

ACÀMICA

¡Bienvenidos/as a Data Science!



Agenda

¿Cómo anduvieron?

Repaso: Redes Neuronales

Hands-On

Break

Terminología (extra) de Redes Neuronales

Actividad: Kahoot

Cierre



¿Cómo anduvieron?

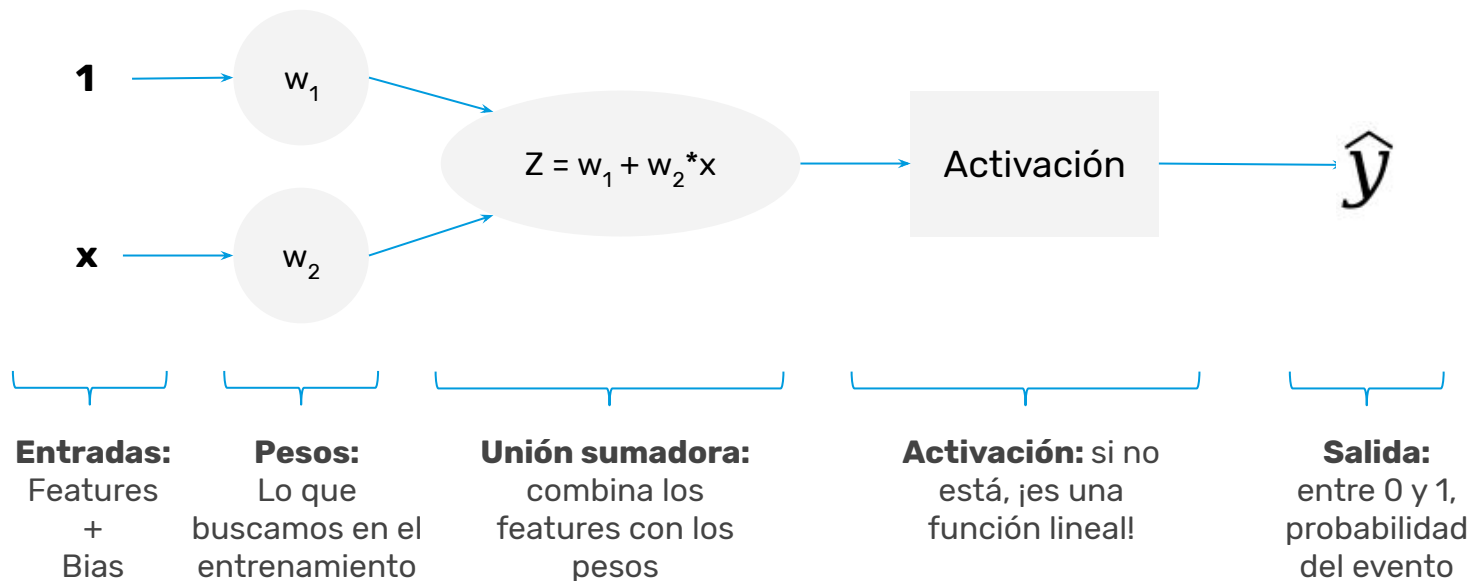


Repaso: Redes Neuronales



Perceptrón con una variable

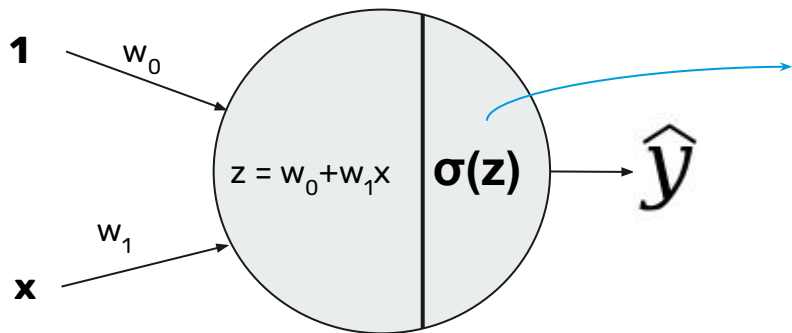
Necesitamos algo que, dado los features, devuelva probabilidades.
Las probabilidades deben estar entre 0 y 1



Perceptrón con una variable

Necesitamos algo que, dado los features, devuelva probabilidades.
Las probabilidades deben estar entre 0 y 1

Otra representación



Activación:

- Sin la activación, es una función lineal
- Necesitamos introducir algo que *sature* la entrada en 0 o en 1 dependiendo del resultado de la unión sumadora

