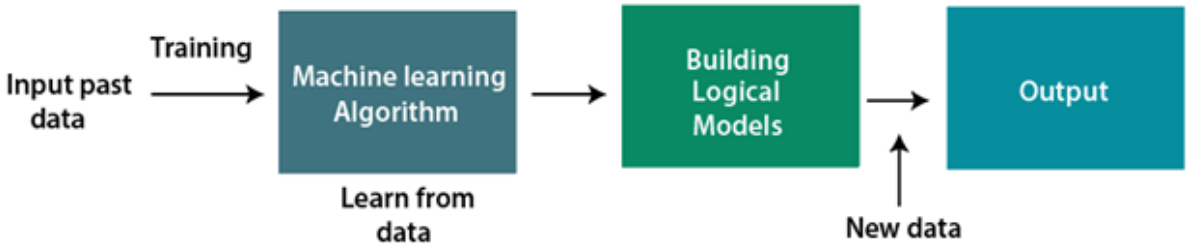
LUCAS SAETA

TEMA 3: Machine Learning

Machine Learning:

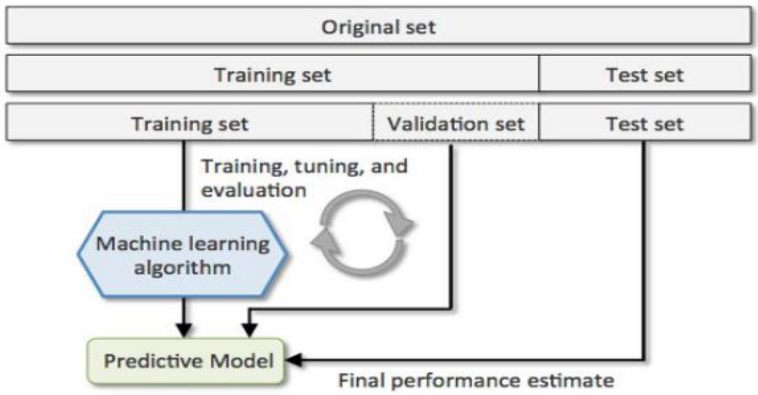
- El aprendizaje automático es una tecnología en auge que permite a los ordenadores aprender automáticamente a partir de datos anteriores. El aprendizaje automático utiliza diversos algoritmos para construir modelos matemáticos y hacer predicciones a partir de datos o información históricos.
- Un sistema de aprendizaje automático aprende de los datos históricos, construye los modelos de predicción y, cuando recibe nuevos datos, predice el resultado. La precisión de la predicción depende de la cantidad de datos, ya que una gran cantidad de datos ayuda a construir un modelo mejor que predice la salida con mayor precisión.
- La inteligencia artificial es una tecnología que permite crear sistemas inteligentes capaces de simular la inteligencia humana, mientras que el aprendizaje automático es un subcampo de la inteligencia artificial que permite a las máquinas aprender de datos o experiencias anteriores.



Target:

- Nominal: tendrás modelos de clasificación. Y luego puedes tener objetivos binarios (dicotómicos) o con más valores posibles (politómicos)

- A nivel de intervalo: Modelos de tipo regresión



Modelos supervisados:

El aprendizaje supervisado es un tipo de método de aprendizaje automático en el que proporcionamos datos etiquetados (sample labeled data) de muestra al sistema de aprendizaje automático para entrenarlo y, basándose en ellos, predice el resultado.

El sistema crea un modelo utilizando datos etiquetados (labeled data) para comprender los conjuntos de datos y aprender sobre cada uno de ellos. Una vez finalizado el entrenamiento y el procesamiento, probamos el modelo proporcionando una muestra de datos para comprobar si predice el resultado exacto o no.

- Los datos contienen un target, es decir, un label o labels que se utilizan para entrenar los modelos.
- Los modelos de aprendizaje supervisados pueden ser algoritmos de clasificación o de regresión
- En función del target (label), se intenta predecir(predict) numéricamente (nivel de intervalo) o clasificar (categoría) nivel nominal.
- Las medidas de rendimiento se eligen en función del nivel de medición del objetivo y de las propiedades matemáticas del modelo o algoritmo.

