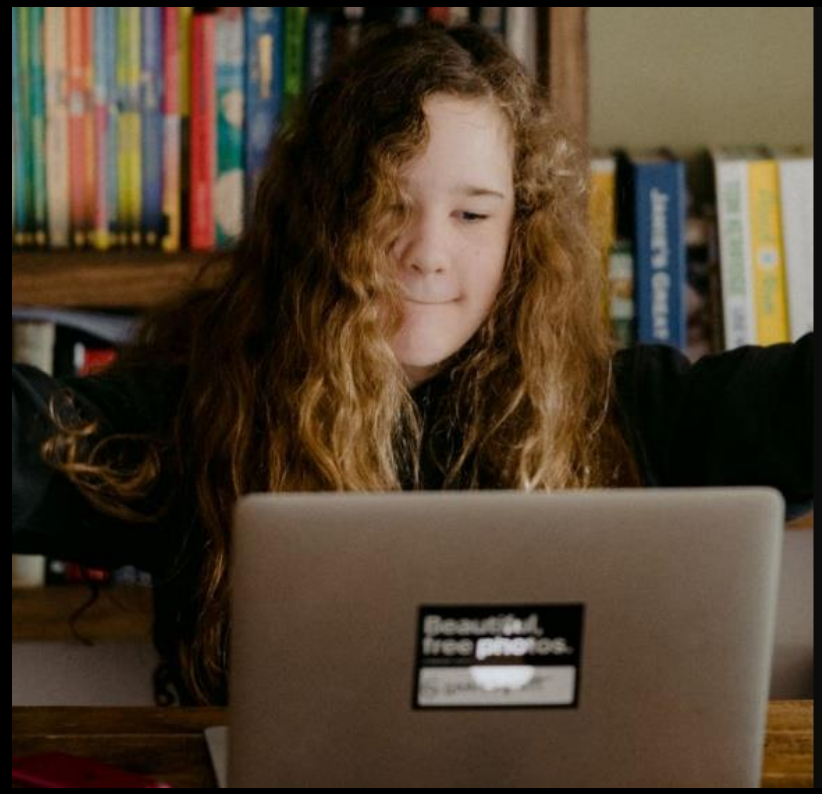
Semilleros
by DSRP

Ruta de Aprendizaje



COMPUTER VISION

- **6 lecciones** de Deep Learning

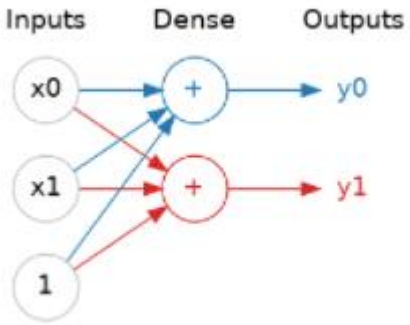
Lesson
#2

Deep Neural Networks

Semilleros
by DSRP

Lesson 2.1: Layers

Las redes neuronales suelen organizar sus neuronas en **layers** (capas). Cuando reunimos unidades lineales que tienen un conjunto común de entradas, obtenemos una capa densa.



A dense layer of two linear units receiving two inputs and a bias.

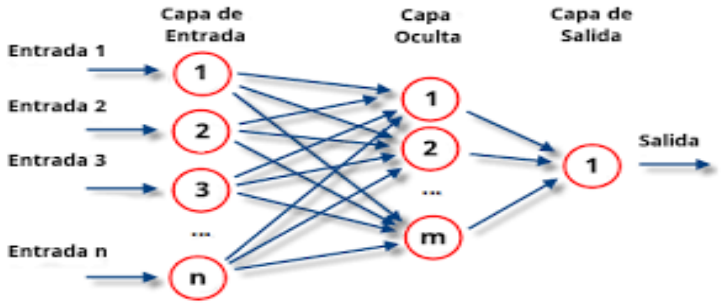
Código anterior

```
from tensorflow import keras
from tensorflow.keras import layers

# Create a network with 1 linear unit
model = keras.Sequential([
    layers.Dense(units=1, input_shape=[3])
])
```

Las capas **Dense** son la capas de cálculo de que conectan cada neurona en una capa con todas las salidas de la capa anterior.

A través de una pila profunda de capas, una red neuronal puede transformar sus entradas de formas cada vez más complejas. En una red neuronal bien entrenada, cada capa es una transformación que nos acerca un poco más a una solución.



Semilleros by DSRP

Lesson 2.2: The activation function

La función de activación se encarga de devolver una salida a partir de un valor de entrada, normalmente el conjunto de valores de salida en un rango determinado como (0,1) o (-1,1).

- Función de activación Tangente hiperbólica
- Función de activación ReLU
- Función de activación Sigmoide
- Función de activación Softmax

Semilleros by DSRP

Lesson 2.2.1: Types of activation functions

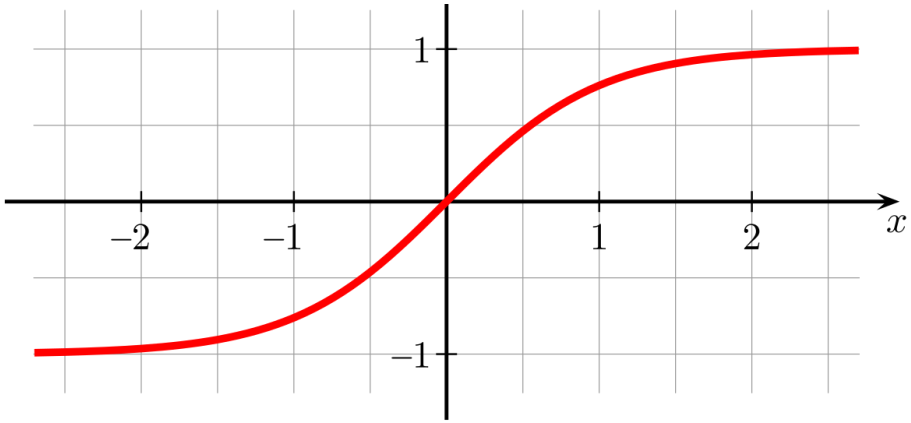
Función de activación “Tangente hiperbólica”

$$f(x) = \frac{2}{1 + e^{-2x}} - 1$$

Función tangente hiperbólica

La función tangente hiperbólica transforma los valores introducidos a una escala (-1, 1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a -1.