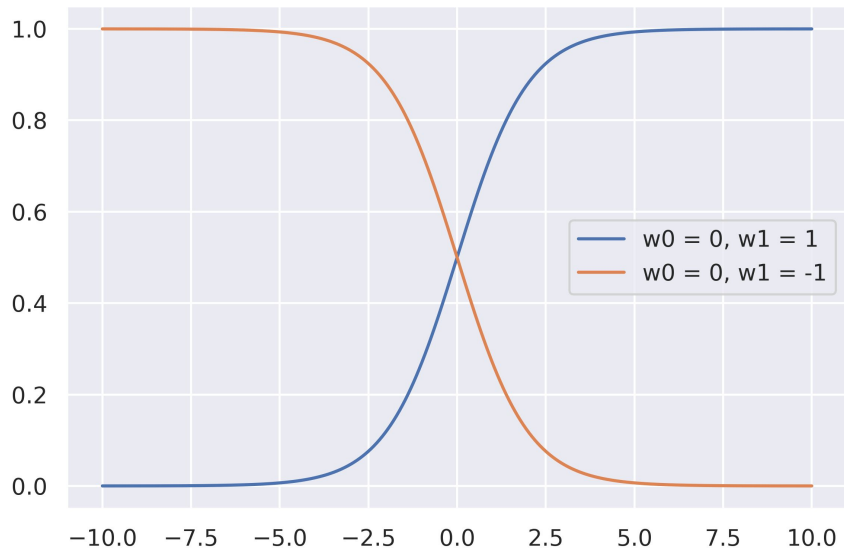
Función Logística / Sigmoide

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

↓

$$z = w_0 + w_1 x$$

$$y(x) = \frac{1}{1 + e^{-(w_0 + w_1 x)}}$$



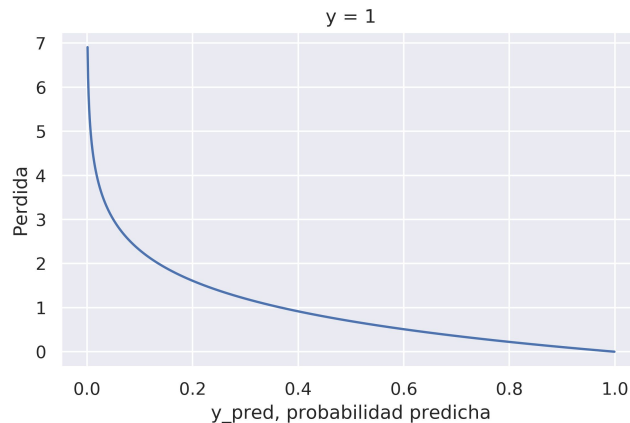
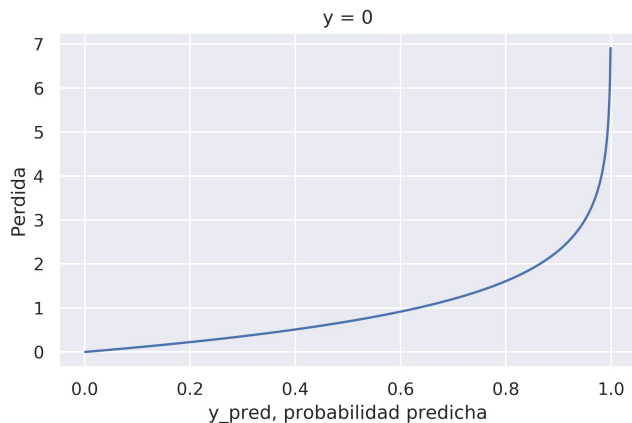
Entropía cruzada (cross-entropy)

Necesitamos una función de pérdida entre una etiqueta (y) y la probabilidad de pertenecer o no a esa etiqueta. \hat{y}

Caso binario: etiquetas $y = 0$ y 1 .

$$L(\hat{y}, y) = -y * \log(\hat{y}) - (1 - y) * \log(1 - \hat{y})$$

Pérdida para una instancia



Entropía cruzada (cross-entropy)

Necesitamos una función de pérdida entre una etiqueta (y) y la probabilidad de pertenecer o no a esa etiqueta. \hat{y}

Caso binario: etiquetas $y = 0$ y 1 .

$$L(\hat{y}, y) = -y * \log(\hat{y}) - (1 - y) * \log(1 - \hat{y})$$

Pérdida para una instancia

Entropía cruzada (cross-entropy)

Necesitamos una función de pérdida entre una etiqueta (y) y la probabilidad de pertenecer o no a esa etiqueta. \hat{y}

Caso binario: etiquetas $y = 0$ y 1 .

$$L(\hat{y}, y) = -y * \log(\hat{y}) - (1 - y) * \log(1 - \hat{y})$$

Pérdida para una instancia

$$J(\overline{W}) = \frac{1}{n} \sum_{i=0}^{n-1} L(\hat{y}^{(i)}, y^{(i)})$$

Costo para todas las instancias

Entropía cruzada (cross-entropy)