Modelos no supervisados:

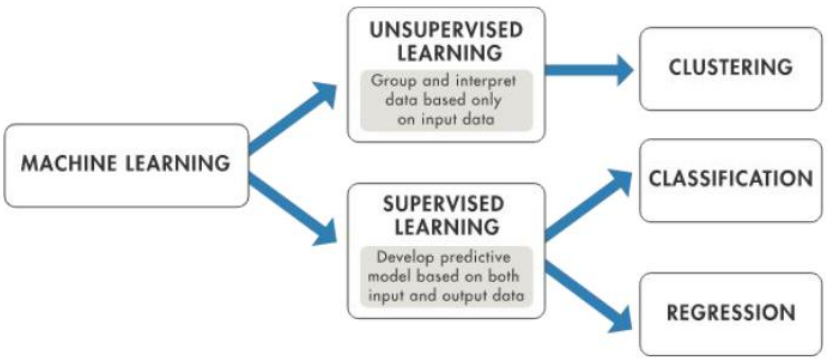
El aprendizaje no supervisado es un método de aprendizaje en el que una máquina aprende sin ningún tipo de supervisión.

La formación se proporciona a la máquina con el conjunto de datos que no han sido etiquetados, clasificados o categorizados (not been labeled , classified or categorized), y el algoritmo tiene que actuar sobre esos datos sin ningún tipo de supervisión. El objetivo del aprendizaje sin supervisión es reestructurar los datos de entrada en nuevas características o en un grupo de objetos con patrones similares.

- Los datos no contienen un target. El target (ahora su "blanco"/objetivo) es encontrar patrones (grupos, conglomerados, segmentos, temas) en datos no etiquetados (un-label).

- Las medidas de rendimiento (performance) no están tan bien definidas como en los modelos supervisados -> es más importante obtener resultados significativos que el rendimiento de una función matemática.

- Los modelos de aprendizaje no supervisados pueden ser algoritmos de clustering o de asociacion



Modelos de Aprendizaje por refuerzo (reinforced learning):

El aprendizaje por refuerzo es un método de aprendizaje basado en la retroalimentación, en el que un agente que aprende obtiene una recompensa por cada acción correcta y recibe una penalización por cada acción incorrecta.

El agente aprende automáticamente con estas retroalimentaciones y mejora su rendimiento. En el aprendizaje por refuerzo, el agente interactúa con el entorno y lo explora. El objetivo de un agente es obtener el mayor número de puntos de recompensa y, por tanto, mejorar su rendimiento**.**

Modelo (estadístico, también algoritmo): la formulación matemática de un

mecanismo para realizar el entrenamiento y la predicción. Puede utilizar tantos como

que su sistema pueda calcular, compararlos y utilizar el que mejor funcione -o el que

el que mejor se entienda o aplique.

Conjunto(s) de datos: conjuntos de entrenamiento, validación y prueba. (training, validation and test sets)

Evaluación del rendimiento: en función y depende del target (clasificación / regresión).

Overfitting: un problema desagradable de algunos modelos que tienen excelente

rendimiento con los datos de entrenamiento y validación, pero fallan con datos de prueba diferentes. Este problema también se conoce como "estos modelos no

generalizan". Solo aprenden lo entrenado.

Regresión lineal: algoritmo de machine learning basado en el aprendizaje supervisado. Realiza una tarea de regresión. La regresión modela un valor de predicción objetivo basado en variables independientes. Se utiliza sobre todo para averiguar la relación entre variables y hacer previsiones.

Regresión lineal Fórmula matemática: y= a0+a1x+ ε donde: Y= Variable dependiente (Variable target/objetivo), X= Variable independiente (Variable predictora), a0= intercepto (intercept) de la recta (Da un grado de libertad adicional), a1 = coeficiente de regresión lineal (factor de escala a cada valor de entrada), ε = error aleatorio

Math notation	Identical Concepts (in **bold** ML usual concepts)	Python examples
y	Predicted variable, **target**	y_training, y_test
X	Predictor variables, **features**	X_training, X_test
ŷ, yhat	Fitted values, estimates, predictions	model.predict(X_test)
β, w	Parameters, coefficients, **weights**	model.coef_()
β0, b	Constant parameter, **intercept, bias**	model.intercept_()
ε	Random error, **noise**	e = y - yhat

X y ^y yhat B w B0 b 3 e (para buscar en el ctrl f, la tabla esta arriba)

y = X1w1+b (creo q es la misma q la de regresión lineal xd)

TEMA 4: Artificial Neural Network

Una red neuronal es un método de inteligencia artificial que enseña a los ordenadores a procesar datos inspirándose en el cerebro humano. Es un tipo de proceso de aprendizaje automático (machine learning), denominado aprendizaje profundo (Deep learning), que utiliza nodos o neuronas interconectados en una estructura en capas que se asemeja al cerebro humano.

