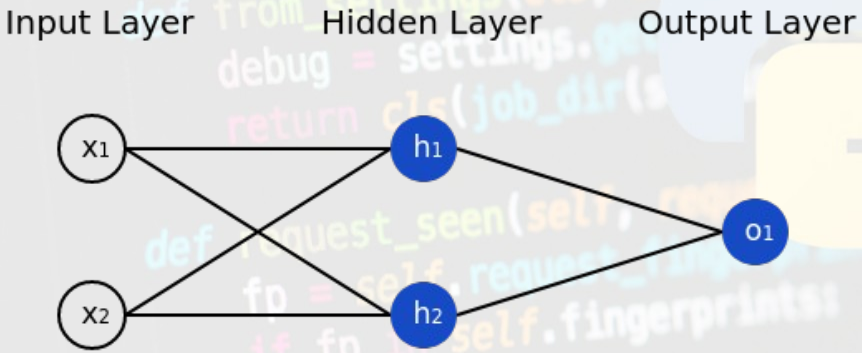
La neurona produce 0,999 dadas las entradas x=[2,3]. Este proceso de pasar entradas hacia adelante para obtener una salida se conoce como **feedforward**.

Ejemplo básico (IV)

Ver ejemplo en el Notebook: icncd_clase3_ejemplo_neurona_1.ipynb

Combinación de neuronas en una red neuronal

Una red neuronal no es más que un montón de neuronas conectadas entre sí.



Esta red tiene 2 entradas, una capa oculta con 2 neuronas (h_1 y h_2) y una capa de salida con 1 neurona (o_1). Observar que las entradas de o1 son las salidas de h_1 y h_2. Una capa oculta es cualquier capa entre la capa de entrada (primera) y la capa de salida (última). Puede haber varias capas ocultas.

Otro ejemplo: Feedforward

- Utilicemos la red representada arriba y supongamos que todas las neuronas tienen los mismos pesos w=[0,1], el mismo sesgo b=0 y la misma función de activación sigmoide.
- Sean h_1, h_2, o_1 las salidas de las neuronas que representan.
- ¿Qué ocurre si pasamos la entrada x=[2,3]?

Otro ejemplo: Feedforward (II)

$$h_1 = h_2 = f(w \cdot x + b)$$

$$= f((0 * 2) + (1 * 3) + 0)$$

$$= f(3)$$

$$= 0.9526$$

$$o_1 = f(w \cdot [h_1, h_2] + b)$$

$$= f((0 * h_1) + (1 * h_2) + 0)$$

$$= f(0.9526)$$

$$= 0.7216$$

Otro ejemplo: Feedforward (III)

- La salida de la red neuronal para la entrada x=[2,3] es 0,7216.
- Una red neuronal puede tener cualquier número de capas con cualquier número de neuronas en esas capas.
- La idea básica sigue siendo la misma: hacer pasar la(s) entrada(s) por las neuronas de la red para obtener la(s) salida(s) al final.

Otro ejemplo: Feedforward (IV)

Ver ejemplo en el Notebook: icncd_clase3_ejemplo_neurona_1.ipynb

Entrenamiento de una red neuronal

Supongamos que tenemos las siguientes medidas:

Nombre	Peso (kg)	Altura (cm)	Genero
Alice	60	165	F
Bob	73	183	M
Charlie	69	178	M
Diana	54	152	F

Entrenamiento de una red neuronal

Representaremos Masculino con un 0 y Femenino con un 1, y desplazamos el peso y la altura respecto a un mínimo (61 kg y 167 cm):

Nombre	Peso (kg)	Altura (cm)	Genero
Alice	-1	-2	1
Bob	12	16	0
Charlie	8	11	0
Diana	-7	-15	1

Pérdidas

- Antes de entrenar nuestra red, necesitamos una forma de cuantificar lo "bien" que lo está haciendo para que pueda intentar hacerlo "mejor". Para eso está la pérdida.
- Utilizaremos el error cuadrático medio (MSE):

$$MSE = \frac{1}{n} \sum \left(y_{true} - y_{pred} \right)^2$$

Pérdidas(II)

- ***n*** es el número de muestras, que es 4 (Alice, Bob, Charlie, Diana).
- ***y*** representa la variable que se predice, que es Género.
- y_{true} es el valor verdadero de la variable (la "respuesta correcta").
- y_{pred} es el valor predicho de la variable.
- $(y_{true}-y_{pred})^2$ es el error al cuadrado. Nuestra función de pérdida es simplemente la media de todos los errores al cuadrado (de ahí el nombre de error cuadrático medio)
- **Mejores predicciones = Menores pérdidas.**
- **Entrenar una red = intentar minimizar sus pérdidas.**

Entrenando la red neuronal (continuación)