Características:



- Muy similar a la sigmoide
- Satura y mata el gradiente
- Lenta convergencia
- Centrada en 0
- Esta acotada entre -1 y 1
- Se utiliza para decidir entre una opción y la contraria
- Buen desempeño en redes recurrentes

Semilleros by DSRP

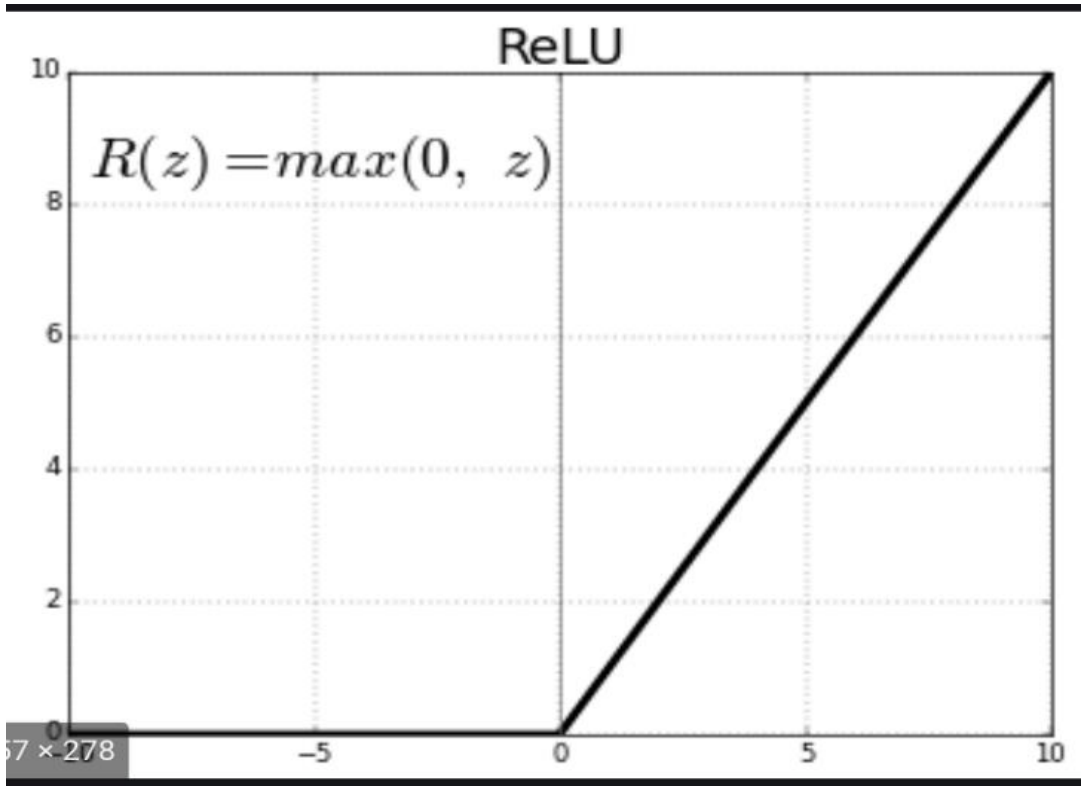
Lesson 2.2.1: Types of activation functions

$$f(x) = \max(0, x) = \begin{cases} 0 \ for \ x < 0 \\ x \ for \ x \geq 0 \end{cases}$$

Función ReLU

Función de activación “ReLU – Rectified Lineal Unit”

La función ReLU transforma los valores introducidos anulando los valores negativos y dejando los positivos tal y como entran.



Características:

- Activación Sparse – solo se activa si son positivos
- No está acotada
- Se pueden morir demasiadas neuronas
- Se comportan bien con imágenes
- Buen desempeño en redes convolucionales

Note: Tiene variantes de activación cómo ‘elu’, ‘selu’, ‘switch’ entre otras las cuáles se pueden usar en Keras

Semilleros by DSRP

Lesson 2.2.1: Types of activation functions

Función de activación “Sigmoide”

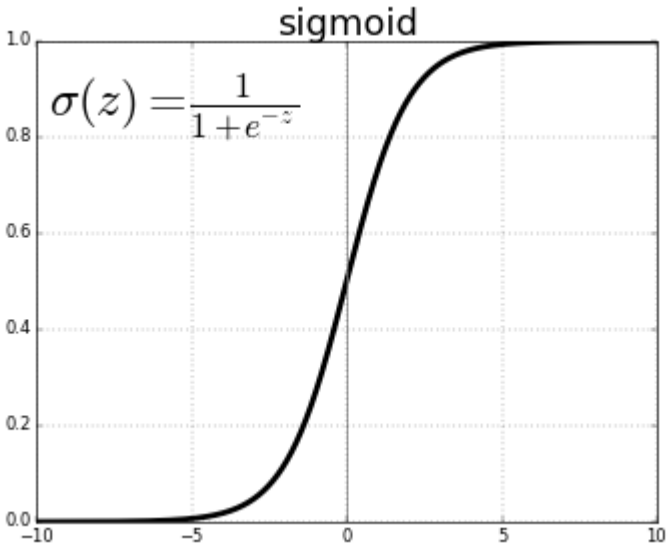
$$f(x) = \frac{1}{1 - e^{-x}}$$

Función Sigmoide

La función de activación sigmoide transforma los valores introducidos a una escala (0,1), donde los valores altos tienen de manera asintótica a 1 y los valores muy bajos tienden de manera asintótica a 0.