Una red neuronal artificial es un intento de simular la red de neuronas que componen un cerebro humano para que el ordenador sea capaz de aprender cosas y tomar decisiones de forma parecida a los humanos. Las RNA/ANN se crean programando ordenadores normales para que se comporten como si fueran células cerebrales interconectadas.

Epoch: Las épocas se definen como el número total de iteraciones para entrenar el modelo de aprendizaje automático con todos los datos de entrenamiento en un ciclo. Una época en el aprendizaje automático significa una pasada completa del conjunto de datos (iteración) de entrenamiento por el algoritmo. El número de épocas es un hiperparámetro importante para el algoritmo.

I**teración:** se define como el número total de batches necesarios para completar una época, donde el número de batches es igual al número total de iteraciones de una época. Iteracion=training examples/batch size

Batch size: El tamaño del batch se define como el número total de ejemplos de entrenamiento que existen en un único lote.

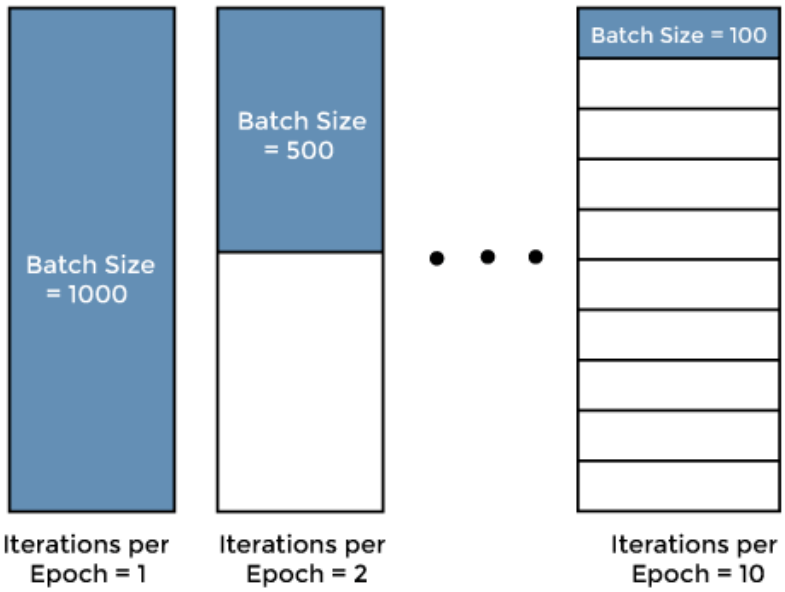
Cuando todas las muestras de entrenamiento se utilizan para crear un batch, el algoritmo de aprendizaje se denomina batch gradient descent. Cuando el batch tiene el tamaño de una muestra, el algoritmo de aprendizaje se denomina stochastic gradient descent. Cuando el tamaño del lote es superior a una muestra e inferior al tamaño del conjunto de datos de entrenamiento, el algoritmo de aprendizaje se denomina mini-batch gradient descent.

Batch Gradient Descent. Batch Size = Size of Training Set

Stochastic Gradient Descent. Batch Size = 1

Mini-Batch Gradient Descent. 1 < Batch Size < Size of Training Set

En el caso del mini-batch gradient descent, los tamaños de batch más comunes son 32, 64 y 128 muestras.



Ejemplo epochs, iterations, batches:

Número total de ejemplos de entrenamiento = 3000;

Supongamos que cada tamaño de batch= 500;

Entonces el número total de iteraciones = Número total de ejemplos de entrenamiento/Tamaño de batch individual = 3000/500

-> Número total de iteraciones = 6

Y entonces -> 1 Época = 6 Iteraciones

Loss: La función los en una red neuronal cuantifica la diferencia entre el resultado esperado y el resultado producido por el modelo de aprendizaje automático. A partir de la función loss, podemos obtener los gradientes que se utilizan para actualizar los pesos. La media de todas las loss constituye el coste.

Para optimizar un algoritmo de aprendizaje automático se utiliza una función loss. La loss se calcula en el entrenamiento y la validación y su interpretación se basa en lo bien que lo hace el modelo en estos dos conjuntos. El valor de la pérdida implica lo mal o bien que se comporta un modelo después de cada iteración de optimización.