Necesitamos una función de pérdida entre una etiqueta (y) y la probabilidad de pertenecer o no a esa etiqueta. \hat{y}

Caso binario: etiquetas $y = 0$ y 1 .

$$L(\hat{y}, y) = -y * \log(\hat{y}) - (1 - y) * \log(1 - \hat{y})$$

Pérdida para una instancia

$$J(\overline{W}) = \frac{1}{n} \sum_{i=0}^{n-1} L(\hat{y}^{(i)}, y^{(i)})$$

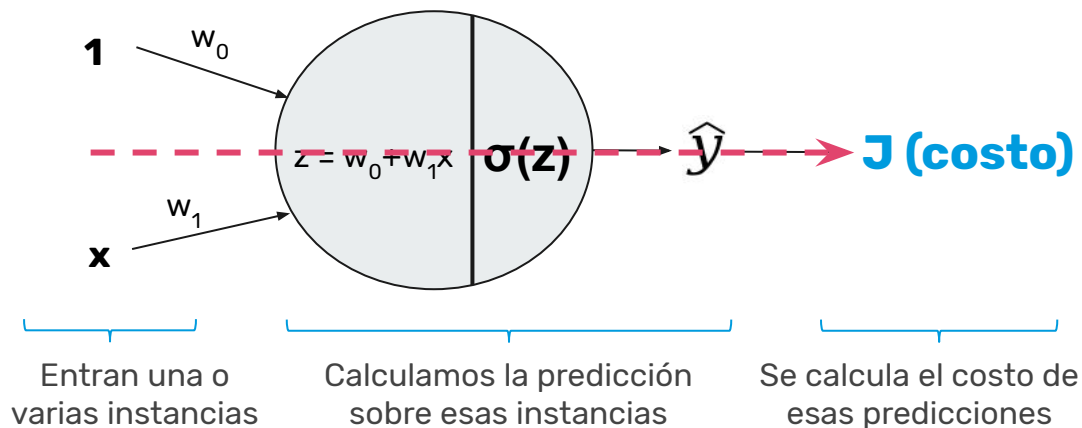
Costo para todas las instancias

$$J(w_0, w_1) = \frac{1}{n} \sum_{i=0}^{n-1} L(\hat{y}^{(i)}, y^{(i)})$$

Costo para todas las instancias, caso 1D

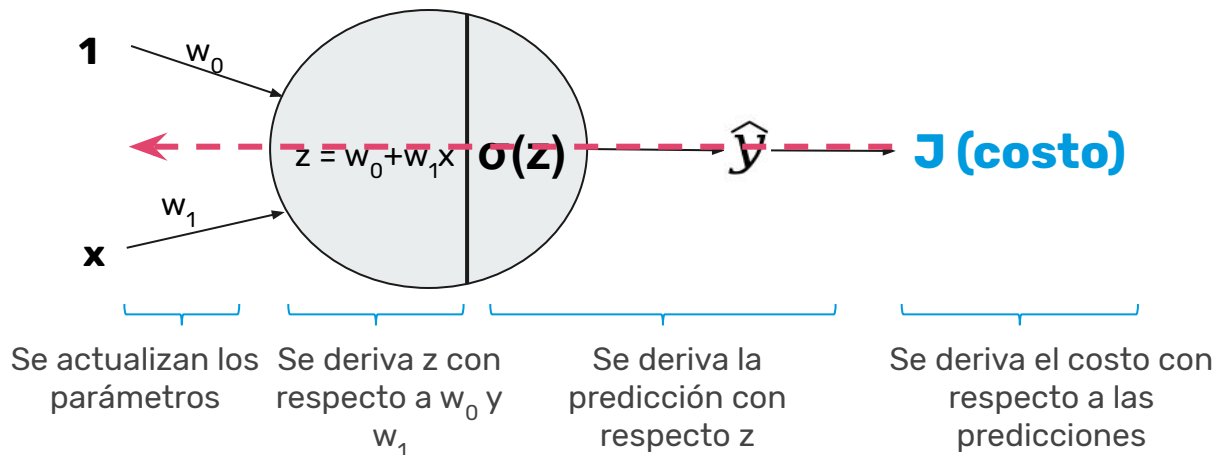
1. Descenso por gradiente calcula la derivada/gradiente del costo y con eso actualiza los parámetros. Este proceso lo va a hacer muchas veces hasta llegar al mínimo.
2. En cada una de esas iteraciones, tiene que calcular el costo. El costo depende de las instancias de entrenamiento y de los parámetros que tengamos hasta ese momento.

Calcular el costo con las instancias de entrenamiento es lo que se conoce como **Forward Propagation**.



