Tarea 3: Métodos de clasificación.

Autores: Bruno Hernández, José Cornejo

1. Clasificador multiclases

Para $K > 2$, se define para una observación $x_0 \in \mathbb{R}^d$ el clasificador multiclase:

$$C(x_0) = \arg \max_{1,\cdots,K} \; y_i(x_0), \tag{1}$$

donde $y_i(x_0)$ son funciones afínes de la forma:

$$y_i(x) = a_i^T x + b_i, \tag{2}$$

que corresponden a K clasificadores lineales.

Podemos notar que, para cada nueva observación $x \in \mathbb{R}^d$, el conjunto $\{y_1(x), \cdots, y_K(x)\}$ es un conjunto finito, por lo tanto el valor máximo siempre ha de existir, implicando que el índice que llega a ese máximo existe también. Así, cada región del espacio es clasificable en al menos una clase.

Sin embargo, si al menos uno de los a_i es difente a otro, entonces existen puntos en los cuales hiperplanos se intersectan. En tal caso, existirán restas (en rigos, otroa hiperplanos de dimensión $d-2$) en que el máximo pueda alcanzarse en más de una función afín (los vertices del poliedro generado por las K funciones). Es posible demostrar que esta instersección compete a sólo 2 de estas funciones (si en un hiperplano intersecta una función extra, se pude demostrara que alguna de ellas es descartable pues no alcanza el máximo en ningún punto, por lo que no aporta información al modelo), entonces el método de decisión puede ser uno aleatorio, digamos *Bernoullie(1/2)*, donde con la misma probabilidad puedo pertenecer a cualquiera de las dos clases.

Para el caso $K = 2$, tenemos sólo dos funciones afínes, por lo tanto: 1.- estas no pueden ser paralelas, en tal caso una siempre es mayor que la otra y todo el espacio es clasificado a una sola clase, 2.- si las rectas no son paralelas entonces existe H, hiperplano, en donde las rectas se crucen y se tiene que este hiperplano es la región de decisión para las dos clases, volviendo al caso binario.

2. Detección de barcos en imágenes satelitales

La siguiente base de datos fue extraída del archivo `shipsnet.json`.

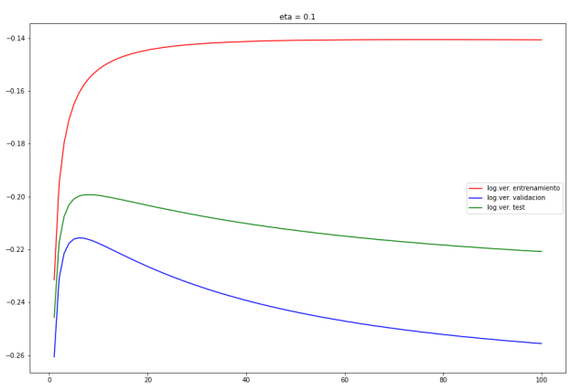
a. **Obtención de la data:** Los datos fueron extraídos formando un *DataFrame* que contenía 4000 observaciones. Cada observación consistía en una imagen en el formato de un vector de tamaño 19200 y un indicador del barco (1 si se aprecia un barco en la imágen y 0 si no). Las imágenes fueron reordenadas formando una matrix de datos de dimensiones $80 \times 80 \times 3$ (imágenes RGB de dimensiones 80×80) para luego ser procesadas mediante el método *HoG*y así obtener un vector de 128 caracteristicas representantes para cada una de las 4000 imágenes.

   Finalmente, la función `train_test_split` de la librería `sklearn.model_selection`, nos permitió seleccionar 3 conjuntos, conjunto *Test*, *Train* y *Validación*, de manera aleatoria con las proporciones 70 %, 15 % y 15 % respectivamente.

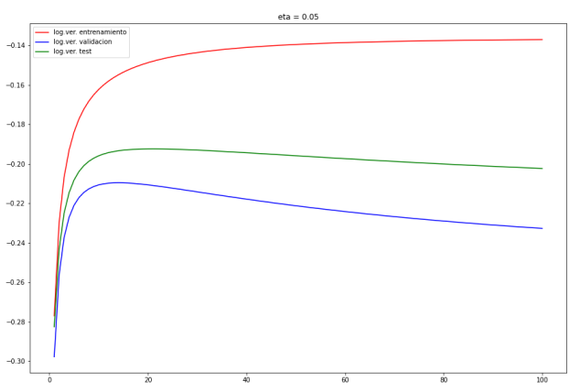
b. **Clasificación mediante regresión logística:** Se creó una clase de python llamada `MyLogisticRegression` conteniendo los siguientes métodos:

   - `___init___`, método iniciador de la clase que recibe los valores η y *epochs* (*learning rate* y número de iteraciones (épocas), respectivamente) que almacena los parámetros necesarios para la regresión.
   - `fit`, método entrenador del modelo que recibe la matriz de datos `X` y el vector de etiquetas `y`, y que almacena el parámetro w encontrado al maximizar la función de *log-verosimilitud* mediante el algoritmo de gradiente estocástico.
   - `predict`, función que recibe una matriz de observaciones y retorna las predicciones estimadas por el modelo según los parámetros encontrados en el método `fit`.
   - `log_verosimilitud`, función que recibe una matriz de datos y sus respectivas etiquetas de clase y retorna el valor de la *log-verosimilitud* asociada a tales observaciones.