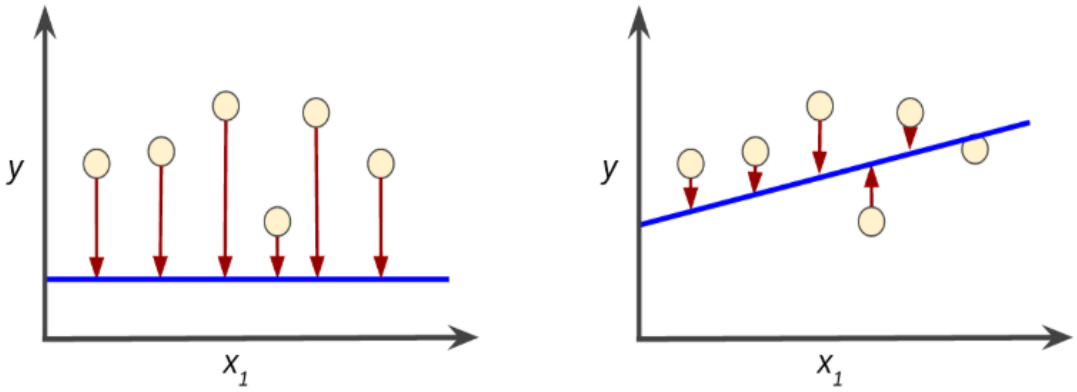
Si la predicción del modelo es perfecta, la loss es cero; en caso contrario, la pérdida es mayor. El objetivo del entrenamiento de un modelo es encontrar un conjunto de weights y biases que tengan una loss baja, de media, en todos los ejemplos. Una loss mayor es peor (mala predicción) para cualquier modelo.

La loss se calcula sobre el entrenamiento y la validación y su interpretación es lo bien que lo hace el modelo para estos dos conjuntos. A diferencia del accuracy, la loss no es un porcentaje. Es una suma de los errores cometidos para cada ejemplo en los conjuntos de entrenamiento o validación.

- Una vez definida la arquitectura de la red, aún hay que elegir dos cosas más:

  La loss (y por tanto el coste) y el optimizador son claves para configurar el proceso de aprendizaje

el loss es la línea roja, la línea azul es la predicción



Binary Cross-Entropy

Estas funciones de pérdida se utilizan para medir el rendimiento del modelo de clasificación. En ellas, a los puntos de datos se les asigna una de las etiquetas, es decir, o bien 0 o bien 1.

Es una función de pérdida predeterminada para problemas de clasificación binaria. La pérdida de entropía cruzada calcula el rendimiento de un modelo de clasificación, que da como salida un valor de probabilidad entre 0 y 1. La pérdida de entropía cruzada aumenta a medida que el valor de probabilidad predicho se desvía de la etiqueta real.

Categorical cross-entropy loss

Calcula la pérdida de crossentropy cruzada entre las etiquetas y las predicciones.

Utilice esta función de loss de crossentropy cruzada cuando haya dos o más clases de etiquetas. Ej:

Suppose I ask you 'Who is this actor?'

And I give you 3 options.

1. Chris Evans
2. RDJ
3. Chris Hemsworth

We all know he is RDJ. So, the correct answer 'y_true' is

$$y\_true = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

But, you don't know who he is and you guess the answer with probabilities 'y_pred'

$$y\_pred = \begin{bmatrix} 0.2 \\ 0.45 \\ 0.35 \end{bmatrix}$$

Optimizer: algoritmos o métodos utilizados para cambiar los atributos de la red neuronal, como los weights y learning rates, con el fin de reducir las losses. Los optimizadores se utilizan para resolver problemas de optimización minimizando la función. Un optimizador es un algoritmo utilizado en el aprendizaje automático para actualizar los parámetros de un modelo en base a los datos de entrenamiento y la función de loss.

Los optimizadores son algoritmos o métodos utilizados para minimizar una función de error (función de pérdida) o para maximizar la eficiencia de la producción. Los optimizadores son funciones matemáticas que dependen de los parámetros de aprendizaje del modelo, es decir, los pesos y los biases. Los optimizadores ayudan a saber cómo cambiar los pesos y el ritmo de aprendizaje de la red neuronal para reducir las pérdidas.

