Tercera práctica de Inteligencia Artificial
Curso 2021-2022

Departament d'Informàtica i Enginyeria Industrial
Universitat de Lleida
carlos@diei.udl.cat
josep.alos@udl.cat
eduard.torres@udl.cat

1. Enunciado

El objetivo de esta práctica es evaluar el conocimiento del alumno sobre algoritmos de aprendizaje supervisado y no supervisado. Estos algoritmos están descritos en las diapositivas de la asignatura, junto a las tareas que componen esta práctica.

La práctica debe desarrollarse en base al material adjunto proporcionado junto al enunciado de la práctica.

1.1. Árboles de decisión - Implementación.

1.1.1. T10 - Construcción del árbol de forma iterativa. 1.5 puntos

Implementad la función `iterative_buildtree(part, scoref, beta)` que depende de una medida de impureza `scoref` y el criterio `beta` para limitar la construcción. El árbol resultante debe ser igual que el de la construcción recursiva (función `buildtree(...)`) que hemos hecho en clase.

1.1.2. T12 - Función de clasificación. 0.5 puntos

Una vez tenemos un árbol construido podemos clasificar datos nuevos usando el árbol. Implementad la función `classify(tree, row)` que devuelve la clase predicha para la `row`. El criterio de asignación de la clase en hojas que tengan múltiples etiquetas debe explicarse en el informe.

NOTA En clase hemos desarrollado el código asumiendo que el último valor de cada observación es la etiqueta. En datos futuros, no vamos a tener este valor. Tenéis que aseguraros que esta función se comporta correctamente cuando el parámetro `row` no contiene este último valor "etiqueta".

1.1.3. T16 - Poda del árbol. 1 punto

Implementad el método `prune(tree: DecisionNode, threshold: float)`. Este método debe ejecutar el algoritmo de poda explicado en clase:

- *Construir un árbol completo* - En nuestro caso lo recibimos ya construido por parámetro.
- Para cada par de hojas con el mismo padre, comprobar si unir las hojas incrementa la impureza por debajo de `threshold`. En caso afirmativo, unir las hojas y convertir el padre en hoja.
- Repetir para cada par de hojas hasta no poder podar más el árbol.

1.2. Árboles de decisión - Evaluación.

1.2.1. T13 - Test performance. 0.25 puntos

Definid la función `get_accuracy(tree: DecisionNode, dataset)` que, dado un árbol y un conjunto de datos (ya sea de entrenamiento o de test), nos devuelva el ratio de ejemplos correctamente clasificados en este dataset.

1.2.2. Cross-validation. 1 punto

Implementad el método `cross_validation(dataset, k, agg, seed, scoref, beta, threshold)`. Este método se va a encargar de ejecutar el proceso de *cross-validation* que se explica a continuación. Los parámetros que recibe són:

- dataset: El conjunto de datos que va a ser usado
- k: El número de particiones (*folds*) que se van a generar
- agg: La función de agregación de los *accuracies*
- seed: La semilla a usar para particionar los datos
- scoref, beta, threshold: Parámetros que se van a usar para construir el árbol.

Como función de agregación, podeis usar la media:

```
def mean(values: List[float]):
    return sum(values) / len(values)
```

1.2.3. Encontrar el mejor parámetro de threshold. 0.25 puntos

Como sabemos, modificar los parámetros va a resultar en árboles con mayor o menor capacidad de generalización. Explora por lo menos 4 valores distintos para el parámetro de *threshold*. Escoged el mejor basándote en la puntuación obtenida mediante *cross-validation*.

Para ello, separad primero el conjunto de datos proporcionado `iris.csv` en entrenamiento y test. Obtened la mejor configuración usando *cross-validation* sobre el conjunto de entrenamiento. Una vez tengáis los mejores parámetros, entrenad un nuevo árbol sobre el conjunto total de datos de entrenamiento, y reportad la *accuracy* final de vuestro árbol en el conjunto de test.

Comentad los resultados en el informe.

1.3. Clustering - Implementación.

Para todos los ejercicios, considerad la función de distancia euclídea al cuadrado.

1.3.1. T9 - Total distance. 0.5 punto

Modificad la función `kcluster` para que devuelva una tupla compuesta de (1) los centroides encontrados, y (2) la suma de las distáncias de todos los puntos a su centroide.

1.3.2. T11 - Restarting policies. 2 puntos

Queremos que k-means asigne los k clústers tal que se minimicen la suma de las distancias de todos los ítems a sus respectivos centroides. Dado que el resultado de k-means depende de la inicialización de los k centroides, deberíamos ejecutarlo varias veces para conseguir la mejor asignación de clústers, aquella que minimiza esa suma de distancias.

Mejorad la función `kcluster` para que tenga en cuenta ese número de veces de ejecución (parametrizable) y se quede el resultado de la mejor configuración.

1.4. Clustering - Evaluación.

1.4.1. T10 - Distáncia en función de k. 0.5 puntos

Mostrad la distáncia total en función de diferentes valores de k.

1.4.2. Escoger un valor de k. 0.5 puntos

Hasta el momento, hemos usado una k fija para nuestros experimentos. Con un conjunto de datos real, es difícil estimar que valor escoger para nuestros datos.

Investigad el "método del codo" (*elbow method*) para escoger un valor k que sea suficientemente bueno sin llegar a sobreentrenar el modelo. Determinad este valor de k para el conjunto de datos `blogdata_full.txt`. Añadid el gráfico a partir de los resultados del ejercicio anterior en la documentación a entregar, y analiza el resultado.

2. Implementación. 1 punto

Se valora la calidad y eficiencia de los algoritmos implementados. Es necesario tener en cuenta los siguientes aspectos de la implementación:

- El lenguaje de programación es Python.
- La utilización de un diseño descendente y orientado a objetos.
- La simplicidad y legibilidad del código.
- Se recomienda seguir la guía de estilo [1].

3. Documentación. 1 punto

Documento en pdf (máximo $\approx$ 4 páginas) que incluya una descripción de las decisiones que se han tomado en la implementación, así como los posibles resultados experimentales y comentarios teóricos. Para ser evaluado, el documento debe contener **como mínimo** los puntos 1.1.2, 1.2.3, y 1.4.2.

La página inicial ha de contener el nombre de los integrantes del grupo. La práctica puede realizarse en grupos de dos personas o individualmente.

Se valorará la redacción y presentación del documento

[1] `https://www.python.org/dev/peps/pep-0008/`

4. Material a entregar

El material a entregar es:

- Todos los archivos de código fuente, modificados o añadidos.
- Documento de la práctica.

Todo el material requerido se entregará en el paquete de nombre `ia-prac3.[tgz|tar.gz|zip]`