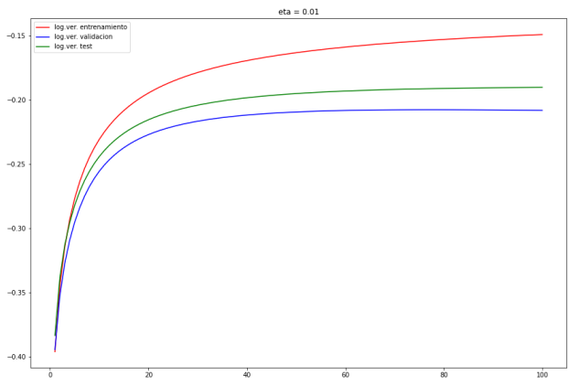
   Para los valores $\eta \in \{0{,}01, 0{,}05, 0{,}1\}$ se bosquejó el cambio de la *log-verosimilitud* en función de la cantidad de iteraciones (*epochs*). Como muestra la Figura 1c, para el valor de η más alto (0.1) el incremento de la curva es abrúptamente más alto en el conjunto de entrenamiento en comparación con los conjuntos de testeo o validación. Resultado que no es sorprendente dado que, por un lado es el conjunto de entrenamiento quién aporta información al modelo, y por el otro lado tenemos que la taza de entrenamiento es más alta, por lo que los incrementos más áltos hacia el valor óptimo se harán presente en las primeras iteraciones.



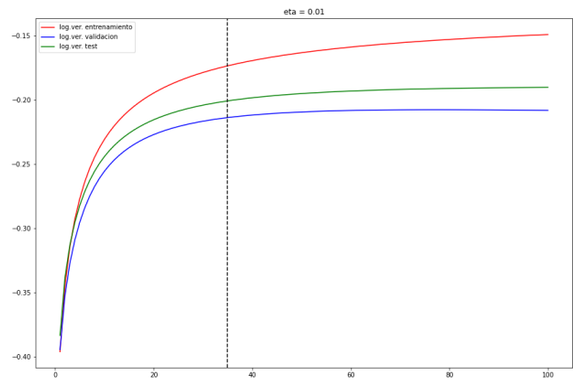
(a) $\eta = 0{,}1$



(b) $\eta = 0{,}05$



(c) $\eta = 0{,}01$



(d) $\eta = 0{,}01$, valor óptimo $= 35$.

Figura 1: En el eje horizontal la cantidad de iteraciones, en el eje vertical el valor calculado de la log-verosimilitud.

El criterio con el que se eligieron los parámetros óptimos $(\eta, epochs)$ serán aquellos que alcancen mayor valor de la función de *log-verosimilitud*, pero a la vez, aquellas curvas que hayan alcanzado valores parecidos en cada cantidad de iteración. Por ejemplo, en la Figura 1a, la curva de log-vorisumulitud del conjunto de entrenamiento alcanza un valor alto en pocas iteraciones, más aún, este valor llega a ser superior a la logverosimilitud de todas las demáses curvas en cualquier cantidad de *epochs*, sin embargo, comparativamente con los conjuntos de testeo y entrenamiento, es considerablemnte diferente. Esto nos hace pensar que el modelo entrenó muy de buena forma para las observaciones entregadas, pero no es consistente con alguna otras muestras obtenidas con la misma distribución. En este sentido hablamos que el modelo generó un sobreajuste (*Overfitting*).

De este modo, el parámetros que cumple mejor este criterio es $\eta = 0{,}01$. Para decidir que *epochs* es necesario para que el modelo quede 'bien' entrenado comparamos la variación del error cuadrático medio en función del aumento de la logverosimilitud para valores altos. Así, el valor establecido fue $epochs = 35$. (Figura 1d)

c. **Implementación del determinante lineal de Fisher:** Se crea la clase `MyFLDA`, con los métodos `___init___` y `predict`, análogos a la calse `MyLogisticRegression`, pero aplicados al algoritmo que implementa el determinante lineal de Fisher. Para su creación fue esencial la librería `NumPy` y sus funciones de álgebra lineal.

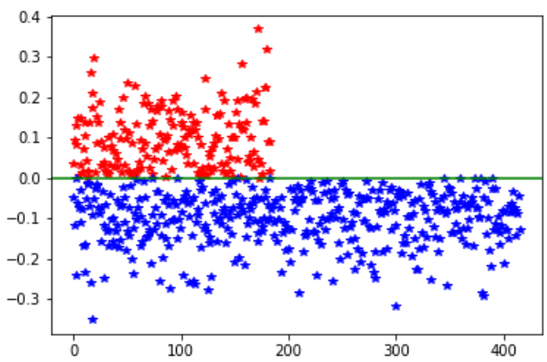


Figura 2: Valores del determinante de Fisher para el conjunto Test.

d. **Comparación de modelos:** Compararemos los modelos mediante el *error cuadrático medio* (ECM). Para los conjuntos de testeo y validación. Considerando el modelo del determinante de Fisher, y el modelo de regresión logística de parámetros $\eta = 0{,}01$ y $epochs = 35$.

Tabla 1: Tabla comparativa de los ECM's para cada conjunto y cada modelo.

	Test	Validación
Regresión logística	0.088	0.078
Det. de Fisher	0.138	0.13

A pesar de asumir distribución en los datos, podemos ver que el modelo logístico mantiene un valor menor de error, por lo tanto un porcentaje mayor de acierto en cada predicción. De todos modos, podemos concluír que ambos modelos resultaron ser consistentes puesto que para la misma cantidad de datos obtuvieron valores de errores parecidos. Además, como los errores fueron calculados con conjuntos disjuntos de la base de entrenaminto, estos valores pueden contar como una medida de sesgo medio para cualquier muestra que conserve la misma distribución.