Dip lenin Cheatsheet

- ¿Qué ventajas tiene la utilización de NumPy?

  Permite realizar operaciones matriciales que de otro modo deberían realizarse con de forma iterativas.

- ¿Para qué es necesario Normalizar los valores en un dataset?

  Para facilitar el entrenamiento de los modelos de Machine Learning.

- Luego de aplicar la normalización z-score:

  - Min-Max (0-1): poner todos los valores en un rango de 0-1. A cada una de las variables le resto el valor minimo y a eso lo normalizo por la diferencia entre el maximo y el minimo.

    ```
    x = (x - min(x)) / (max(x) - min(x))
    ```

    El mínimo termina siendo siempre 0 y el máximo, 1 para todas las variables.

  - z-score: a cada variable restarle la media de esa variable y normalizarlo por la desviación estándar.

    ```
    x = (x - mean(x)) / std(x)
    ```

    La media de cada distribución distribución queda en 0 (ya que le resté la media a cada variable) y la desviación estándar en 1 para todas las variables (esto se da porque normalizo por la distribución).

- ¿Qué hace la siguiente instrucción?: `dataframe["GRASA"].mean()`

  Calcula el promedio para el atributo "grasa" del dataframe. Al array *dataframe*, específicamente a la columna *GRASA* (ver que no especifica posición específica, por lo que entendemos que se trata de la columna), se le aplica la operación *mean* que es el promedio.

- ¿Qué hace la siguiente instrucción?: `(dataframe["TIPO"]=="CC").sum()`

  Calcula la cantidad de registros que en la columna "tipo" tiene el valor "cc". Al array *dataframe*, primero se le aplica un filtro para que solo considere los campos con el valor *CC*. Luego se aplica la operación de numpy *sum* sobre esto, que suma los registros y devuelve la cantidad.

- ¿Qué hacen las siguientes instrucciones?: `dataframe.name=pd.Categorical(dataframe.name); dataframe.name=dataframe.name.cat.codes`

  Reemplazan cada valor nominal en la columna "name" del dataframe por uno numérico entero distinto

  En la primera línea, se puede apreciar que se define la columna name del dataframe como categorical (es probable que tal columna contenga strings y que lo que se busque es traducir eso a integers para el procesamiento). Una vez convertido a categorical, la segunda línea le asigna a cada categoría (es decir, cada nombre) un número que lo identificará. De forma gráfica:

name	size	weight
caniche	2	10.3
caniche	2	12.3
sharpay	5	20.4
labrador	9	50.2
pequines	3	25.9
pequines	5	27.4

Name contiene a caniche, sharpay, labrador y pequines. El primer paso es definir la columna como Categorical. El segundo paso, es asignarle índices. La asignación podría ser la que sigue:

```
caniche = 0
sharpay = 1
labrador = 2
pequines = 3
```

La tabla, entonces, quedaría así:

name	size	weight
0	2	10.3
0	2	12.3
1	5	20.4
2	9	50.2
3	3	25.9
3	5	27.4

- Si un conjunto de datos tiene 20 ejemplos y 3 variables de entrada, para predecir una sola variable de salida ¿Cuál es es el tamaño de w?

  3x1. La clave está en que w es (*variables_de_entrada* x *variables_de_salida*), en este caso, 3x1.

- Si un conjunto de datos tiene 20 ejemplos y 3 variables de entrada, para predecir una sola variable de salida ¿Cuál es es el tamaño de b?

  Escalar (1x1). b se calcula siempre como (*variables_de_salida* x 1), en este caso, 1x1.

- Dada la siguiente salida de un modelo de regresión lineal junto con los valores esperados, calcule el error cuadrático medio (MSE).

Salida de la red	Valor esperado o verdadero
2	-1
1	5
-3	-2
0	1

Las diferencias de cada fila son 3, 4, 1 y 1. Eso lo elevo al cuadrado, quedándome 3^2, 4^2, 1^2 y 1^2. Luego, la suma de diferencias al cuadrado es 9+16+1+1=27. Como hay 4 ejemplos, el promedio es 27/4 = 6.75.