Características:

- Satura y mata el gradiente
- Lenta convergencia
- No esta centrada en el cero
- Esta acotada entre 0 y 1
- Buen rendimiento en la última capa



Semilleros by DSRP

Lesson 2.2.1: Types of activation functions

Función de activación “Softmax”

$$f(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

Función Softmax

La función Softmax transforma las salidas a una representación en forma de probabilidades, de tal manera que el sumatorio de todas las probabilidades de las salidas de 1.

Características:

- Se utiliza cuando queremos tener una representación en forma de probabilidades
- Esta acotada entre 0 y 1
- Muy diferenciable
- Se utiliza para normalizar tipo multiclase
- Buen rendimiento en las últimas capas

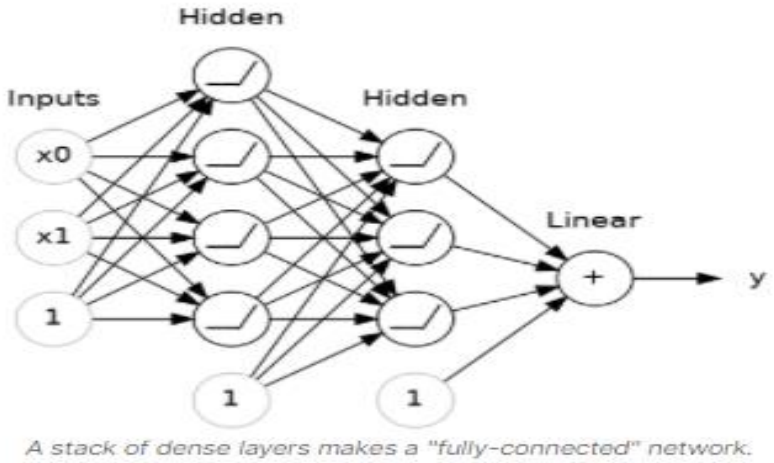
Note: Utilice las funciones de activación sigmoide y softmax para realizar una clasificación de clases múltiples y una clasificación de etiquetas múltiples. Generalmente, softmax se utiliza para la clasificación de clases múltiples. sigmoide se utiliza para la clasificación de dos o múltiples etiquetas.

Semilleros by DSRP

Lesson 2.3: Stacking Dense Layers

Las capas antes de la capa de salida a veces se llaman **hidden** (ocultas), ya que nunca vemos sus salidas directamente.

Ahora, observe que la capa final (salida) es una unidad lineal (es decir, sin función de activación). Eso hace que esta red sea apropiada para una tarea de regresión, donde estamos tratando de predecir algún valor numérico arbitrario.



A stack of dense layers makes a "fully-connected" network.

```
#crear una red neuronal perceptrón multicapa

model = keras.Sequential([

#crear capa de entrada
    layers.Dense(units=1, activation = 'relu', input_shape=[15]),

#crear capa oculta
     layers.Dense(units=1, activation = 'relu'),

#crear capa de salida

     layers.Dense(units=1, activation='sigmoid'),

   ])
```

Otras tareas (como la clasificación) pueden requerir una función de activación en la salida.

Semilleros by DSRP

Lesson 2.4: Building Sequential Models

El modelo secuencial que hemos estado usando conectará una lista de capas en orden de la primera a la última: la primera capa obtiene la entrada, la última capa produce la salida. Esto crea el modelo en la figura anterior:

In [1]:

```
from tensorflow import keras
from tensorflow.keras import layers

model = keras.Sequential([
    # the hidden ReLU layers
    layers.Dense(units=4, activation='relu', input_shape=[2]),
    layers.Dense(units=3, activation='relu'),
    # the linear output layer
    layers.Dense(units=1),
])
```

Note: Ahora, observe que la capa final (salida) es una unidad lineal (es decir, sin función de activación). Eso hace que esta red sea apropiada para una tarea de regresión, donde estamos tratando de predecir algún valor numérico arbitrario.

Note: Otras tareas (como la clasificación) pueden requerir una función de activación en la salida.