- Ahora tenemos un objetivo claro: minimizar la pérdida de la red neuronal. Sabemos que podemos cambiar los pesos y sesgos de la red para influir en sus predicciones, pero ¿cómo lo hacemos para reducir las pérdidas?
- Esta sección utiliza un poco de cálculo multivariable.
- Para simplificar, vamos a suponer que sólo tenemos un registro en nuestro conjunto de datos

Entrenando la red neuronal (continuación, II)

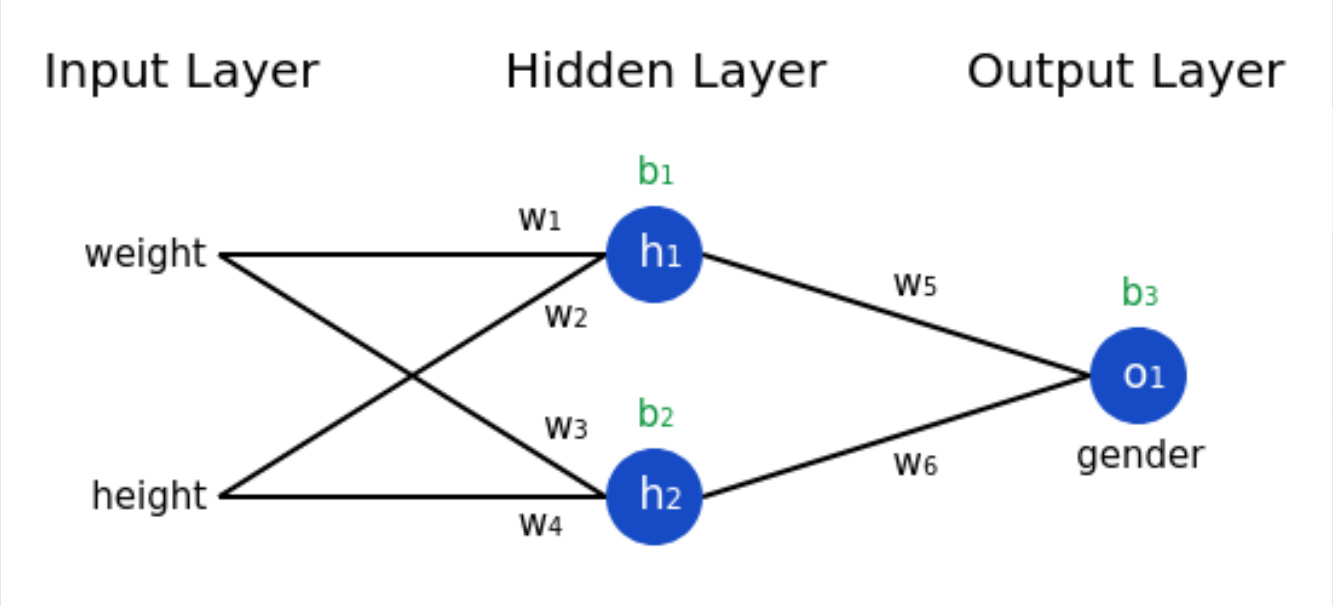
Entonces la pérdida de error cuadrático medio es sólo el error cuadrático:

$$MSE = \frac{1}{1}\sum \left(y_{true} - y_{pred} \right)^2 = \left(y_{true} - y_{pred} \right)^2$$

Otra forma de ver las pérdidas

Podemos escribir la pérdida como una función multivariable:

L(w1,w2,w3,w4,w5,w6,b1,b2,b3)



Otra forma de ver las pérdidas(II)

- Imaginemos que queremos modificar w_1.
- ¿Cómo cambiaría la pérdida L si cambiáramos w_1?
- Derivamos parcialmente L respecto a w_1.

$$\frac{dL}{dw_1} = \frac{dL}{dy_{\text{pred}}} \frac{dy_{\text{pred}}}{dw_1}$$

(regla de la cadena)

Otra forma de ver las pérdidas(III)

Sabiendo que L = MSE

$$\frac{dL}{dy_{pred}} = \frac{d\left(y_{\text{true}} - y_{pred}\right)^2}{dy_{\text{pred}}} = -2\left(y_{\text{true}} - y_{\text{pred}}\right)$$

Otra forma de ver las pérdidas(IV)

$$y_{\text{pred}} = o_1 = f\left(w_5 h_1 + w_6 h_2 + b_3\right)$$

$$\frac{dy_{pred}}{dw_1} = \frac{dy_{pred}}{dh_1}\frac{dh_1}{dw_1}$$

$$\frac{dy_{pred}}{dh_1} = w_5 \cdot f'\left(w_5 h_1 + w_6 h_2 + b_3\right)$$

$$h_1 = f(w_1 x_1 + w_2 x_2 + b_1)$$

$$\frac{dh_1}{dw_1} = x_1 + f'\left(w_1 x_1 + w_2 x_2 + b_1\right)$$

Otra forma de ver las pérdidas(V)