- Si tengo un parámetro w y su derivada respecto del error δE/δw, y una tasa de aprendizaje α, ¿cómo debo actualizar a w en base a la derivada, si estoy utilizando descenso de gradiente?

  w = w - α δE/δw. La fórmula a usar será la que sigue:

  ```
  parámetro = parámetro - (tasa_de_aprendizaje x derivada_respecto_error)
  ```

- Cuando termina el algoritmo de descenso de gradiente para regresión lineal utilizando error cuadrático medio, siempre se encuentra el mínimo global de la función de error.

  Falso, si bien siempre existe un mínimo global no siempre se encuentra exactamente. Esto es porque el algoritmo se detiene cuando encuentra que la función crece. Una función tiene muchos picos, no es posible asegurar que ese sea el mínimo global.

- En el descenso de gradiente estocástico, si tengo 100 ejemplos, y un tamaño de lote (batch_size) de 5 ejemplos ¿Cuántas iteraciones (actualizaciones de parámetros) se realizan en una época?

  20. N = 100; batch_size = 5. Las iteraciones reales por epoch están definidas como `N / batch_size`, en este caso 100/5 = 20.

  Iteraciones reales totales: `N / batch_size * epochs`.

- Si un modelo obtiene un accuracy de 1 en el conjunto de entrenamiento y también en el conjunto de prueba

  No tiene sobreajuste (overfitting). Para que se de el overfitting, tiene que haber un error bajo en training pero alto en testing. En este caso, el accuracy es perfecto tanto en el conjunto de training como en el de testing.

- Seleccione verdaderas:

  - Las redes neuronales “aprenden” mediante un algoritmo que intenta minimizar una función de error.

    No hay mucho para acotar...

  - Una red neuronal con muchas capas y muchas neuronas por capa podría generar un sobreajuste, ya que modelan polinomios complejos.

Esto es asi ya que al ser cada vez mas complejo y tener tantas variables, en el intento de cubrir todo en el training, en realidad podria caer en errores en el testing.

- Las redes neuronales son sensibles a la escala de los datos.

  Cuantos mas datos, mas complejo el asunto.

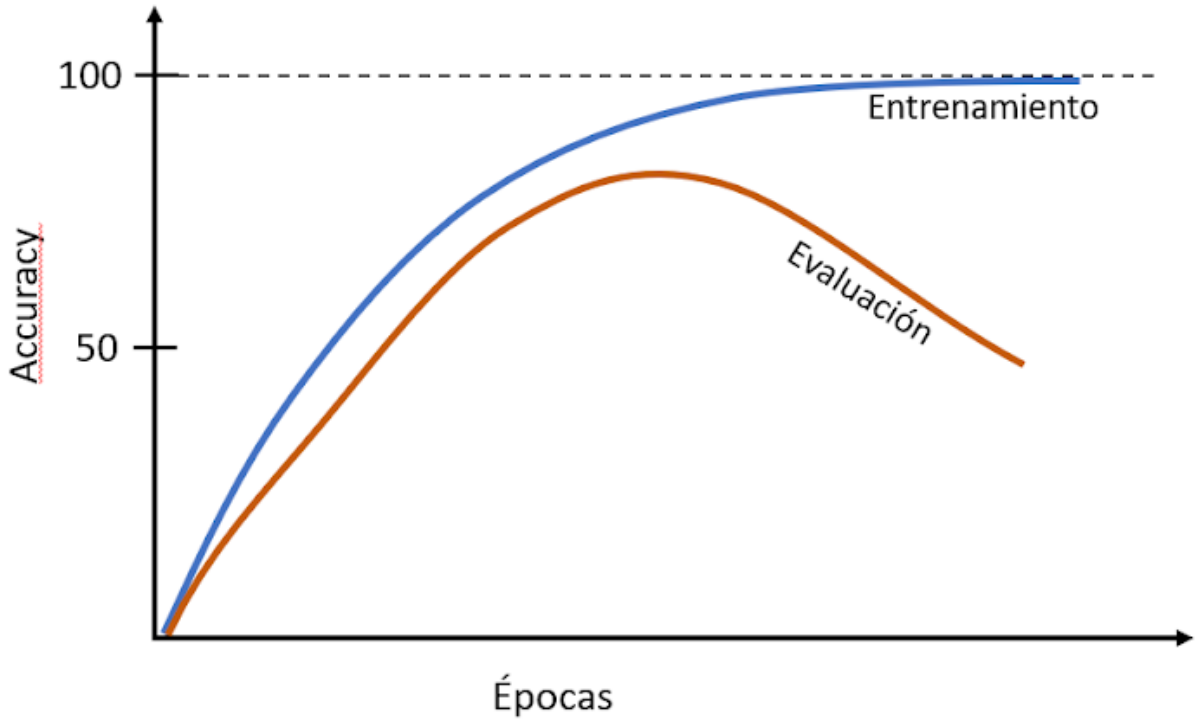
- La capa de salida de una red neuronal tiene tantas neuronas como clases se desee clasificar.