Semilleros by DSRP

Diferencias entre algoritmos de Clasificación y Regresión

- Predecir si una persona está infectada con dengue

- Predecir el precio de una casa
- Eficiencia de combustible de un automóvil

Semilleros by DSRP

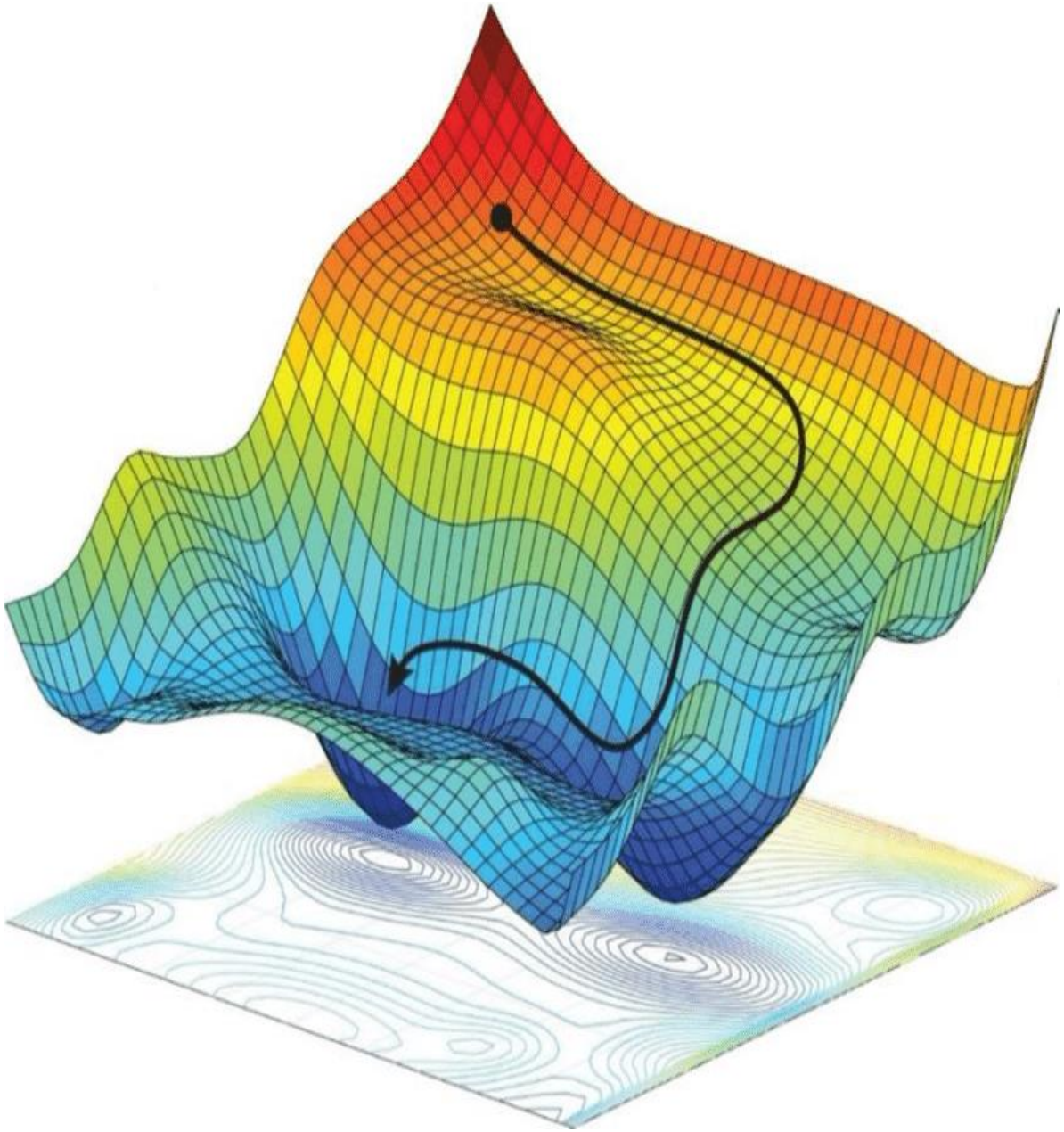
Ejercicios

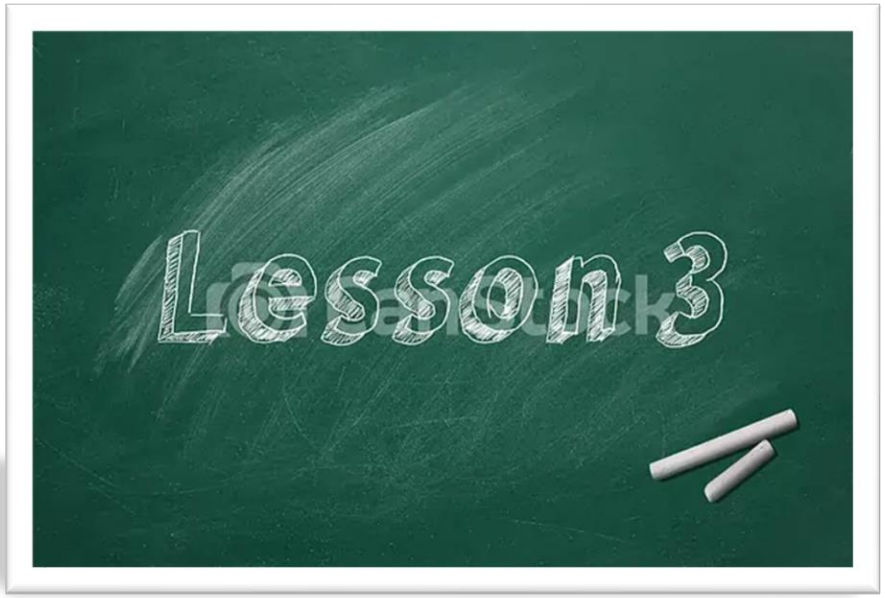
Si una nueva película que está saliendo en los cines te va a gustar o no

---

En caso de que si te guste la nueva película, ¿cuántas veces la verás?


