Regresión Logística

Se desea desarrollar un clasificador de imágenes de gatos y perros utilizando regresión logística. Para ello descargar la base de datos `dog-and-cat-classification-dataset` de kaggle. ☞: La siguiente instrucción puede ayudar con la descarga:
`kagglehub.dataset_download("bhavikjikadara/dog-and-cat-classification-dataset")`

(**a**) *Análisis:*

1. Calcular la función inversa $\sigma^{-1}(p)$ con $p \in (0, 1)$.
2. Sea $p = \sigma(z)$ la función sigmoide, calcular la derivada $\sigma'(z)$. ☞: Me parece que si se expresa el resultado en función de p el resultado se simplifica bastante.
3. Hallar una expresión analítica para la función costo y su gradiente. ☞: Tenga en cuenta el modelo asociado a una regresión logística de dos clases.

(**b**) *Pre-Procesamiento:*

1. Las imágenes poseen diferentes tamaños. Convertirlas todas a 16×16 utilizando `resize` (PIL).
2. Las imágenes también poseen diferentes formatos. Convertirlas todas a escala de grises.
3. Mostrar 5 imágenes.
4. Utilice el comando `train_test_split` (sklearn) para definir dos conjuntos de datos. El conjunto de entrenamiento debe contener 20000 muestras, el resto serán de testeo.

(**c**) *Clasificación:*

1. Utilizando `LogisticRegression` (sklearn), realizar una regresión logística sin regularización.
2. Implementar una función que permita calcular la *accuracy* a partir de la salida de `predict`. Reportar el accuracy de entrenamiento y testeo.
3. Implementar una función que permita calcular la *cross-entropy* a partir de la salida de `predict_log_proba`. Reportar la cross-entropy de entrenamiento y testeo. ☞: Por un tema de continuidad asuma que $p \log(q) = 0$ si $p = q = 0$.
4. Crear una imagen de 16×16 a partir del consejero de la guía ☞ y clasificarlo.

(**d**) *Regularización:* Utilice un mapa polinómico de orden 2 y un término de penalización para regularizar el problema. Indicar el *accuracy* tanto para el entrenamiento como el testeo. ☞: Una regresión de muchos parámetros puede tardar un tiempo. Un buen tip es configurar `max_iter` en un valor muy bajo para probar el código y una vez funcionando darle un valor razonable.

(**e**) A partir de la salida de `predict_proba` del conjunto de testeo, implementar la curva ROC. Comparar las ROC de los clasificadores con y sin regularización con la de uno que decide al azar.