  Por cada neurona, un tipo de clasificacion diferente.

- La métrica “Accuracy” (pred. correctas/total) es susceptible al desbalance de clases.

  Si el dataset esta desbalanceado, el accuracy podria variar mucho porque se basa en una parte de la matriz sin ver la otra parte.

- Dado el siguiente gráfico que muestra las curvas de accuracy en función de las épocas para el conjunto de entrenamiento y evaluación, elija la respuesta correcta sobre el modelo entrenado al finalizar el entrenamiento.



El modelo sobreajusta (overfitting) al conjunto de entrenamiento y no será capaz de clasificar nuevos datos del mundo real. Esto se puede ver en la curva de entrenamiento que roza lo perfecto y la curva de evaluacion que tiende a ser baja.

- Dada la siguiente matriz de confusión, calcule la métrica Accuracy. Escriba el resultado como un número entre 0 y 1.

True Labels	Clase 1	Clase 2	Clase 3
Clase 1	10	1	3
Clase 2	10	10	1
Clase 3	5	5	5

Predicted Labels

0.5. En la diagonal encontramos los ejemplos clasificados correctamente. Entonces son 10+10+5=25. Sumando el total de ejemplos (todas las celdas) hay 50. Entonces hay 25/50 de accuracy, o sea 1/2 o 0.5.

- Dada la siguiente salida de una red neuronal junto con los valores esperados, calcule la métrica Recall

Salida de la red	Valor esperado o verdadero
0	0
1	0
0	0
0	0
0	0
0	0
0	1
0	1
1	1

0.3. Precision se define como TP/(TP+FN). En este caso, TP+FN son los casos en los que los datos eran originalmente de la clase 1, o sea TP+FN=3. TP es la cantidad de aciertos de la clase 1, que en este caso fue una sola. Entonces precision=1/3= 0.33

- Dada una red para reconocer 10 tipos de animales ¿Qué tipo de función de activación final y de error deben utilizarse?

Tipo	Min.	Max.	Error	Problema
relu(x)	0	inf.	Error cuadrático medio	Regresión
id(x)	menos inf.	inf.	Error cuadrático medio	Regresión
sigmoidea(x)	0	1	Entropía cruzada binaria	Clasificación
tanh(x)	-1	1	Error cuadrático medio	Regresión
softmax(x)	0	1	Entropía cruzada	Clasificación

NOTA: recordar que la softmax normaliza los valores para que sea más fácil clasificar.

Sabiendo lo anterior, hay que pensar qué tipo de problema queremos resolver. En este caso, lo que se busca es reconocer 10 tipos de animales, es decir, clasificarlos, de ahí es que usaría una softmax con entropía cruzada.

- Dada una red para calcular la edad de un animal en base a sus características ¿Qué tipo de función de activación final y de error deben utilizarse?

  Tomando de base la tabla anterior, dado que queremos generar un número real mayor a 0 (la edad), es un problema de regresión y por ende la función de error que corresponde es el ECM(MSE en inglés). Por otro lado, podemos utilizar una función de activación ReLu para predecir siempre valores mayores a 0, o directamente la salida de la red sin ninguna activación.