LABORATORIO 2
PRECISE V7
Semilleros
by DSRP





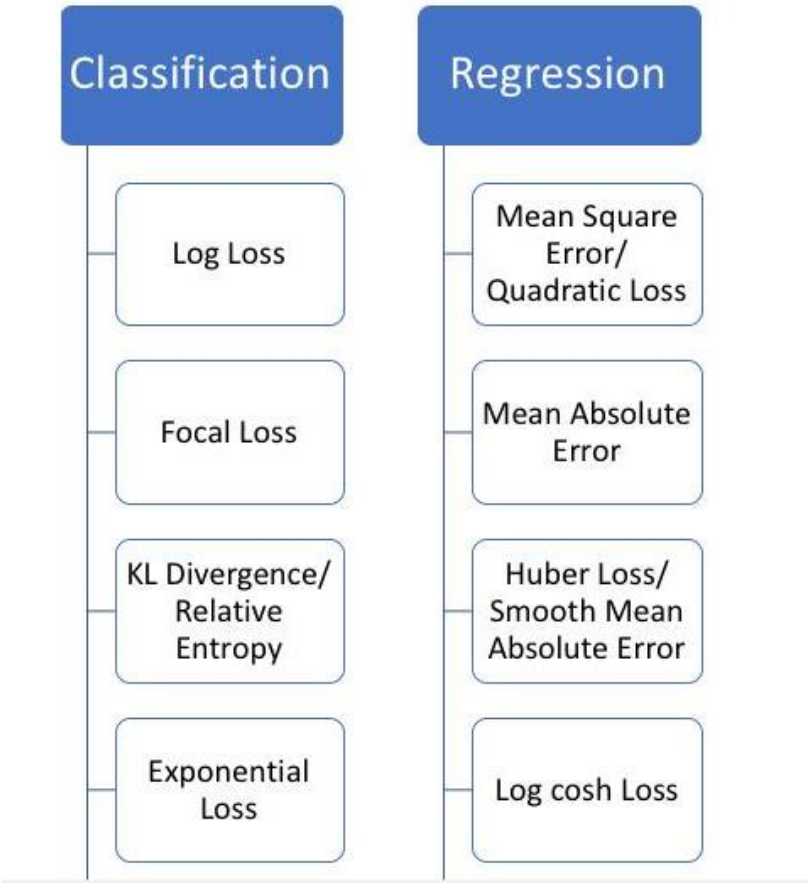
Stochastic Gradient Descent

Semilleros by DSRP

Lesson 3.1: The lost function

La función de pérdida evalúa la desviación entre las predicciones realizadas por la red neuronal y los valores reales de las observaciones utilizadas durante el aprendizaje. Cuanto menor es el resultado de esta función, más eficiente es la red neuronal.

La función de pérdida se puede dividir aproximadamente en dos categorías: pérdida de clasificación y pérdida de regresión.



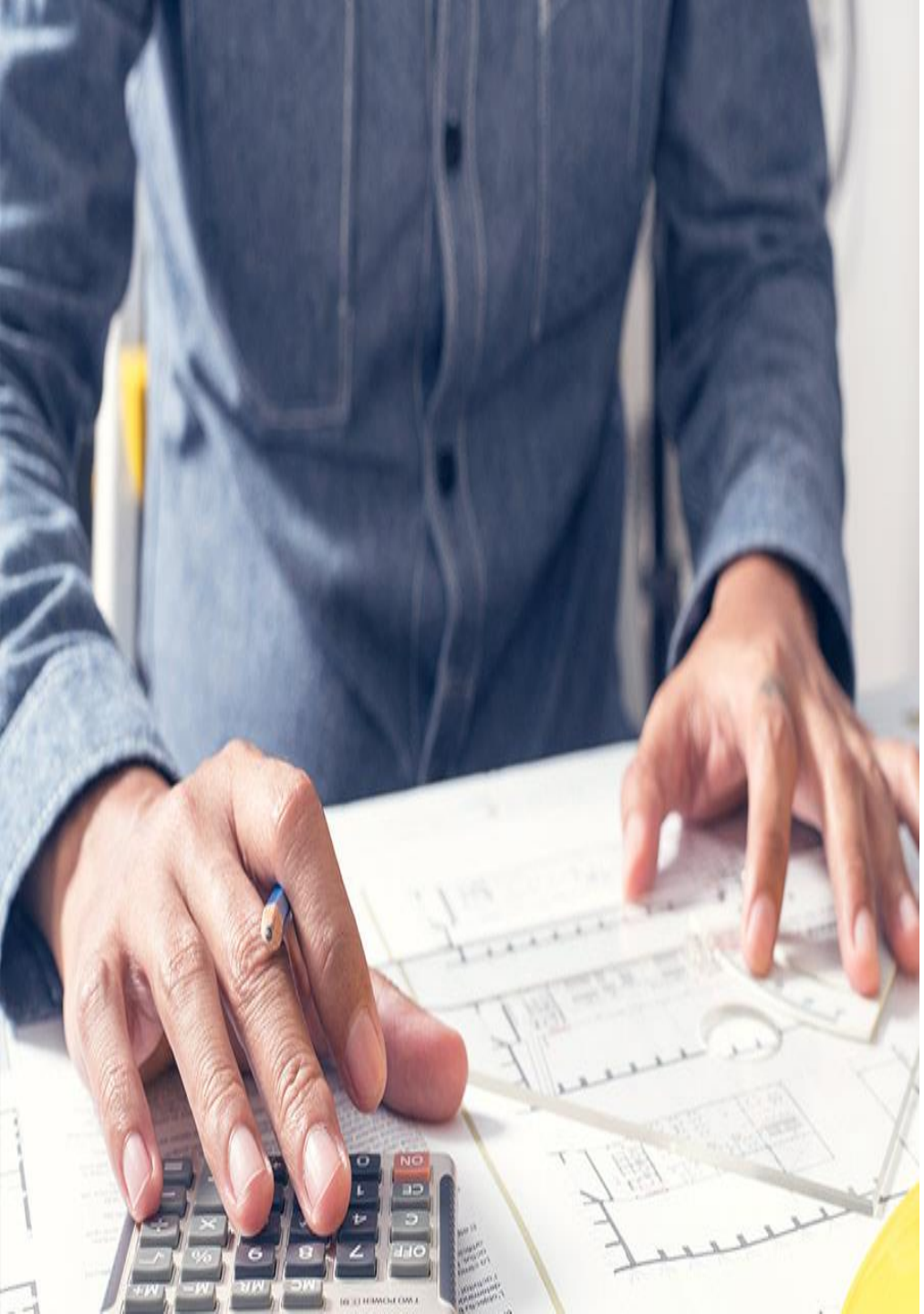
Classification

La función de clasificación predice la etiqueta.

