Tarea 3: Clasificación

Profesor: Felipe Tobar
Auxiliares: José Díaz, Diego Garrido, Jou-Hui Ho, Luis Muñoz
Consultas: Diego Garrido.
Fecha entrega: 10/7/2019
Formato entrega: En grupos máximo dos integrantes entregue un informe en formato PDF con una **extensión máxima de 2 páginas solo las P1 y P2**. Este informe debe presentar y analizar sus resultados y detallar la metodología utilizada. Adicionalmente, usted debe entregar un Jupyter Notebook con los códigos que creó para resolver la tarea.

P1. Clasificador multiclase (1.0 pto)
En clases se vio que es posible extender la formulación de la clasificación lineal binaria para $k > 2$ clases utilizando k funciones lineales de la forma

$$y_k = a^\top x + b,$$

y luego asignar la clase a una muestra x mediante $k^\star = \arg\max_k y_k$. Muestre que este clasificador multiclase es coherente a diferencia de los métodos *one-versus-all* y *one-versus-one*. En particular, muestre que usando este clasificador propuesto:

- No quedan regiones del espacio de entrada sin clasificar.
- No existen regiones del espacio de entrada asignadas a más de una clase.
- Eligiendo $k = 2$, este criterio colapsa al presentado para el caso binario.
- Muestre cómo es la región de decisión y el subconjunto correspondiente a cada clase.

P2. Detección de barcos en imágenes satelitales (4 ptos)

El objetivo de esta pregunta es ayudar a abordar la difícil tarea de detectar la ubicación de grandes barcos usando imágenes satelitales. La importancia de esta tarea es múltiple, en particular, la detección automática de barcos podría facilitar el monitoreo de los niveles de actividad del puerto y el análisis de la cadena de suministro. El dataset disponible consiste de imágenes satelitales de barcos recopiladas en las áreas de la bahía de San Francisco y la bahía de San Pedro de California. Este incluye 4000 imágenes 80x80 RGB, con la etiqueta "ship" (1) y "no-ship" (0). Más detalles sobre cómo cargar los datos y como extraer vectores de características a partir de una imagen en el notebook **tutorial.ipynb**. **Observación:** en el informe solo debe responder las preguntas planteadas, no adjunte código y debe ser lo más breve posible en los puntos a) y c).

a) (0.5 ptos.) Cargue los datos y extraiga los vectores de características (vea **tutorial.ipynb**), pues las siguientes partes de esta pregunta serán todas en base a las características, no a las *entradas crudas*. Divida los datos de forma aleatoria en conjuntos de entrenamiento (70%), validación (15%) y testeo (15%).

   Observación: Le puede ser útil la función *train_test_split* de *sklearn*.

b) (1.5 ptos.) Implemente la regresión logística bajo el método del descenso del gradiente estocástico (ver sección 4.5.2 del apunte). Su implementación debe incluir dos hiperparámetros: la tasa de aprendizaje η, y el número de épocas de entrenamiento e, donde una época corresponde a una pasada por todo el conjunto de entrenamiento (pues recuerde que el gradiente estocástico solo usa una observación a la vez).
   Grafique la log-verosimilitud normalizada por el número de observaciones (eje y) versus el número de épocas

(eje x), tanto para el conjunto de entrenamiento como el de validación para distintas tasas de aprendizaje, para ello considere $\eta \in \{0.01, 0.05, 0.1\}$ y épocas $e \in \{1, \ldots, 100\}$. Luego, en base a los gráficos responda la siguiente pregunta: ¿Cuál es la configuración óptima de η y *épocas*? **Justifique su elección y explique qué pasa con tazas de aprendizaje muy grandes o pequeñas**.
Observaciones: La clase anterior debe ser escrita usando el stack de Python y Numpy, no está permitido el uso de algún solver para optimizar la log-verosimilitud. El archivo **tutorial.ipynb** cuenta con una plantilla a modo de guía para su implementación.

c) (1.0) Implemente el discriminante lineal de Fisher. En la sección 4.3 del apunte se describe cómo obtener a, con lo que luego podemos definir un umbral b para asignar un x a $\mathcal{C}_1$ si $a^T x \geq -b$ y a $\mathcal{C}_2$ en caso contrario. Para efectos de esta pregunta considere:

$$-b = a^\top \frac{1}{2}(\mu_1 + \mu_2) = \frac{1}{2}(m_1 + m_2)$$

.
Observaciones: La clase anterior debe ser escrita usando el stack de Python y Numpy. El archivo **tutorial.ipynb** cuenta con una plantilla a modo de guía para su implementación.

d) (1.0 pts.) ¿Cuál modelo (regresión logística o discriminante de Fisher) tiene mejor desempeño en el conjunto de testeo? Para responder esta pregunta proponga y justifique una métrica para evaluar el desempeño de un método general de clasificación, luego, evalúe ambos clasificadores usados y compárelos. ¿Por qué cree usted que es necesario comparar ambos modelos en el conjunto de testeo en vez de el conjunto de entrenamiento o validación?
Observaciones: para el caso de la regresión logística considere la mejor configuración del punto b) y a la hora de escoger la métrica tenga en cuenta que las clases están **desbalanceadas**, i.e., la cantidad de elementos por clase es distinta.

P3. Proyecto curso (1 pto)

Esta parte debe ser respondida por **un único** integrante del grupo, donde debe indicar los demás integrantes del grupo. Se habilitará una tarea aparte para su entrega en formato PDF cuya **extensión máxima es media página**.

a) (0.5 ptos) Enmarque su proyecto en algún tipo de aprendizaje (clasificación, regresión, clustering, aprendizaje, reforzado, etc.). Identifique según el caso el rol de sus variables, por ejemplo, si su proyecto es de clasificación, describa su variable independiente, sus características y sus clases. Si su proyecto no parece caber evidentemente en ninguno de estos casos, explíquelo en detalle y propongo cómo lo abordará desde el punto de Aprendizaje de Máquinas.

b) (0.5 ptos) Proponga al menos 2 métodos para resolver su problema, fundamentando brevemente su elección.