1. Con el costo calculado, queremos actualizar los valores de los parámetros según la regla vista en la clase anterior.
2. Para eso, tenemos que derivar el costo y propagar esa derivada hacia atrás, hasta llegar a los parámetros w_0 y w_1 .

Calcular las derivadas y actualizar los parámetros “hacia atrás” se conoce como **Backpropagation**.



Perceptrón Multicapa

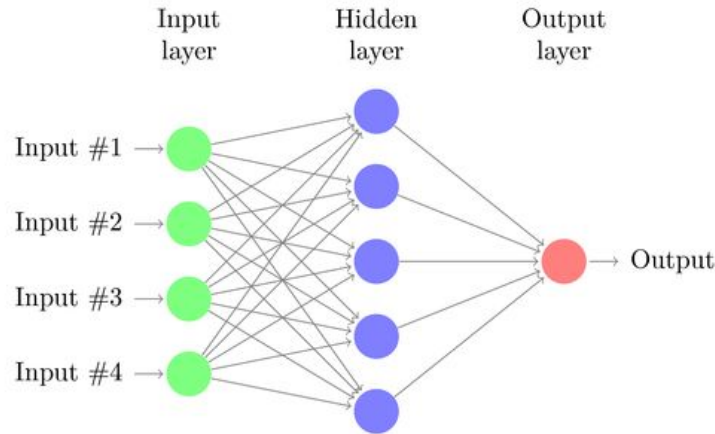
Ampliando el Perceptrón

Problema con el Perceptrón:
solo encuentra fronteras lineales.



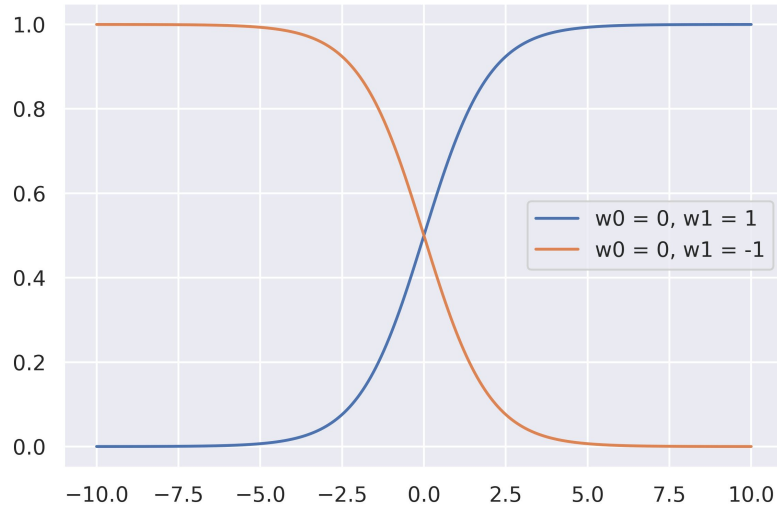
Ampliando el Perceptrón

Solución: Perceptrón Multicapa



- Cada neurona tiene sus propios pesos/parámetros. En aplicaciones comunes suelen ser desde miles a millones de parámetros para toda la red.
- **Deep Learning es** encontrar esos pesos de manera eficiente, bajo la condición de realizar correctamente una tarea objetivo.