- Dado un ejemplo cuya etiqueta de clase es 2, si tengo 5 clases, ¿cómo sería su codificación one-hot?

  (0,1,0,0,0). La notacion one-hot utiliza solo 0 y 1. En la etiqueta va el 1, el resto en 0. Dado que hay 5 clases es que es un quinteto.

- Dada una imagen en escala de grises con dimensiones 10x10, si aplico un filtro convolucional de tamaño 5x5 sin padding ¿Cuáles son las dimensiones espaciales (alto x ancho) de la imagen de salida?

  Hay una fórmula para esto:

  ```
  H'=H+2*P-(K-1)
  W'=W+2*P-(K-1)

  Siendo que H es altura, P es padding y K es kernel. Todo depende de si lo
  que se busca es el alto o el ancho, respectivamente.
  ```

  En nuestro caso, H=10, P=0 y K=5. Podríamos haber también usando W en vez de H, pero com ambos valores son iguales es indiferente. Reemplazando en la fórmula: H'=10+2*0-(5-1) --> H'=10+0-4 --> H'=6. Por lo tanto, la respuesta es 6x6.

- Dada una imagen RGB con dimensiones 10x10x3, si aplico un filtro convolucional de tamaño 3x3 con padding de 1 pixel, ¿cuáles son las dimensiones espaciales (alto x ancho) de la imagen de salida?

  Nuevamente, aplicamos la fórmula. En este caso, H=10, P=1 y K=3. Reemplazando en la fórmula: H'=10+2*1-(3-1) --> H'=10+2-2 --> H'=10. Por lo tanto, la respuesta es 10x10.

- Dada una capa convolucional con 8 filtros de 5x5, que recibe como entrada una imagen RGB de 10x10x3, suponiendo que se utiliza padding de 2 pixeles, ¿Qué tamaño tendrá el Feature Map resultante de la capa?

  Nuevamente, aplicamos la fórmula pero con una consideración: hay más de un filtro, y esto se nota de manera diferente. Primero la fórmula: H=10, P=2 y K=5. Reemplazando en la fórmula: H'=10+2*2-(5-1) --> H'=10+4-4 --> H'=10. Por lo tanto, la respuesta es 10x10. Ahora bien, son 8 filtros, por lo que la notación final será 10x10x8.

- Una imagen RGB de tamaño 5x4 ¿Cuántos valores necesita para representarse?

  Si la imagen tiene de tamaño 5x4 = 20px, cada uno requiere 3 valores para representarse (por el RGB), 20x3=60.

- Una imagen en escala de grises de tamaño 10x10 pixeles ¿cuántos valores necesita para representarse en memoria?

  Igual que antes: si el tamaño es 10x10 = 100px. Ahora bien, en vez de usar RGB estamos usando escala de grises, por lo que con un byte nos basta (2^8 = 256 valores para representar -> 8 bits -> 1 byte), por lo que multiplicamos al 100 x 1, y nos queda 100.

- Si una capa convolucional recibe feature maps de tamaño HxWxC y devuelve feature maps de tamaño HxWxD ¿Cuál es el tamaño del kernel?

  El tamaño del kernel no tiene nada que ver con la cantidad de filtros!

- Si una capa convolucional recibe feature maps de tamaño HxWxC y devuelve feature maps de tamaño HxWxD ¿Cómo se definiría en Keras?

  Conv2D(D). En Keras solo es necesario especificar la cantidad de filtros de salida, la capa conoce la cantidad de filtros de entrada automáticamente.

- Dada una imagen en escala de grises con dimensiones 10x10, si aplico un filtro convolucional de tamaño 5x5 con padding de 2 pixeles ¿Cuáles son las dimensiones espaciales (alto x ancho) de la imagen de salida?

  Nuevamente, aplicamos la fórmula. En este caso, H=10, P=2 y K=5. Reemplazando en la fórmula: H'=10+2*2-(5-1) --> H'=10+4-4 --> H'=10. Por lo tanto, la respuesta es 10x10.