Regression

La regresión, por otro lado, trata de predecir un valor continuo, por ejemplo, el área del piso , el número de habitaciones, el tamaño de las habitaciones, predecir el precio de la habitación.

Semilleros by DSRP



Lesson 3.1: The lost function

Regression

MSE (MEAN SQUARE ERROR)

El error cuadrático medio se mide como el promedio de la diferencia al cuadrado entre las predicciones y las observaciones reales. Solo le preocupa la magnitud promedio del error, independientemente de su dirección.

- El MSE tiene buena propiedades matemáticas que hacen que sea más fácil calcular gradientes.

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Semilleros by DSRP

Lesson 3.1: The lost function

Regression

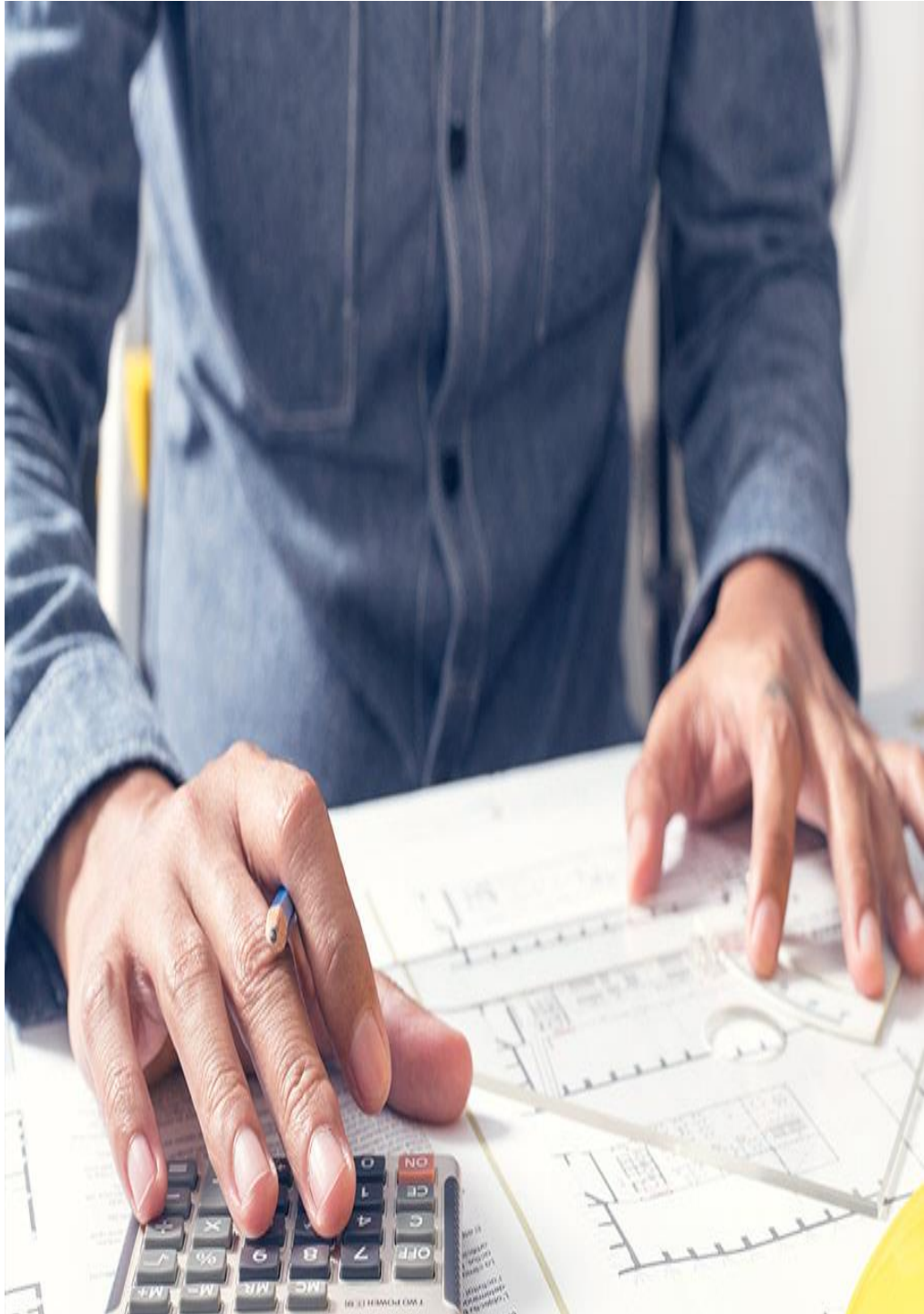
MAE (MEAN ABSOLUTE ERROR)

