Aprendizaje Automático: Trabajo práctico 0

M. Sc. Saúl Calderón Ramírez
Instituto Tecnológico de Costa Rica,
Escuela de Ingeniería en Computación, Programa de Ciencias de Datos,
PAttern Recongition and MAchine Learning Group (PARMA-Group)

8 de junio de 2021

Fecha de entrega: Domingo 27 de Junio.
Entrega: Un archivo .zip con el código fuente LaTeX o Lyx, el pdf, y un notebook Jupyter, debidamente documentado. A través del TEC-digital.
Modo de trabajo: Grupos de 3 personas.

Resumen

En el presente trabajo práctico se introducir los arboles de decision por medio de su implementación para resolver un problema práctico.

1. Implementación de la clasificación multi-clase con árboles de decisión (60 puntos)

1. El conjunto de datos disponible en `https://www.kaggle.com/yashsawarn/wifi-stretgth-for-rooms` corresponde a lecturas de 7 fuentes de señal Wi-Fi, los cuales pretender ser utilizados para determinar si el receptor de las lecturas se encuentra en la habitación 1, 2, 3 o 4. La Figura 1 muestra una muestra del conjunto de datos, donde se observa que usualmente las lecturas de la señal Wi-Fi son valores negativos.

2. Para resolver el problema de discriminar en cual habitación se encuentra el receptor de la señal Wi-Fi, su equipo decidió construir un arbol de decisión (CART por sus siglas en inglés). Para ello utilizará el código provisto en el *notebook* de *Jupyter*.

    a) El código provisto define las clases *CART* y Node_*CART*, las cuales permiten construir un CART binario. Cada nodo del arbol tiene atributos como el *feature,* el umbral, y el coeficiente de *gini* (o la entropía) de la partición definida en tal nodo. Además define el atributo *dominant_class* para el nodo, el cual es el resultado de calcular la clase con mayor cantidad de apariciones en la partición que define al nodo. Finalmente el código incluye la funcionalidad para generar

	A	B	C	D	E	F	G	H	I
1	ID_1	-64	-56	-61	-66	-71	-82	-81	1
2	ID_2	-68	-57	-61	-65	-71	-85	-85	1
3	ID_3	-63	-60	-60	-67	-76	-85	-84	1
4	ID_4	-61	-60	-68	-62	-77	-90	-80	1
5	ID_5	-63	-65	-60	-63	-77	-81	-87	1
6	ID_6	-64	-55	-63	-66	-76	-88	-83	1
7	ID_7	-65	-61	-65	-67	-69	-87	-84	1
8	ID_8	-61	-63	-58	-66	-74	-87	-82	1
9	ID_9	-65	-60	-59	-63	-76	-86	-82	1

Figura 1: Muestra del conjunto de datos a utilizar.

un archivo *xml* (el cual se puede abrir en cualquier navegador web), para representar fácilmente el árbol.

b) **(10 puntos)** Implemente el método *calculate_gini(data_partition_torch, num_classes = 4)*, el cual calcule el coeficiente de gini para el conjunto de datos recibido en un tensor de *pytorch*. Para ello utilice la definición inidicada en el material del curso:

$$E_{\text{gini},\rho}\left(\tau_d\right) = 1 - \sum_{k=1}^{K} a_k^2.$$

c) **(10 puntos)** Implemente el método *calculate_entropy(data_partition_torch, num_classes = 4)*, el cual calcule la entropía de las etiquetas, representadas en un tensor de *pytorch*. Para ello utilice la definición inidicada en el material del curso:

$$E_{\text{entropy},\rho}\left(\tau_d\right) = -\sum_{k=1}^{K} p\left[k\right] \log\left(p\left[k\right]\right),$$

donde $p\left[k\right]$ es la función de densidad de las etiquetas en la partición de datos recibida por parámetro.

d) **(30 puntos)** Implemente el método *select_best_feature_and_thresh(data_torch, list_features_selected = [], num_classes = 4)*, de la clase *Node_CART*. Este método recibe como parámetros el conjunto de datos en un tensor tipo torch a analizar, además de la lista de *features* seleccionados hasta ahora (para ignorarlos en futuras pruebas), y la cantidad de clases a discriminar. El método debe probar de forma extensiva todos los posibles features y sus correspondientes umbrales en los datos recibidos, hasta dar con el menor coeficiente ponderado de gini (o la

minima entropía, dependiendo de la función de error a utilizar) (ignorando los features en *list_features_selected = []*). **Utilice indexación lógica para evitar al máximo el uso de estructuras de repetición tipo *for***. Solamente puede usar estructuras de repetición para iterar por los *features* y posibles umbrales dentro del conjunto de datos. Recuerde que para evaluar una posible partición, es necesario calcular el coeficiente de gini ponderado sugerido para decidir el feature y umbral óptimos es:

$$\overline{E}_{\text{gini}}(\tau_d, d) = \frac{n_i}{n} E_{\text{gini}}(D_i) + \frac{n_d}{n} E_{\text{gini}}(D_d).$$

Realice un ponderado similar para la entropía.

e) **(10 puntos)** Implemente la función *test_CART* la cual evalúe un CART previamente entrenado para un conjunto de datos D representado en un tensor. Calcule la tasa de aciertos (*accuracy*), definida como:

$$a = \frac{c}{n}$$

donde c corresponde a las estimaciones correctas, para tal conjunto de datos y retornela.

2. Evaluación del CART (40 puntos)

1. **(20 puntos)** Evalúe el CART implementado con el conjunto de datos completo provisto `https://www.kaggle.com/yashsawarn/wifi-stretgth-for-rooms` usandolo como conjunto de datos de entrenamiento y prueba. Reporte la tasa de aciertos obtenida e incluya el código de la evaluación. Pruebe con una profundidad máxima de 2 y 3 nodos, siempre con minimo 2 observaciones por hoja.

   a) Realice lo anterior usando la entropía y el coeficiente de Gini. Compare los resultados.

2. **(20 puntos)** Para una profundidad máxima de 2 y 3 nodos: evalúe el CART implementado usando 10 particiones aleatorias del conjunto de datos, con un 70 % del conjunto de datos como conjunto de datos de entrenamiento, y el restante 30 % como conjunto de datos de prueba. Reporte una tabla con la tasa de aciertos de cada una de las 10 corridas, y el promedio y desviación estándar para las 10 corridas.

   a) Realice lo anterior usando la entropía y el coeficiente de Gini. Compare los resultados y comente las posibles ventajas y desventajas de cada función de error. En la comparativa, incluya el tiempo de ejecución en entrenar cada modelo con cada función de error distinta.