Este sistema de cálculo de derivadas parciales trabajando hacia atrás se conoce como retropropagación (**backpropagation**).

$$\frac{dL}{\partial w_1} = \frac{dL}{dy_{\text{pred}}} * \frac{dy_{\text{pred}}}{dh_1} * \frac{dh_1}{dw_1}$$

Probando con Alice

$$
\begin{aligned}
h_1 &= f(w_1 x_1 + w_2 x_2 + b_1) \\
&= f(-2 + -1 + 0) \\
&= 0.0474
\end{aligned}
$$

$$h_2 = f(w_3 x_1 + w_4 x_2 + b_2) = 0.0474$$

$$
\begin{aligned}
o_1 &= f(w_5 h_1 + w_6 h_2 + b_3) \\
&= f(0.0474 + 0.0474 + 0) \\
&= 0.524
\end{aligned}
$$

El resultado de la red es $y_{pred} = 0{,}524$

(¿?)

Probando con Alice(II)

$$\frac{dL}{dw_1} = \frac{dL}{dy_{\text{pred}}} * \frac{dy_{\text{pred}}}{dh_1} * \frac{dh_1}{dw_1}$$

$$\frac{dL}{dy_{\text{pred}}} = -2\,(1 - y_{\text{pred}}\,)$$

$$= -2(1 - 0.524)$$

$$= -0.952$$

$$\frac{dy_{\text{pred}}}{dh_1} = w_5 * f'\,(w_5 h_1 + w_6 h_2 + b_3)$$

$$= 1 * f'(0.0474 + 0.0474 + 0)$$

$$= f(0.0948) * (1 - f(0.0948))$$

$$= 0.249$$

$$\frac{dh_1}{dw_1} = x_1 * f'\,(w_1 x_1 + w_2 x_2 + b_1)$$

$$= -2 * f'(-2 + -1 + 0)$$

$$= -2 * f(-3) * (1 - f(-3))$$

$$= -0.0904$$

$$\frac{dL}{dw_1} = -0.952 * 0.249 * -0.0904$$

$$= 0.0214$$

Esto nos dice que si aumentamos w_1, L disminuiría levemente como resultado.

Entrenamiento: Descenso Gradiente Estocástico

- Ya tenemos todas las herramientas necesarias para entrenar una red neuronal
- Utilizaremos un algoritmo de optimización llamado descenso gradiente (GD) que nos indica cómo cambiar nuestros pesos y sesgos para minimizar las pérdidas.
- Básicamente es esta ecuación de actualización:

$$w_1 \leftarrow w_1 - \eta \frac{dL}{dw_1}$$

Entrenamiento: Descenso Gradiente Estocástico(II)

- η es una constante llamada tasa de aprendizaje que controla lo rápido que entrenamos. Todo lo que estamos haciendo es restar $\eta dL/dw_1$ de w_1:

- Si dL/dw_1 es positivo, w_1 disminuirá, lo que hace que L disminuya.

- Si dL/dw_1 es negativo, w_1 aumentará, lo que hace que L disminuya.

Si hacemos esto para cada peso y sesgo de la red, la pérdida disminuirá lentamente y nuestra red mejorará.

Entrenamiento: Descenso Gradiente Estocástico (III)

Nuestro proceso de entrenamiento será así:

1) Elegir una muestra de nuestro conjunto de datos. Esto es lo que hace que el descenso por gradiente estocástico - que sólo operan en una muestra a la vez.

2) Calcular todas las derivadas parciales de la pérdida con respecto a los pesos o sesgos.

3) Utilice la ecuación de actualización para actualizar cada peso y sesgo.

4) Volver al paso 1.

Ejemplo completo

Ver ejemplo en el Notebook: [icncd_clase3_ejemplo_neurona_1.ipynb]

Redes Profundas y Deep Learning

Retropropagación (Backpropagation)

Durante el entrenamiento, el modelo aprende propagando el error a los nodos y actualizando los parámetros (pesos y sesgos) para minimizar la pérdida.

Descenso gradual (Gradient Descent)

Algoritmo de optimización utilizado para entrenar redes neuronales que encuentra el mínimo local de la función de pérdida dando pasos repetidos en la dirección del descenso más pronunciado.

¿Qué es el Deep Learning?