Un optimizador utiliza la información proporcionada por la función de pérdida para actualizar los parámetros del modelo de manera que minimice el error y mejore el rendimiento del modelo. Existen muchos optimizadores diferentes disponibles, cada uno con sus propias características y ventajas.

Un ejemplo común de un optimizador es **Adam** (Adaptive Moment Estimation), que es un método de optimización basado en el gradiente que se utiliza a menudo en el entrenamiento de redes neuronales. Adam combina dos técnicas de optimización: el gradiente descendente estocástico y el momento. Esto permite que Adam se ajuste automáticamente a los datos de entrenamiento

Otros ejemplos puede ser el Gradiente descendente estocástico **(SGD):** Este es un optimizador basado en el gradiente que se utiliza para actualizar los parámetros del modelo una vez por cada muestra de entrenamiento. **SGD** es fácil de implementar y tiene una complejidad computacional baja, pero puede tardar más tiempo en converger que otros optimizadores.

RMSProp: Este optimizador es similar a SGD, pero utiliza una tasa de aprendizaje adaptativa que se ajusta automáticamente en función de la historia del gradiente. Esto puede ayudar a acelerar la convergencia y mejorar el rendimiento del modelo.

Pesos (weights en inglés) y los Sesgos (biases en inglés): parámetros que se utilizan en los modelos para controlar la salida de una neurona o el comportamiento del modelo en general. Son parámetros que se utilizan en los modelos para controlar cómo se procesa la información y cómo se toma una decisión.

Los pesos son valores numéricos asociados a las conexiones entre las neuronas de una red neuronal. Cada conexión tiene un peso asociado que controla la importancia de la señal de entrada para la salida de la neurona. Durante el entrenamiento de una red neuronal, los pesos se ajustan para minimizar el error entre la salida deseada y la salida del modelo. Los pesos se utilizan para determinar la importancia de cada una de las entradas de un modelo en la toma de decisiones. Por ejemplo, en una red neuronal, cada conexión entre dos neuronas tiene un peso asociado. Durante el entrenamiento, estos pesos se ajustan para reflejar la importancia de cada conexión en la producción de una salida correcta.

Los sesgos son valores numéricos que se utilizan para desplazar la curva de activación de una neurona hacia arriba o hacia abajo. Esto permite que la salida de la neurona sea más o menos sensible a la entrada. Al igual que los pesos, los sesgos también se ajustan durante el entrenamiento para minimizar el error. Por ejemplo, en una red neuronal, cada neurona tiene un sesgo asociado

que se suma a la señal de entrada antes de aplicar la función de activación. Esto permite que el modelo tome decisiones basándose en un umbral específico.

Funciones de activación: elemento esencial en los modelos de aprendizaje automático, en particular en las redes neuronales artificiales. Se utilizan para calcular la salida de una neurona a partir de la señal de entrada que recibe y su peso asociado.

La función de activación **softmax:** función de activación utilizada a menudo en las redes neuronales para problemas de clasificación múltiple. Se utiliza en la capa final de la red para producir una distribución de probabilidad sobre un conjunto de clases. La función softmax toma una serie de valores de entrada y los transforma en valores de salida que suman 1 y varían entre 0 y 1. Cada valor de salida representa la probabilidad de que la entrada pertenezca a una clase determinada. Por ejemplo, si un modelo tiene tres clases, la función softmax produciría tres valores de salida que suman 1 y representan las probabilidades de que la entrada pertenezca a cada una de las tres clases.

Otras de las funciones de activación más comunes utilizadas en las redes neuronales incluyen:

Función **sigmoide**: La función sigmoide tiene una forma de S y produce una salida que varía entre 0 y 1. Esto es útil para modelos que necesitan producir una salida binaria, como 0 o 1.

Función tangente hiperbólica (**tanh**): La función tangente hiperbólica produce una salida que varía entre -1 y 1. Esto puede ser útil en problemas en los que se necesitan salidas de amplio rango.

Función **ReLU** (Rectified Linear Unit): La función ReLU produce una salida igual a la entrada si es mayor que cero y cero en caso contrario. Esto puede ayudar a mejorar la velocidad de entrenamiento y el rendimiento del modelo en muchos problemas.

Función **Leaky ReLU**: La función Leaky ReLU es similar a la función ReLU, pero produce una salida pequeña en lugar de cero cuando la entrada es negativa. Esto puede ayudar a evitar el problema del "muerto" (dead) en las neuronas, en el que las neuronas no producen salida.