Perceptrón Multicapa • Funciones de activación



1. Sigmoide/logística

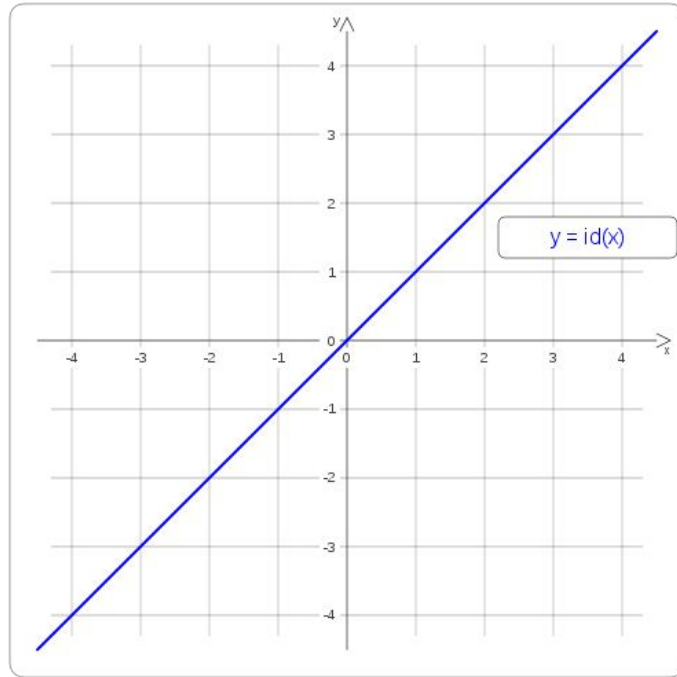
2. Identidad: $f(x) = x$

3. Escalón: $f(x)=0$ si $x<0$, 1 si $x\geq 0$

4. Tangente Hiperbólica: $f(x)=\tanh(x)$

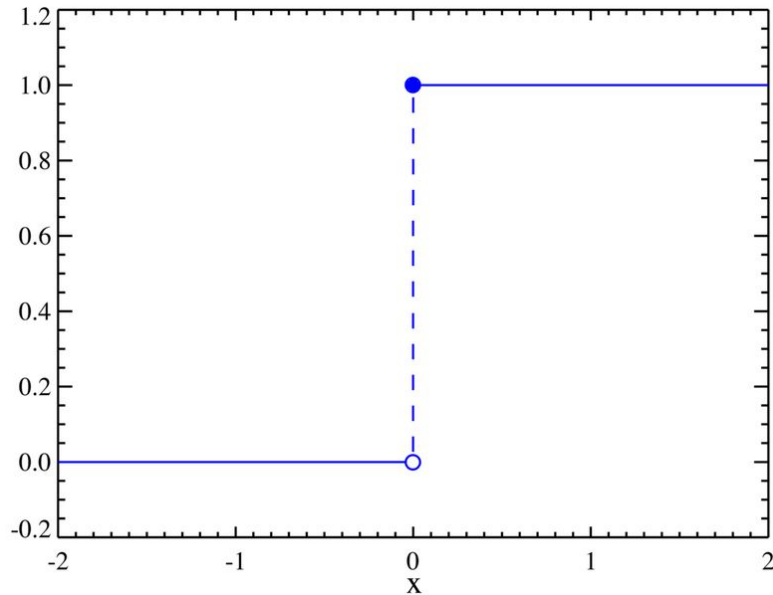
5. ReLU: $f(x)=0$ si $x<0$, x si $x\geq 0$

Perceptrón Multicapa • Funciones de activación



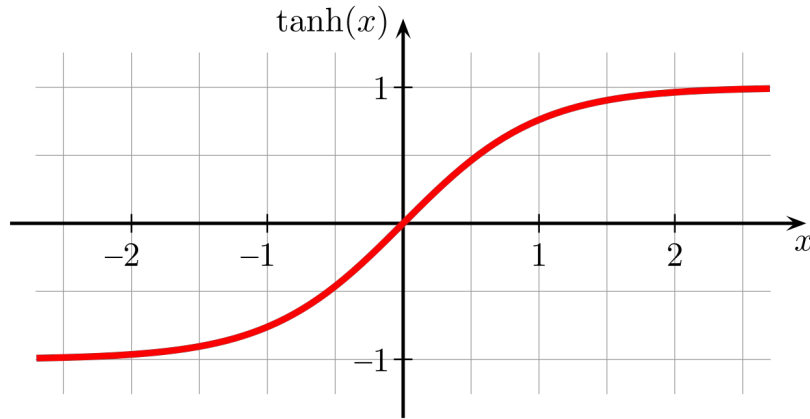
1. Sigmoide/logística
- 2. Identidad: $f(x) = x$**
3. Escalón: $f(x)=0$ si $x<0$, 1 si $x\geq 0$
4. Tangente Hiperbólica: $f(x)=\tanh(x)$
5. ReLU: $f(x)=0$ si $x<0$, x si $x\geq 0$

Perceptrón Multicapa • Funciones de activación



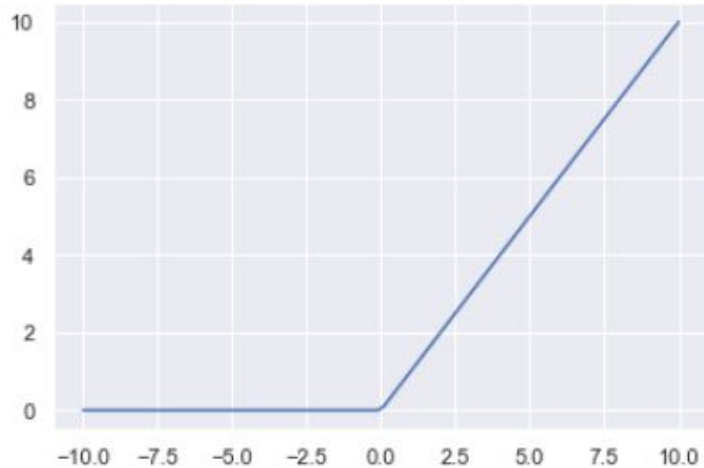
1. Sigmoide/logística
2. Identidad: $f(x) = x$
- 3. Escalón: $f(x)=0$ si $x<0$, 1 si $x\geq 0$**
4. Tangente Hiperbólica: $f(x)=\tanh(x)$
5. ReLU: $f(x)=0$ si $x<0$, x si $x\geq 0$