Por arriba

- Redes neuronales de muchas capas

Mas en detalle

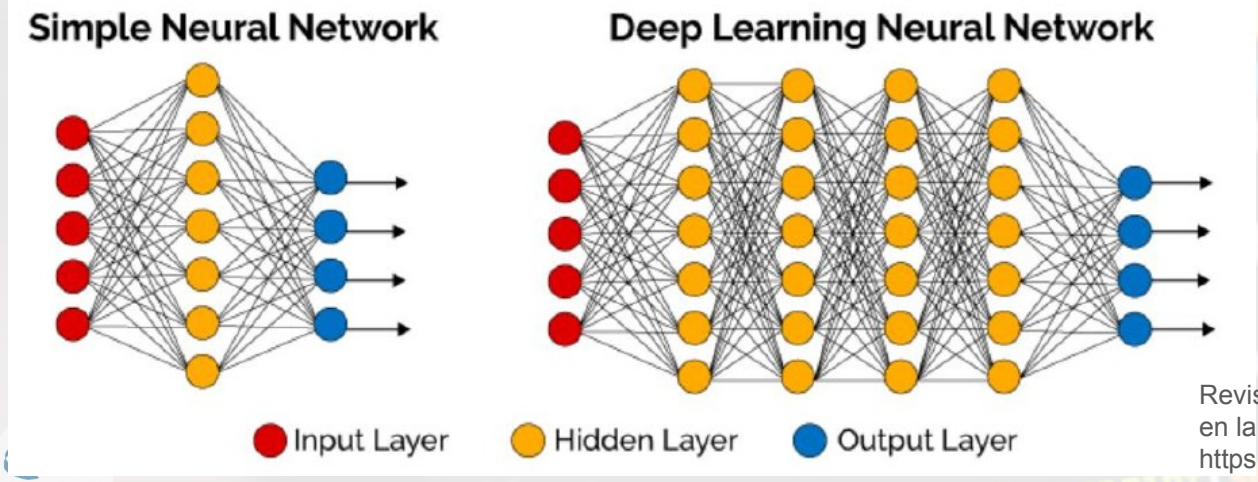
- Nuevas y mejores arquitecturas
- Nuevos y mejores algoritmos
- Más y mejor hardware
- Muchos más y mejores datos

¿Qué es el Deep Learning? (II)

El aprendizaje automático consiste en la asignación de pesos que hagan que la red neuronal muestre el comportamiento esperado y en el caso del **aprendizaje profundo** el concepto está estrechamente relacionado con la cantidad de capas escondidas con la cual se diseña la red.

¿Qué es el Deep Learning? (III)

Una red se dice que es amplia y superficial cuando posee pocas capas escondidas compuestas de una gran cantidad de nodos, contrariamente, una red profunda hace alusión a una gran cantidad de capas que pueden estar conformadas con muchas o pocas unidades.



Revista ECYS. Deep Learning y su increíble impacto en la realidad conocida. [En línea] 2019. https://revistaecys.github.io/13Edicion/03_ggiron.html.

¿Qué es el Deep Learning? (IV)

- El aprendizaje profundo permite que los modelos compuestos por varias capas aprendan representaciones de datos con múltiples niveles de abstracción, permitiendo representar una función no lineal mucho más compleja brindando resultados sorprendentes.

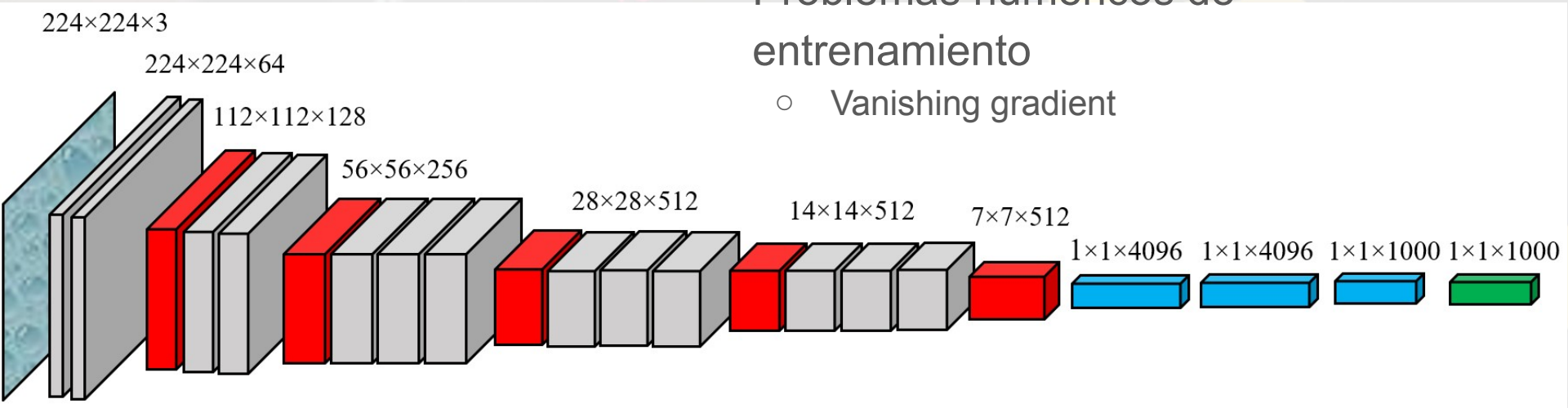
- En contrapartida, la complejidad de la red también la hace más difícil de manejar e incluso entender, se vuelve primordial alimentarla con una gran cantidad de datos y su entrenamiento puede insumir mucho tiempo y poder de procesamiento debido a la cantidad de cálculos realizados.

