Tarea 1 EL4106 - Semestre Otoño 2019

Profesor: Javier Ruiz del Solar
Auxiliar: Patricio Loncomilla
Ayudantes: Gabriel Azócar, Nicolás Cruz, Francisco Leiva, Giovanni Pais

Fecha enunciado: lunes 1 de abril de 2019
Plazo entrega tarea: domingo 14 de abril de 2019

El objetivo de esta tarea es implementar un clasificador de *air showers* generados por rayos gamma primarios v/s hadrones. Se usará el set de datos MAGIC Gamma Telescope Data Set, el cual corresponde a un conjunto de simulaciones del proceso indicado. El dataset forma parte del *UC Irvine Machine Learning Repository*. El set de datos contiene 10 características, además de una clase, la cual puede ser g (gamma) o h (hadrón). Hay 8992 ejemplos de rayos gamma y 4384 ejemplos de hadrones. El set de datos contiene 11 columnas: las primeras 10 son las características, y la última es la clase. El set de datos está disponible tanto en u-cursos como en la página del repositorio.

Se les pide programar, entrenar y calibrar un clasificador bayesiano para determinar en forma probabilística si un candidato corresponde a un pulsar. La regla de decisión del mencionado clasificador está dada por:

$$\frac{P(características|hadrón)}{P(características|no\_hadrón)} \geq \Theta$$

El umbral Θ depende de los costos asociados a cada decisión ($c_{nohadrón}$ y $c_{hadrón}$) y de las probabilidades a priori:

$$\Theta = \frac{\mathrm{c_{nohadrón}P(hadrón)}}{\mathrm{c_{hadrón}P(no\_hadrón)}},$$

En este caso concreto, las distribuciones de probabilidad de ambos conjuntos, las verosimilitudes, no se tienen, por lo que deberán estimarlas de 2 maneras: calculando histogramas normalizados y calculando un modelo gaussiano a partir de un conjunto de entrenamiento de cada clase.

La tarea se debe realizar usando Python. En esta tarea se debe programar la separación de datos y los clasificadores desde cero; es decir, no se puede usar *scikit-learn*, *pandas*, *tensorflow* o herramientas similares. El uso de *numpy* está permitido.

Se pide:

1) Base de Datos:
   a) Explique la función y posibles subconjuntos de los conjuntos de entrenamiento y prueba.
   b) Dividir la base de datos en 2 conjuntos representativos: entrenamiento (80%) y prueba (20%). Compruebe la representatividad de éstos, verificando si la proporción de ambas clases se mantiene cercana a la proporción del conjunto completo. Se recomienda hacer una permutación al azar de los datos de cada clase antes de definir los conjuntos de entrenamiento y prueba.

2) Modelo con histogramas:
   a) Explique la aproximación Naive Bayes y cuál es la suposición que se asume al utilizarla.
   b) Utilice Naive Bayes y encuentre los histogramas de cada clase a partir de las muestras del conjunto de entrenamiento. Elija un número de bins que le parezca apropiado a priori (recuerde que la verosimilitud es una función de densidad de probabilidades, por lo que su integral suma 1)

c) Encontrar las verosimilitudes en ambas clases para cada muestra del conjunto de prueba, usando los histogramas.
d) Mover Θ, clasificar el conjunto de prueba y calcular la *Tasa de Verdaderos Positivos* y *Tasa de Falsos Positivos* cada vez para luego generar la curva ROC (TVP vs TFP), recuerde que Θ puede tomar valores entre 0 e ∞, pero no todo el rango entrega información importante.
e) Analice el efecto de cambiar la cantidad de bins en los histogramas. ¿Tiene la cantidad óptima de bins alguna relación con el número de muestras del conjunto de entrenamiento? Comente.

3) Modelo gaussiano:
a) Investigue y encuentre la función de densidad de probabilidad de una gaussiana multidimensional, explique las variables que la caracterizan.
b) Entrenar un modelo gaussiano multidimensional para cada clase, encontrando la media y covarianza, a partir del conjunto de entrenamiento. Puede basarse en las funciones np.mean( ) y np.cov( ) de *numpy*.
c) Encontrar las verosimilitudes en ambas clases para cada muestra del conjunto de prueba, usando las gaussianas.
d) Mover Θ, clasificar el conjunto de prueba y calcular la *Tasa de Verdaderos Positivos* y *Tasa de Falsos Positivos* cada vez para luego generar la curva ROC (TVP vs TFP).

4) Comparación:
a) Compare ambas curvas ROC y comente ventajas y desventajas de ambos métodos.
b) Implemente una función llamada *tarea1*(**entrenamiento**,**prueba**) que reciba como parámetro una matriz del conjunto de entrenamiento y otra del conjunto de prueba, que genere como salida un gráfico de las curvas ROC generadas con ambos modelos.

5) Entregar un informe donde aparezca: (i) funciones importantes del código explicadas, (ii) comentarios de cada parte en el desarrollo de la tarea, (iii) el análisis de los resultados obtenidos.

El código debe implementado en Python. Los informes y códigos deben ser subidos a u-cursos a más tardar a las 23:59 del día domingo 14 de abril. Incluir un corto archivo de texto explicando cómo se utiliza su programa. **Las Tareas atrasadas serán penalizadas con un punto base más un punto de descuento adicional por cada día extra de atraso.**

Importante: La evaluación de la tarea considerará el correcto funcionamiento del programa, la inclusión de los resultados de los pasos pedidos en el informe, la calidad de los experimentos realizados y de su análisis, la inclusión de las partes importantes del código en el informe, así como la forma, prolijidad y calidad del mismo.

Importante 2: Dado que los dos clasificadores parten de supuestos distintos, es posible que uno de ellos funcione y el otro no para este set de datos.

Importante 3: Se adjunta un documento que indica la estructura recomendada para el informe.