Perceptrón Multicapa • Funciones de activación



1. Sigmoide/logística
2. Identidad: $f(x) = x$
3. Escalón: $f(x)=0$ si $x<0$, 1 si $x\geq 0$
- 4. Tangente Hiperbólica: $f(x)=\tanh(x)$**
5. ReLU: $f(x)=0$ si $x<0$, x si $x\geq 0$

Perceptrón Multicapa • Funciones de activación



1. Sigmoide/logística
2. Identidad: $f(x) = x$
3. Escalón: $f(x)=0$ si $x<0$, 1 si $x\geq 0$
4. Tangente Hiperbólica: $f(x)=\tanh(x)$
5. **ReLU: $f(x)=0$ si $x<0$, x si $x\geq 0$**

Perceptrón Multicapa • Funciones de activación

Clasificación:
lo más común es
encontrar ReLU en
las capas interiores
y Sigmoides en la
salida

1. **Sigmoide/logística**
2. Identidad: $f(x) = x$
3. Escalón: $f(x)=0$ si $x<0$, 1 si $x\geq 0$
4. Tangente Hiperbólica: $f(x)=\tanh(x)$
5. **ReLU: $f(x)=0$ si $x<0$, x si $x\geq 0$**

Perceptrón Multicapa

	Funciones de activación	Costos (Keras)
Multiclase	<ol style="list-style-type: none">1. Sigmoide/logística2. Softmax	Categorical_crossentropy
Regresión	Identidad	<ol style="list-style-type: none">1. mean_squared_error2. mean_absolute_error3. Otras

Perceptrón Multicapa

	Funciones de activación	Costos (Keras)
Multiclase	<ol style="list-style-type: none">1. Sigmoide/logística2. Softmax	Categorical_crossentropy
Regresión	Identidad	<ol style="list-style-type: none">1. mean_squared_error2. mean_absolute_error3. Otras



Generalización de la sigmoide, útil cuando las clases son excluyentes.

Hands-on training



Hands-on training



DS_Encuentro_31_Perceptron_Multicapa.ipynb

A close-up photograph of a white ceramic cup filled with a latte. The surface of the milk is decorated with intricate latte art, featuring a central heart shape surrounded by concentric, wavy lines. The cup is placed on a matching white saucer. In the background, a white napkin and a silver spoon are visible, though they are out of focus. The overall lighting is soft and even, highlighting the textures of the coffee and the smooth surface of the cup.

¡BREAK!

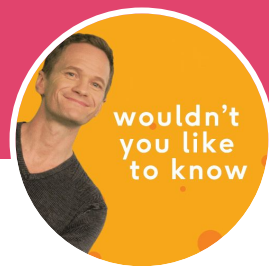


Terminología (extra) de Redes Neuronales



- Regularización
- Batch size
- Epochs
- CNN (Convolutional Neural Networks)

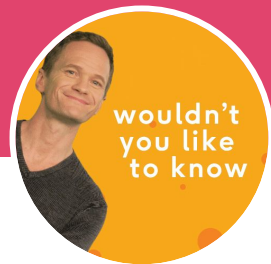
Regularización



Objetivo: castigar parámetros/pesos muy grandes.

están asociados a overfitting.

Regularización



Objetivo: castigar parámetros/pesos muy grandes.

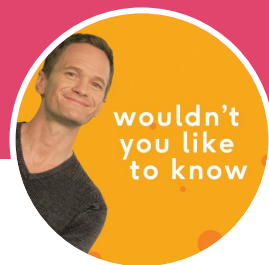
están asociados a overfitting.

¿Cómo? Tres técnicas muy comunes

- Regularización L2 y L1: agregan un término a la función de costo que castiga los pesos grandes.
- **Dropout:** funciona como una capa que “apaga” neuronas de la capa anterior al azar.



Regularización



¿Cómo?

Dropout: funciona como una capa que “apaga” neuronas de la capa anterior al azar.