Muchas capas

- Antes: 2 o 3 capas máx
- Ahora: decenas de capas

Por qué antes no se podía?

- Más capas, muchos más parámetros
- Problemas numéricos de entrenamiento
  - Vanishing gradient



Otro ejemplo

Ver ejemplo en el Notebook: icncd_clase3_ejemploRN_diabetes.ipynb

Nuevas arquitecturas

Convolutional Networks

- Primeras capas son filtros repetidos
- Muy pocos parámetros
- Representación multiescala

Redes neuronales convolucionales

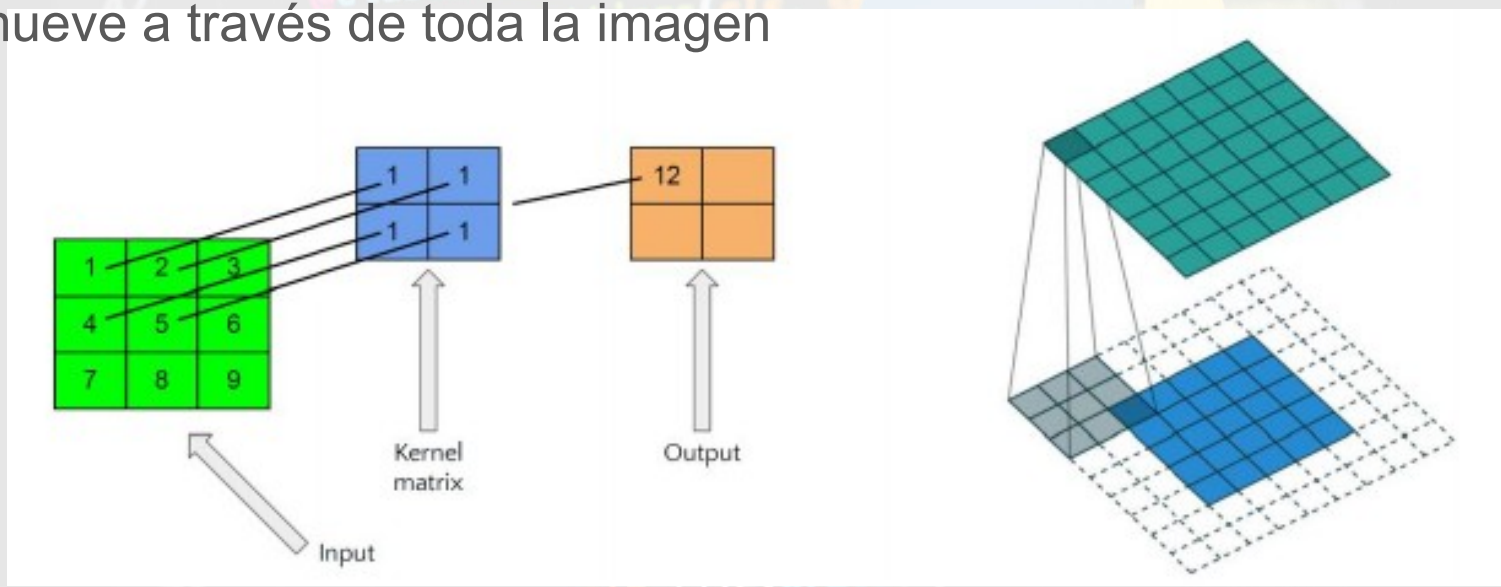
- Convolutional Neural Networks (CNN) se denominan de esa manera debido a que reemplazan la multiplicación de matrices básica por una operación de convolución en alguna de sus capas.
- Convolución es una operación matemática que se puede entender como la transformación de dos funciones en una sola.
- Una imagen se describe como una matriz de píxeles, en donde cada uno de ellos es representado por tres números enteros que especifican el nivel de intensidad de los colores rojo, verde y azul.
- Un banco de filtros compone una capa de la red neuronal, y un filtro es una función que toma como input un conjunto de valores de píxeles próximos espacialmente entre sí y permite detectar la presencia de algún patrón en los datos.

Redes neuronales convolucionales(II)

Para el caso de las imágenes digitalizadas, un filtro es una matriz la cual es movida a través de toda la imagen, obteniendo el producto interno entre los valores del filtro y los valores de los píxeles en cada paso, agregándolos y finalmente obteniendo una nueva imagen “filtrada”.