Pequeño resumen de algunos conceptos simples

Sample: Representa una única fila de un conjunto de datos.

Epoch: Denota el número de veces que el algoritmo opera en todo el conjunto de datos de entrenamiento.

Batch: Es el número de muestras a considerar para actualizar los parámetros del modelo.

Función de cost/función de loss: Una función de coste ayuda a calcular el coste, que representa la diferencia entre el valor real y el valor predicho. La función de loss mide el error entre la salida del modelo y los resultados esperados y se utiliza para evaluar el rendimiento del modelo.

Learning rate: Ofrece un grado que denota cuánto deben actualizarse los pesos del modelo.

TEMA 5: Convolutional Neural Network (CNN)

CNN (Convolutional Neural Network) es un tipo de modelo de aprendizaje automático de inteligencia artificial que se utiliza comúnmente en tareas de procesamiento de imagen y visión por computadora. Estos modelos se basan en la idea de las redes neuronales artificiales y utilizan capas de neuronas conectadas entre sí que se entrenan para realizar tareas específicas.

La característica distintiva de las CNN es su uso de capas de convulsión, que son capas de neuronas que se aplican a regiones pequeñas de una imagen y comparten pesos. Esto permite que las CNN aprendan características específicas de una imagen y las utilicen para realizar tareas como la clasificación de imágenes o la detección de objetos.

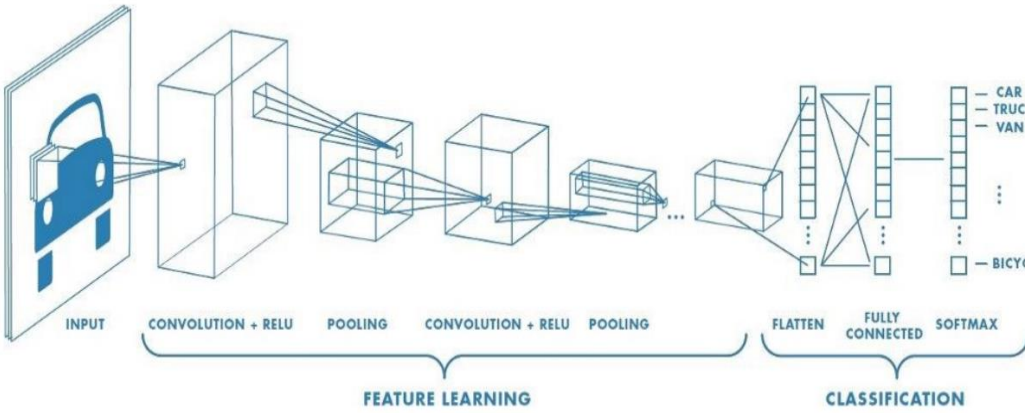
Las capas de neuronas que se aplican a regiones pequeñas de una imagen se conocen como capas de **convulsión** (convolución, en español). La operación de convulsión consiste en aplicar un conjunto de filtros a la imagen y producir una salida que es una versión transformada de la imagen de entrada. Cada filtro es una matriz pequeña de pesos que se aplica a la imagen y produce una salida que es una versión "enfatizada" de una característica específica de la imagen.

Por ejemplo, si se utiliza un filtro que enfatiza las líneas verticales en una imagen, la salida del filtro será una imagen en la que las líneas verticales son más pronunciadas y el resto de la imagen está atenuada. Al aplicar varios filtros a una imagen, se pueden extraer diferentes características de la imagen y utilizarlas para realizar tareas como la clasificación o la detección de objetos.

La ventaja de utilizar capas de convulsión en lugar de capas tradicionales de redes neuronales es que permiten a las CNN aprender características específicas de una imagen de manera automática, en lugar de tener que diseñar manualmente las características a utilizar.

Maxpooling: El maxpooling (maximo pooling, en español) es una técnica utilizada comúnmente en las redes neuronales convolucionales (CNN, por sus siglas en inglés) para reducir la dimensión de las características aprendidas por las capas de convulsión.

El proceso de maxpooling se realiza mediante la aplicación de una ventana móvil a lo largo de la imagen y, en cada posición, se toma el valor máximo de los pixeles dentro de la ventana. De esta manera, se reduce la resolución de la imagen y se eliminan algunos detalles menos importantes. La operación de maxpooling consiste en dividir la imagen de entrada en regiones pequeñas y tomar el valor máximo de cada región. Esto produce una salida que es una versión reducida de la imagen de entrada en la que se han eliminado algunos detalles y se han resaltado las características más prominentes.