Muy utilizado. Al apagar neuronas, obliga a que ninguna se aprenda “de memoria” una muestra, sino que tengan que aprender entre todas.
Otra interpretación: Ensamble

Batch size



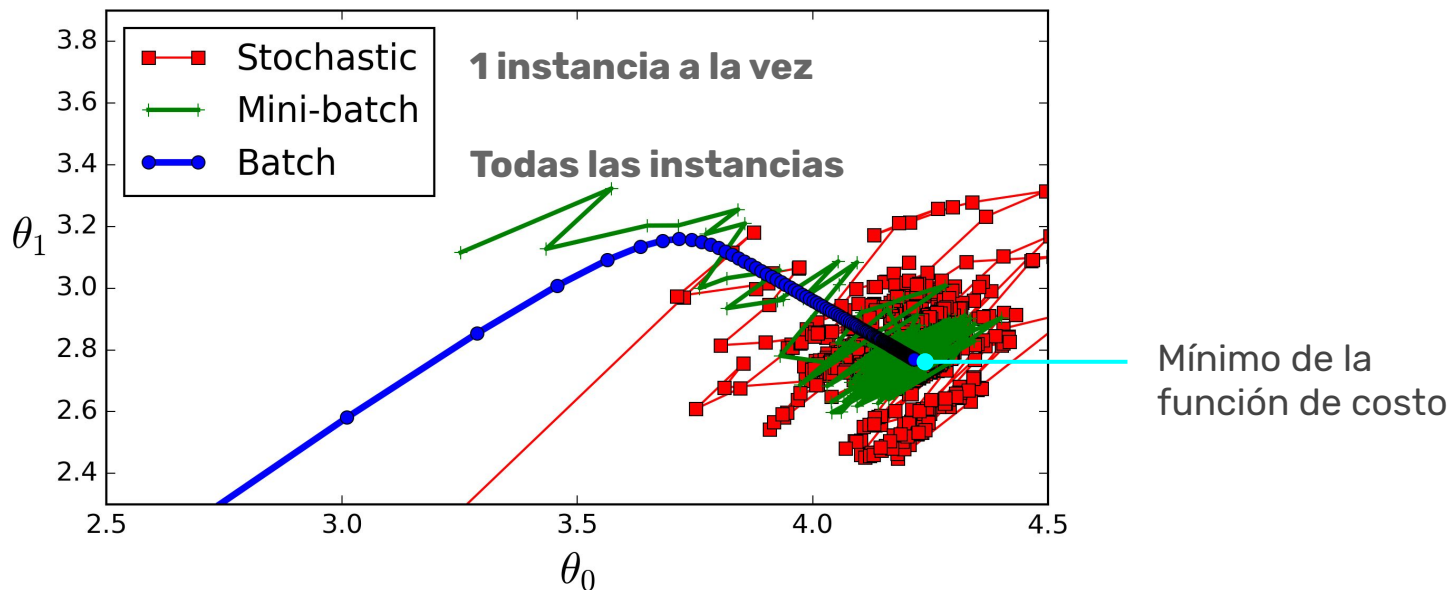
Observación: En muchos casos, el proceso de hacer Forward-propagation y back-propagation puede ser computacionalmente muy costoso.

¿Por qué? Al tener muchas instancias y muchas neuronas, hay que realizar muchas operaciones cada vez que queremos actualizar los parámetros.

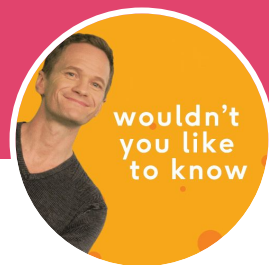
Solución: Separamos nuestro dataset en distintos grupos o 'lotes' (batch).

Batch size

- **Ventaja:** Converge mas rápido (llego antes al mínimo)
- **Desventaja:** El 'camino' (en términos de los parámetros) al máximo no es el 'ideal'



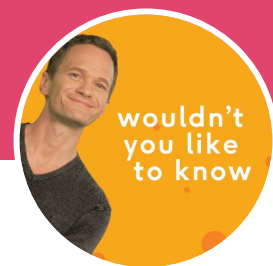
Epochs



Definición: Es el parámetro que controla la cantidad de veces que 'pasamos' (**hacemos Forward y Backpropagation**) sobre todo el dataset (todas las instancias).

Observación: Si el `batch_size` es menor a la cantidad total de instancias `N`, entonces habrá que 'pasar' varias veces para lograr 1 Epoch.

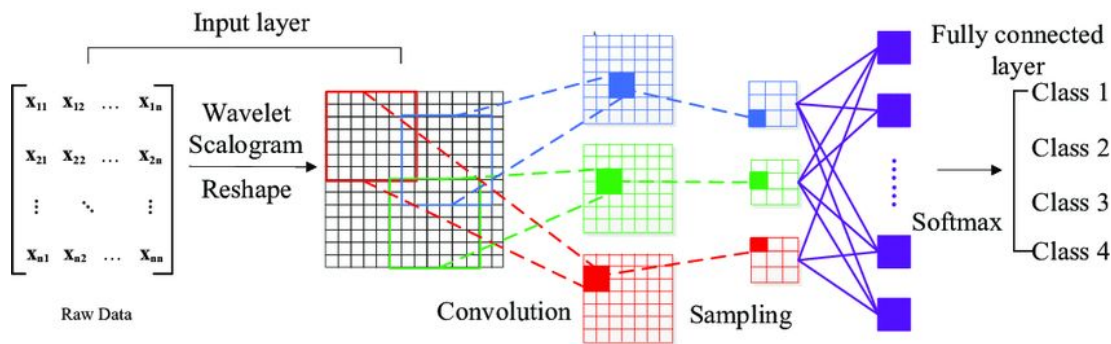
CNNs: Convolutional Neural Networks



Definición: Son un tipo particular de redes neuronales.