Redes neuronales convolucionales(III)

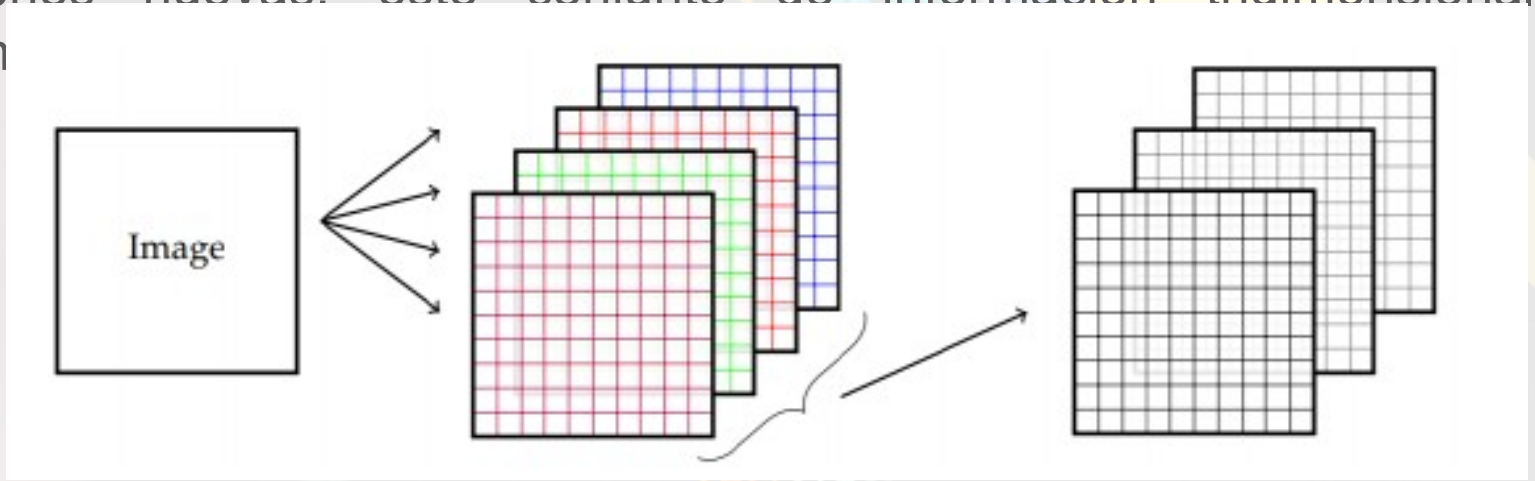
Operación de convolución. Ejemplo de un filtro aplicado a una matriz, el mismo se mueve a través de toda la imagen



Global Startup Labs, MIT. Day 12 Intro to Computer Vision. 2020.

Redes neuronales convolucionales(IV)

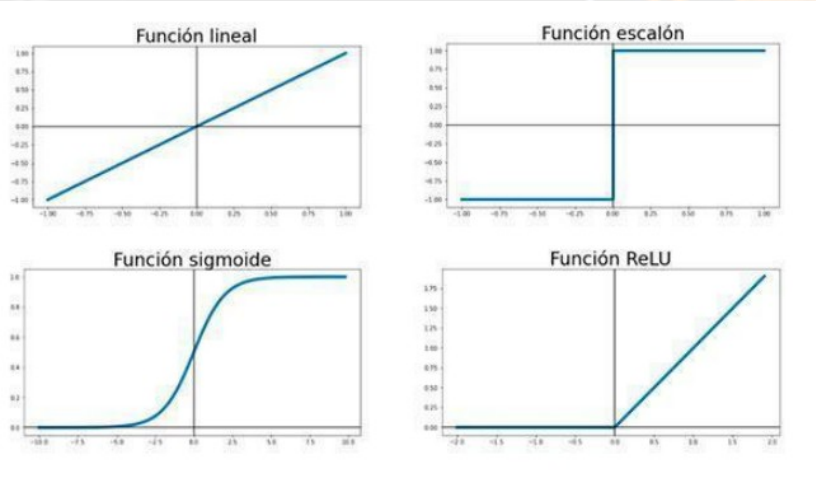
Al aplicar un banco de filtros a una misma imagen, se obtiene un nuevo conjunto de imágenes, es decir, si se tienen k filtros, el resultado será k imágenes nuevas, este conjunto de información tridimensional se denon



Introduction to Machine Learning MIT 6.036 [aut. libro] MIT. Capítulo 8 Neural Networks.

Redes neuronales convolucionales (V)

- En una capa de convolución cada filtro se puede ver como un nodo y los elementos de cada filtro individual son los “pesos” de la red que serán entrenados utilizando descenso por gradiente, detallado posteriormente.
- Por último, luego de cada capa de filtros generalmente se encuentra una capa de activación utilizando, por ejemplo, la función ReLu (Rectified Linear Unit).