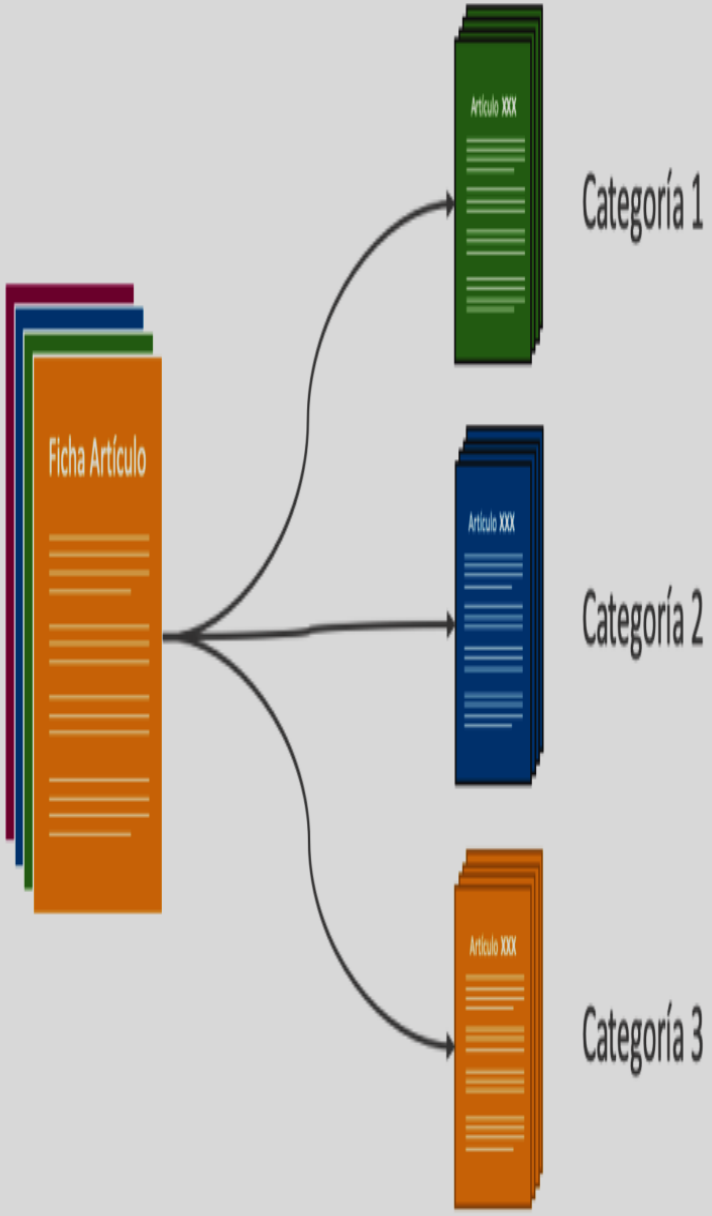
El error absoluto medio, se mide como el promedio de la suma de las diferencias absolutas entre las predicciones y las observaciones reales. También mide la magnitud del error sin considerar su dirección al igual que el MSE.

- Es más robusto para los **outliers** (valores atípicos), ya que no utiliza cuadrado.
- Necesita herramientas más complicadas, como la programación lineal para calcular los gradientes.

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Semilleros by DSRP





Lesson 3.1: The lost function

Classification

CrossEntropyLoss

Esta es la más común para los problemas de clasificación. La pérdida de entropía cruzada o la pérdida de registro mide el rendimiento de un modelo de clasificación cuya salida es un valor de probabilidad entre 0 y 1.

- La pérdida de entropía cruzada aumenta a medida que la probabilidad prevista diverge de la etiqueta real.

$$L_{\text{cross-entropy}}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i y_i \log(\hat{y}_i)$$