Flattening: El término "flattening" se refiere al proceso de convertir una matriz multidimensional en una matriz de una sola dimensión. En el contexto de redes neuronales convolutional (CNN, por sus siglas en inglés), el flattening se utiliza a menudo después de que se han aplicado varias capas de procesamiento de imagen, como filtros y maxpooling.

La idea detrás del flattening es que, después de aplicar varias capas de procesamiento de imagen, la salida de la red se ha convertido en un conjunto de características abstractas que representan la imagen de entrada de manera más generalizada. Al aplicar el flattening, se convierten estas características abstractas en una sola fila de valores, que luego pueden ser utilizados como entrada para una capa de clasificación. Hay que convertir la salida de la parte convolucional de la CNN en un vector de características 1D, para que lo utilice la parte ANN (capa densa) de la misma. Esta operación se denomina flattening. Obtiene la salida de las capas convolucionales, aplana toda su estructura para crear un único vector de características largo que será utilizado por la capa densa para la clasificación final.

Dense layer: Un "dense layer" o capa densamente conectada es un tipo de capa en una red neuronal que está completamente conectada a todas las neuronas de la capa anterior. La capa densamente conectada es una capa importante en una CNN que se utiliza para realizar la clasificación final de la imagen utilizando características abstractas extraídas por las capas de procesamiento de imagen anteriores. No es más que un clasificador de red neuronal artificial (RNA) (artificial neural network ANN).

Proceso:

En una red neuronal convolutional (CNN, por sus siglas en inglés), la capa densamente conectada se encuentra generalmente después de varias capas de procesamiento de imagen, como filtros y maxpooling. La salida de estas capas se pasa a través del flattening, que convierte la salida en una matriz de una sola dimensión, y luego se pasa a la capa densamente conectada.

La capa densamente conectada es utilizada para realizar la clasificación final de la imagen. A menudo, se utiliza una función de activación como la función de softmax en la capa de salida para producir una salida de probabilidades para cada clase posible. Estas probabilidades pueden utilizarse para determinar a qué clase pertenece la imagen de entrada con mayor probabilidad.

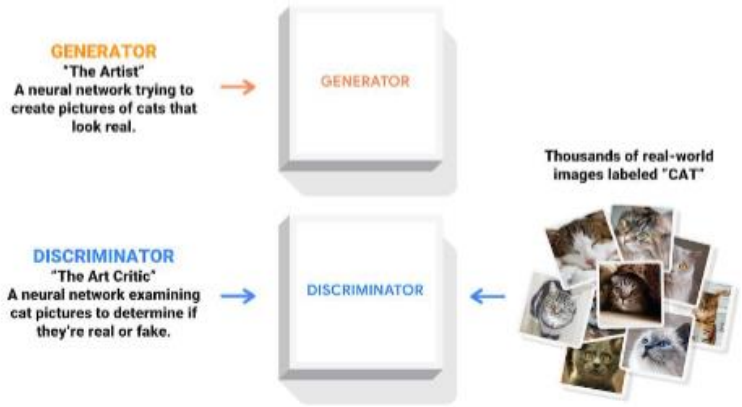
TEMA 8: Generative Adversarial Networks (GAN)

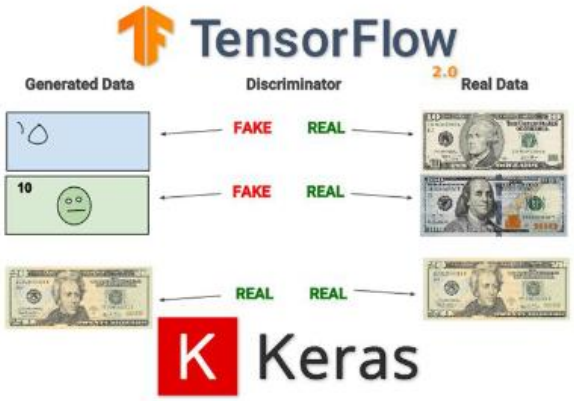
Las Redes Generativas Antagónicas (RGAs), también conocidas como GANs en inglés, son una clase de algoritmos de inteligencia artificial que **se utilizan en el aprendizaje no supervisado, implementadas por un sistema de dos redes neuronales que compiten mutuamente**. **Las redes adversariales generativas** (**GAN**, por sus siglas en inglés) son un tipo de modelo de aprendizaje automático que se utiliza para generar contenido nuevo, como imágenes, música o texto. Una GAN consta de dos redes neuronales que trabajan juntas: una generadora y una discriminadora.

