Segunda práctica de Inteligencia Artificial
Curso 2020-2021

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.

1.1. Árboles de decisión. 3.5 puntos

Podéis utilizar el dataset `decision_tree_example.txt` para probar vuestras implementaciones (lo podéis encontrar dentro del campus virtual, en Recursos → Laboratorio → learning).

Para poder visualizar el árbol utilizad el código del fichero `printtree.py`, que se encuentra en la misma carpeta del campus virtual.

1.1.1. T9 - Construcción del árbol de forma recursiva. 1 punto

Implementación de la función `buildtree` que depende de una medida de impureza `scoref=(entropy|gini_impurity)` y el criterio β para limitar la construcción.

1.1.2. T10 - Construcción del árbol de forma iterativa. 1.75 puntos

Implementación de la función `buildtree_ite` que depende de una medida de impureza y el criterio *beta* para limitar la construcción. El árbol resultante debe ser igual que el de la construcción recursiva.

1.1.3. T12 - Función de clasificación. 0.75 puntos

Una vez tenemos un árbol construido podemos evaluar nuevos datos contra el árbol. Implementar la función `classify` que devuelve la partición del nodo hoja donde acaba cayendo un nuevo dato.

1.2. Clustering. 4 puntos

Para todos los ejercicios, considerad la función de distancia euclídea al cuadrado. Utilizad el conjunto de datos `seeds.csv` que tenéis subido en la carpeta Recursos → Laboratorio → learning → Clustering del campus.

1.2.1. T9 - Total distance. 1 punto

Modifica la función `fit` de la clase KMeans para que guarde la suma de las distancias de todos los puntos a su centroide, y lo guarde en un atributo de la clase llamado `inertia_`.

1.2.2. T10 - Distancia en función de *k*. 1 punto

Muestra, mediante una tabla o un gráfico, la distancia total del algoritmo para diferentes valores de k.

1.2.3. T11 - Restarting policies. 1.5 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 de entrenamiento de nuestra clase para que tenga en cuenta ese número de veces de ejecución (parametrizable) y se quede como resultado la mejor configuración.

1.2.4. Escoger un valor de *k*. 0.5 puntos

Hasta el momento, hemos usado una k fija para nuestros experimentos. En la práctica, es difícil estimar que valor usar para nuestros datos.

Investiga algún método para escoger un número óptimo de clústers (ya sea a priori de la ejecución o después de múltiples ejecuciones del algoritmo). Calcula este valor óptimo para el conjunto de datos de la práctica.

1.3. Scikit-learn. 0.5 puntos

1.3.1. Missing data. 0.5 puntos

Hasta ahora, todos los conjuntos de datos que tenemos han sido previamente limpiados y procesados para centrarnos exclusivamente en el algoritmo. En la vida real esto no es así, y habitualmente tendremos conjuntos de datos con valores que faltan.

En la carpeta Recursos → Laboratorio → learning → sklearn tenéis los conjuntos de datos de entrenamiento y de testeo (missing_data_train.csv y missing_data_test.csv respectivamente).

Investiga sobre los métodos que proporciona Scikit-learn para imputar valores inexistentes en nuestro conjunto, y crea un archivo que contenga todo el siguiente procedimiento:

- Cargar los dos conjuntos de datos (train y test) por separado, usando Pandas
- Preprocesar los datos de entrenamiento para sustituir los datos que faltan en el conjunto
- Entrenar una instancia de nuestra clase KMeans con los datos de entrenamiento (importad directamente el archivo *clusters.py*, no hagáis un copia-pega de todo el código.
- Predecir los clústers del conjunto de datos de test, e imprimirlos.

Nota Para comprobar si un conjunto de datos de Pandas tiene un valor nulo puedes usar (asumiendo un DataFrame llamado df) `df.isnull().values.any()`

Nota 2 Aunque a veces tenga sentido eliminar los registros con valores nulos, en este caso no podéis optar por esta opción. Tenéis que usar todos los registros.

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 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.

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

4. Material a entregar

El material evaluable de esta práctica 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-prac2.[tgz|tar.gz|zip]`

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