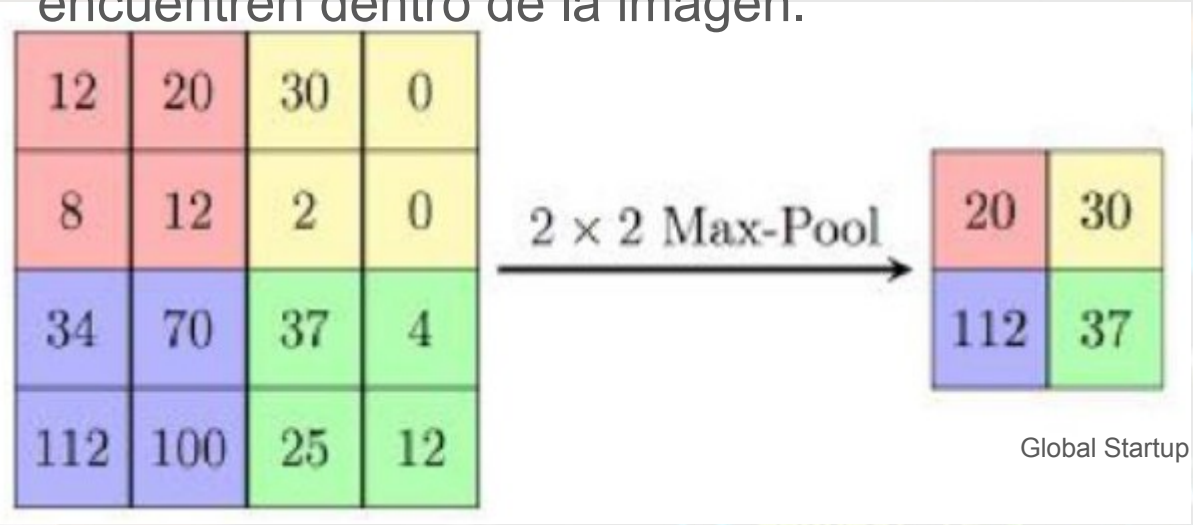


Redes neuronales convolucionales(VI)

- Las capas de reducción, universalmente conocidas como max-pooling o average-pooling son capas en donde se aplica un filtro a la imagen con un paso > 1, generando nuevas imágenes reducidas en tamaño.
- La gran diferencia entre esta capa y la capa de convolución es que la misma no tiene pesos a entrenar, por lo que se podría ver como una capa funcional.
- En la capa max-pooling, el resultado es el máximo valor de la parte de la imagen recubierta por el filtro.
- Mientras que en la capa average-pooling, el resultado es el promedio de los valores de la imagen recubiertos por el filtro.

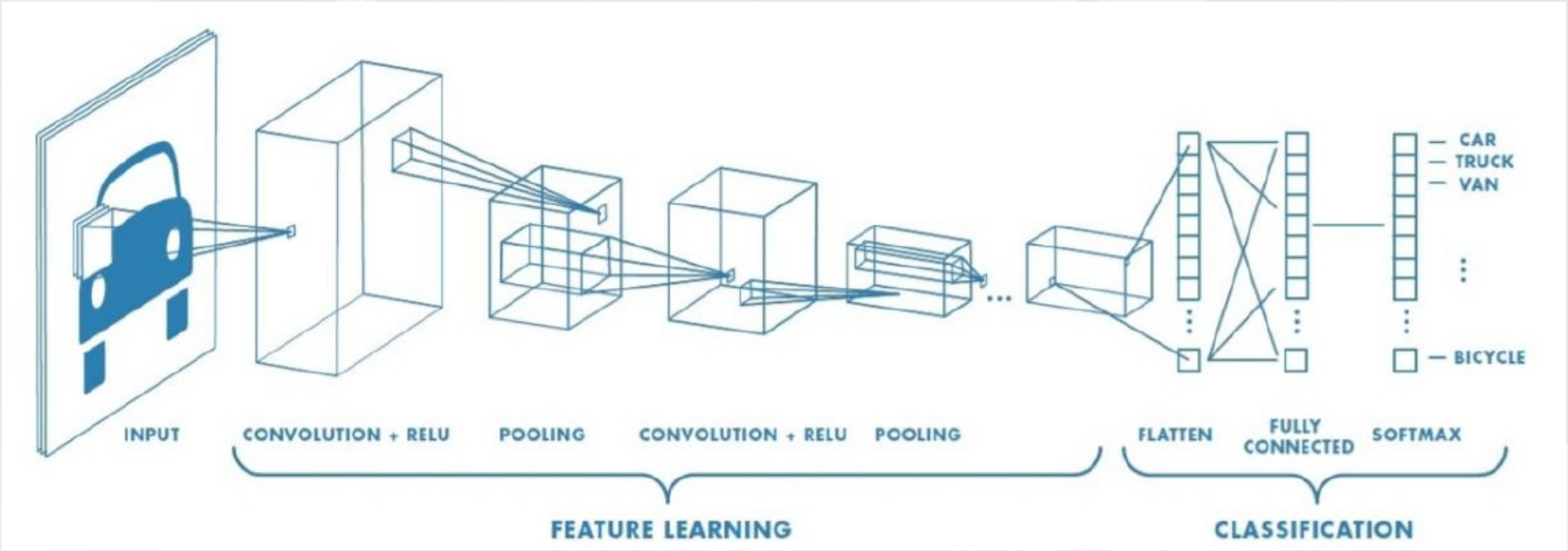
Redes neuronales convolucionales(VI)