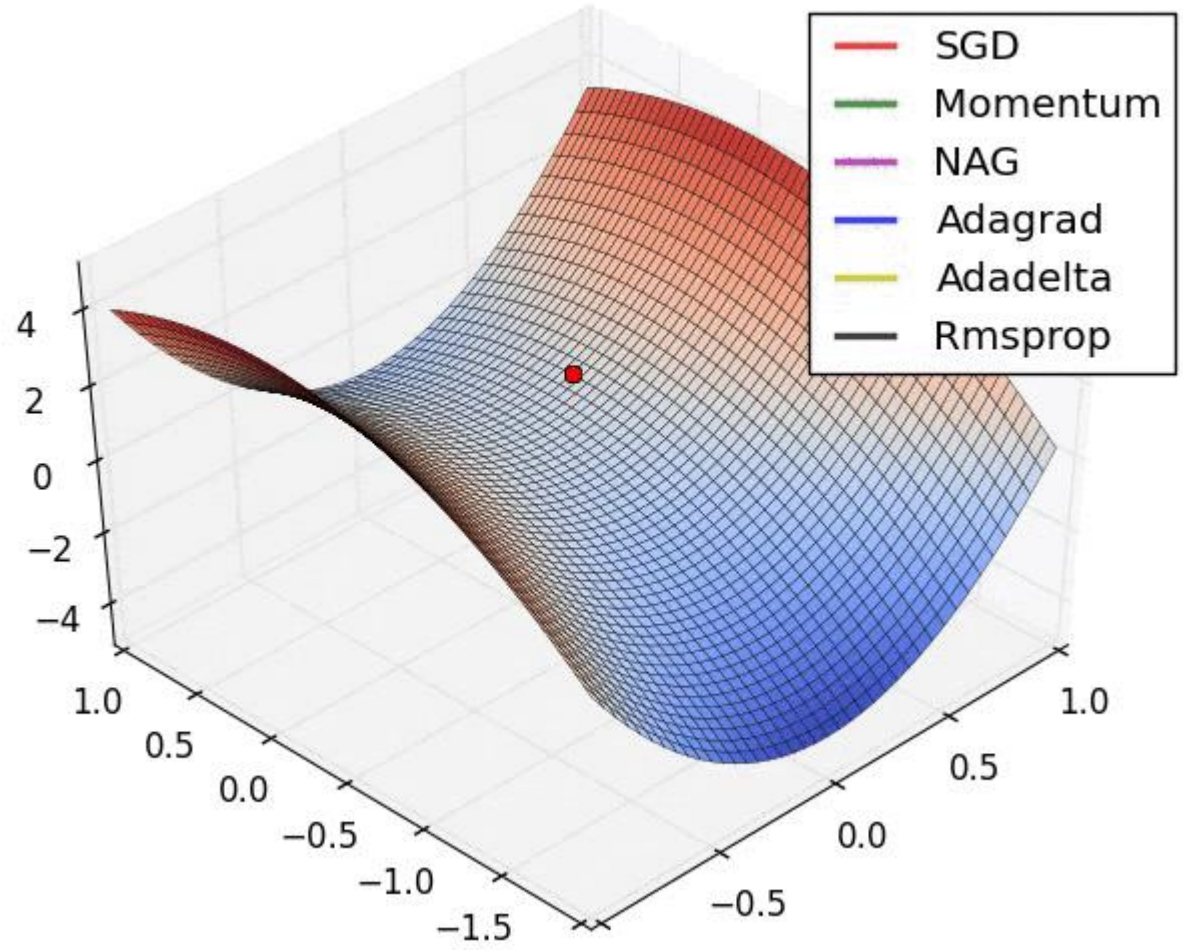
Semilleros by DSRP

Lesson 3.2: The optimizer

Hemos descrito el problema que queremos que resuelva la red, pero ahora tenemos que decir cómo resolverlo. Este es el trabajo del optimizador. El optimizador es un algoritmo que ajusta los pesos para minimizar la pérdida.

El optimizador Descenso de Gradiente visto no el único utilizado. De hecho, existe toda una familia de optimizadores que, basados en Descenso de Gradiente, intentan mejorar el rendimiento de éste. Entre otros, tenemos:

- Stochastic Gradient Descent
- Mini-batch Gradient Descent
- Momentum
- AdaGrad
- RMSProp
- Adam



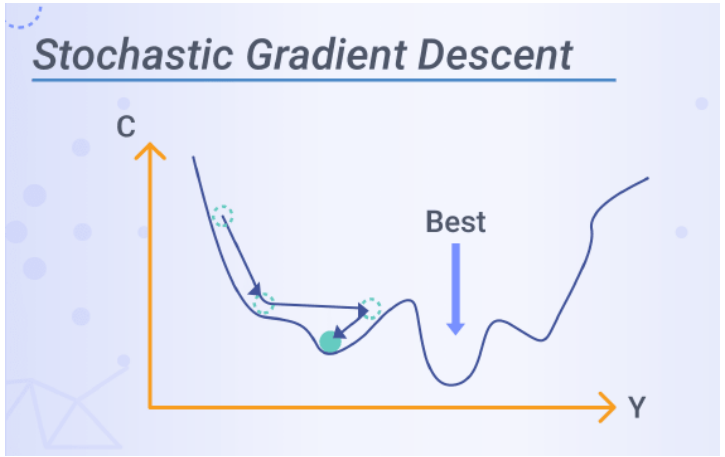
Semilleros by DSRP

Lesson 3.2: The optimizer

Stochastic Gradient Descent

El optimizador Descenso de Gradiente Estocástico simplifica el cálculo considerando solo una muestra escogida de forma aleatoria cada vez que realiza el cálculo del gradiente.

Es decir, pasamos una única muestra, calculamos la función de error asociada y, a partir de esta, el gradiente y los incrementos a aplicar.

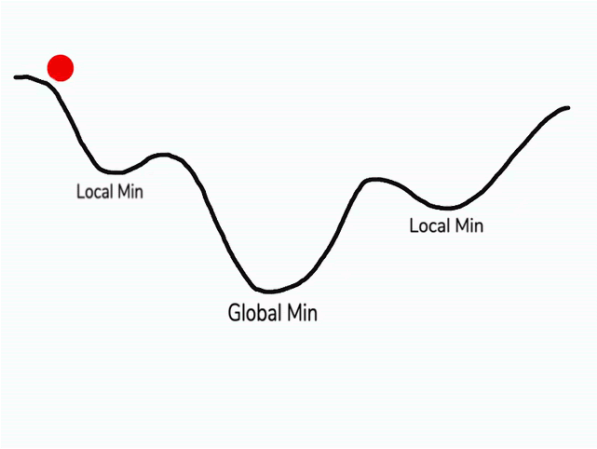


Adam

El algoritmo de Adam obtiene las ventajas de los algoritmos AdaGrad y RMSProp.

- Utiliza las estimaciones de primer y segundo momento del gradiente para ajustar dinámicamente la tasa de aprendizaje de cada parámetro.

- La principal ventaja de Adam es que después de la corrección de desplazamiento, la tasa de aprendizaje de cada iteración tiene un cierto rango, lo que hace que los parámetros sean relativamente estables.



Semilleros by DSRP

Lesson 3.2: Adding the Loss and Optimizer

Después de definir un modelo, puede agregar una función de pérdida y un optimizador con el método de compilación del modelo:

```
model.compile(
    optimizer="adam",
    loss="mae",
)
```

Conceptos básicos

Minibatch: La muestra de datos de entrenamiento de cada iteración

Epochs: Es una ronda completa de datos de entrenamiento

Learning Rate: La tasa de aprendizaje es un parámetro de ajuste que determina el tamaño del paso en cada iteración.

Semilleros by DSRP