La red generadora tiene como objetivo generar contenido nuevo que se asemeje lo más posible a un conjunto de datos reales, mientras que la red discriminadora tiene como objetivo distinguir entre el contenido real y el contenido generado por la red generadora. Estas dos redes juegan un juego de "suma cero" entre sí, en el que la red generadora trata de engañar a la red discriminadora para que no pueda distinguir el contenido generado de forma real, mientras que la red discriminadora trata de mejorar constantemente su capacidad para distinguir el contenido generado del real.

A medida que ambas redes entrenan juntas, la red generadora mejora su capacidad para generar contenido que se asemeje cada vez más al contenido real, mientras que la red discriminadora mejora su capacidad para distinguir el contenido real del generado. El resultado final es una red generadora que es capaz de generar contenido nuevo que se asemeja de manera realista al conjunto de datos reales utilizado para entrenar la GAN.

La red generadora cada vez será mejor, puesto que irá aprendiendo de lo que va “marcando” la discriminadora. Es una especie de target iterativo en el que las dos redes aprenden la una de la otra y compiten por hacer el mejor “objeto falso” (porque no forma parte del conjunto de entrenamiento, aunque se base en él).





Aspectos técnicos de cómo funciona:

El término "**conv2d**" se refiere a una capa de convolución de dos dimensiones en una red neuronal. Una capa de convolución se utiliza para aplicar filtros a una imagen o matriz de datos, lo que permite extraer características de la imagen.

Un "**conv2d**" se aplica a cada canal de la imagen de entrada y se utiliza un conjunto diferente de pesos para cada canal. Esto permite a la capa de convolución aprender características diferentes en cada canal de la imagen de entrada. Por ejemplo, si la imagen de entrada es en escala de grises, solo habría un canal y se utilizaría un único conjunto de pesos. Si la imagen de entrada es en color, habría tres canales (rojo, verde y azul) y se utilizaría un conjunto de pesos diferente para cada canal.

Por otro lado, el término "**conv2dtranspose**" se refiere a una capa de deconvolución de dos dimensiones en una red neuronal. Una capa de deconvolución es similar a una capa de convolución, pero en lugar de aplicar filtros a la imagen de entrada, se utiliza para aumentar la resolución de una imagen. Esto se logra mediante el uso de pesos que "invierten" la operación de la capa de convolución.

En el contexto de las redes adversariales generativas (GAN, por sus siglas en inglés), las capas de convolución y deconvolución se utilizan a menudo en las redes generadoras y discriminadoras, respectivamente. La **red generadora utiliza capas de deconvolución para aumentar la resolución de la imagen generada y hacerla más realista, mientras que la red discriminadora utiliza capas de convolución para extraer características de la imagen y determinar si es real o generada**.

Tenemos **dos funciones de loss: la del generador y la del discriminador**. Y los optimizadores serán diferentes (no quiere decir que sean diferente algoritmo, sino que cada uno debe trabajar por separado). De hecho, estamos entrenando dos redes.

El proceso de entrenamiento no podrá utilizar los socorridos .compile y .fit, hay que usar funciones de bajo nivel de Tensorflow para hacer los procesos de optimización. Para ello usaremos:

- El decorador @tf.function para “compilar” el grafo de operaciones

- tf.GradientTape para guardar los gradientes en cada paso

Radford (2015): Radford et al. (2015) es un paper científico titulado "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks" (Aprendizaje de representaciones no supervisadas con redes adversariales generativas profundas de convolución), escrito por Alec Radford, Luke Metz y Soumith Chintala. En este trabajo, los autores presentan una nueva versión de las redes adversariales generativas (GAN, por sus siglas en inglés) que utiliza una arquitectura de redes neuronales profundas de convolución para aprender a generar imágenes realistas sin necesidad de etiquetas de clase.

Radford et al. recomiendan la siguiente arquitectura para tener GANs estables:

- Reemplazar los layers de pooling con strided convolutions (layer Conv2D conel parámetro stride)

- Utilizar batch normalization en red generadora y discriminadora

- Eliminar los layers fully-connected (Dense)

- Usar ReLU en el generador excepto en capa final que usará tanh

- Usar Leaky ReLU en el discriminador