El objetivo principal de estas capas es reducir el tamaño de los mapas de características y perder el rastro de la localización de los patrones, consiguiendo de esta manera que los filtros aprendan a extraer características independientes a la posición que se encuentren dentro de la imagen.



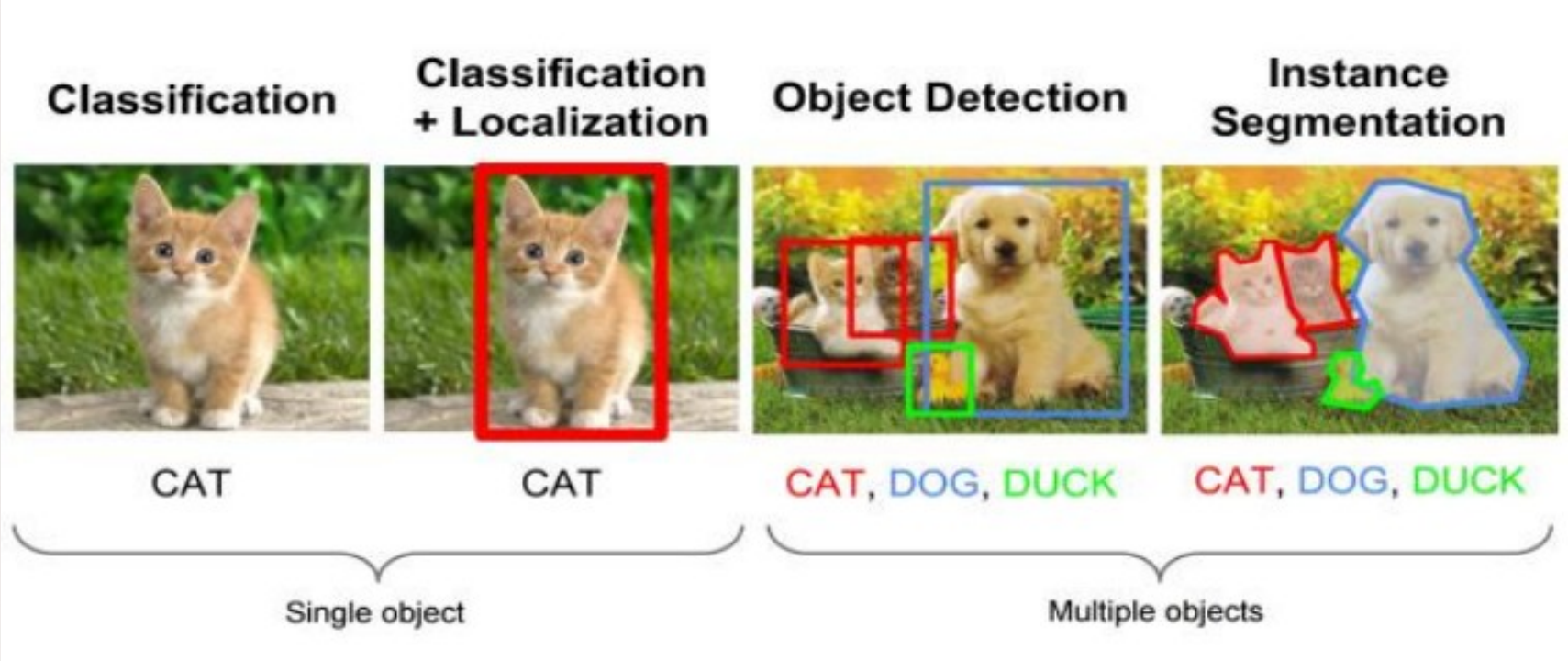
Global Startup Labs, MIT. Day 12 Intro to Computer Vision. 2020.

Estructura típica de una red neural convolucional



Global Startup Labs, MIT. Day 12 Intro to Computer Vision. 2020.

Aplicaciones



Global Startup Labs, MIT. Day 12 Intro to Computer Vision. 2020.