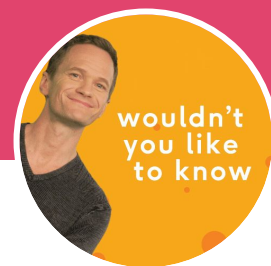
¿Para qué sirven? Son muy buenas para trabajar con imágenes, pero demostraron funcionar bien para diversa clases de problemas

¿Cómo funcionan? La idea principal es que las neuronas del input no ven todos los datos, solo ven partes de los datos (una región). Veremos más de esto el encuentro que viene.



1/3

CNNs: Convolutional Neural Networks



Definición: Son un tipo particular de redes neuronales.

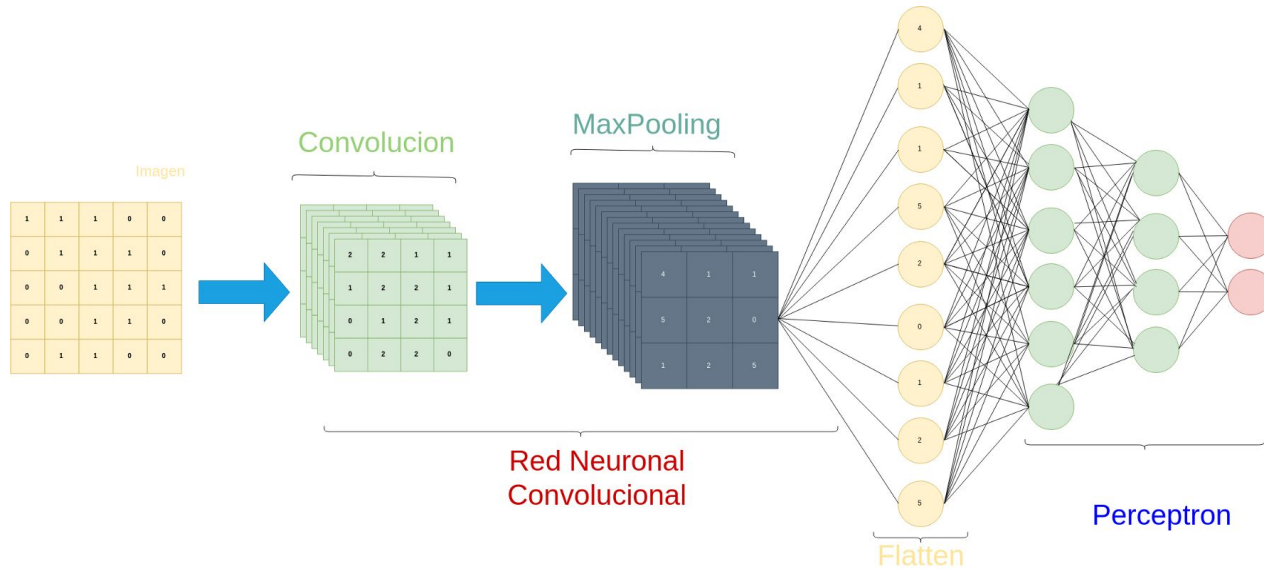
¿Para qué sirven? Son muy buenas para trabajar con imágenes, pero demostraron funcionar bien para diversa clases de problemas

¿Cómo funcionan? La idea principal es que las neuronas del input no ven todos los datos, solo ven partes de los datos (una región). Veremos más de esto el encuentro que viene.



2/3

CNNs: Convolutional Neural Networks



Actividad





Para la próxima: Data Science en mi vida



Data Science en mi vida

¡Preparen sus charlas relámpago!

En 7 minutos con 7 slides comparte con tus compañeros:

En qué problemas estás aplicando lo aprendido en Data Science y cómo lo estás haciendo.

O bien, en qué problemas te gustaría aplicar Data Science y cómo lo harías.

¡Elige algún tema o proyecto que te interese y relaciónalo con lo aprendido!

Para la próxima

1. Ver los videos de la plataforma “Procesamiento del lenguaje natural”
2. Terminar notebook de hoy y atrasados.
3. Preparar el relato “Data Science en mi vida”.

ACÀMICA