PAC 1

Presentación

Nuestra empresa ha creado una aplicación móvil con la que los usuarios pueden solicitar sugerencias de restaurantes. Una vez han visitado el restaurante, los usuarios hacen una valoración de su precio y calidad que se añade a una base de datos que ayuda al sistema a generar nuevas sugerencias por los usuarios, comparando usuarios con preferencias similares.

En esta PAC 1 repasaremos estos conceptos básicos de recomendación siguiendo el hilo de este ejemplo.

Competencias

En este enunciado se trabajan en un determinado grado las siguientes competencias general de máster:

- Capacidad para proyectar, calcular y diseñar productos, procesos e instalaciones en todos los ámbitos de la ingeniería en informática.
- Capacidad para el modelado matemático, cálculo y simulación en centros tecnológicos y de ingeniería de empresa, particularmente en tareas de investigación, desarrollo e innovación en todos los ámbitos relacionados con la ingeniería en informática.
- Capacidad para la aplicación de los conocimientos adquiridos y para solucionar problemas en entornos nuevos o poco conocidos dentro de contextos más amplios y multidisciplinares, siendo capaces de integrar estos conocimientos.
- Poseer habilidades para el aprendizaje continuado, autodirigido y autónomo.
- Capacidad para modelar, diseñar, definir la arquitectura, implantar, gestionar, operar, administrar y mantener aplicaciones, redes, sistemas, servicios y contenidos informáticos.
- Capacidad para asegurar, gestionar, auditar y certificar la calidad de los desarrollos, procesos, sistemas, servicios, aplicaciones y productos informáticos.

Las competencias específicas de esta asignatura que es trabajan son:

- Entender que es el aprendizaje automático en el contexto de la Inteligencia Artificial.
- Distinguir entre los diferentes tipos y métodos de aprendizaje.
- Aplicar las técnicas estudiadas en un caso concreto.

Objetivos

En este PAC se practicarán los conceptos del temario relacionados con recomendación y clustering mediante su aplicación a un caso concreto.

Descripción de la PAC a realizar

Datos

En la base de datos de nuestros servidores hay un total de 50 restaurantes (identificados con un número 0 .. 49). Por otro lado, hay 100 usuarios de la aplicación (identificados con un número 0 .. 99).

Se dispone de tres archivos de datos. El primero, *valoraciones.data*, contiene el histórico de valoraciones que el sistema ha ido registrando. Cada fila es una valoración de un restaurante por un usuario, con las siguientes columnas: idUsuari, idRestaurant, precio, calidad. En general los usuarios no han visitado todos los restaurantes, sólo unos cuantos.

El precio está expresado en euros por persona, mientras la calidad está expresada en una escala desde el 0 (peor) hasta el 10 (mejor).

El segundo fichero, *descubrimientos.data*, recoge los descubrimientos realizados por los usuarios, donde cada usuario indica cuál es el restaurante que más le gustó de entre los que no había valorado previamente. En este archivo hay una línea por usuario, con las columnas idUsuari y idRestaurant. Este par (idUsuari, idRestaurant) no sale en el archivo *valoraciones.data*.

Finalmente, el archivo *restaurantes.data* hay un listado de los restaurantes con su posición geográfica. Hay una línea para restaurante, con las columnas idRestaurant, coordX, coordY.

Actividades

Para resolver esta PAC debe utilizar el código de los programas 2.2-2.7 con las modificaciones apropiadas para trabajar con los nuevos ficheros de datos. También utilizará el código 4.4 (k-means).

En todas las actividades hay que justificar razonadamente las respuestas.

Actividad 1

El paso previo a cualquier análisis de datos es efectuar un tratamiento para adecuar sus valores. Entre las operaciones habituales encontramos la normalización o estandarización de los datos (ver por ejemplo http://es.wikipedia.org/wiki/Distribución_normal).

Ante todo se pide, pues, que realice el tratamiento previo de los datos que considere necesario (en algún caso podría no ser necesario hacer ningún tratamiento previo).

Como en las actividades que se plantean no hay que combinar datos con diferentes escalas o unidades, no es necesario llevar a cabo ningún tipo de normalización o estandarización de los datos.

Pero es importante comprobar que los datos se encuentran dentro de los rangos esperados. En este ejercicio encontramos que, en el fichero *valoraciones.data*, hay algunos precios (tercera columna) negativos. Eso es claramente un error, porque de ninguna manera se puede aceptar que el precio de comer en un restaurante sea negativo.

Como no hay manera de editar los precios negativos que no implique inventarse datos, lo que podría distorsionar los resultados, lo más adecuado en este caso es eliminar las entradas con precios negativos. Se pierden algunos datos pero a cambio no se introducen valores extraños.

En el fichero *apartat1.py* hay cuatro funciones que leen los ficheros de datos y los transforman en diccionarios adecuados para su procesamiento posterior. Las funciones son:

- **leeValoraciones**: lee el fichero *valoraciones.data* y devuelve un diccionario con las valoraciones de cada usuario para cada restaurante que ha valorado: **{usuario : {restaurante : valoración}}**
- **leePrecios**: lee el fichero *valoraciones.data* y devuelve un diccionario con los precios medios de cada restaurante, descartando las entradas con precios negativos: **{restaurante : precio}**
- **leeDescubrimientos**: lee el fichero *descubrimientos.data* y devuelve un diccionario con el nuevo restaurante preferido por cada usuario, con la siguiente estructura: **{usuario : restaurante}**
- **leePosiciones**: lee el fichero *restaurantes.data* y devuelve un diccionario con la posición geográfica de cada restaurante: **{restaurante: (posiciónX, posiciónY)}**

Estas funciones se utilizarán en los apartados siguientes.

Actividad 2

Con los datos del fichero *valoraciones.data* genere un recomendador ponderado de restaurantes que vuelva, por un usuario determinado, una lista de recomendación (ordenada por afinidad estimada) de los restaurantes que el usuario aún no ha visitado. Pruebe con ` euclidiana y con correlación de Pearson como medidas de similitud, teniendo en cuenta sólo la valoración de calidad, no el precio.

Para evaluar la calidad de la recomendación, se encuentra en qué posición se encuentra el restaurante que el usuario ha indicado como descubierta preferida. el archivo *descubrimientos.data*. Se supone que cuanto más baja

sea su posición en la lista, mejor. Calcula la posición media de los descubrimientos en la lista de recomendación para todos los usuarios.

Ejemplo: si un usuario ha dicho que su descubrimiento ha sido el restaurante 15 y el recomendador genera la lista [3 22 15 4 41 32], la posición de su descubrimiento será la 2 (suponiendo que la primera posición es la cero) . Si la posición fuera cero, esto indicaría una recomendación perfecta.

Compare la calidad de la recomendación con similitud euclidiana y Pearson.

Para resolver este apartado se utilizará el recomendador ponderado que se encuentra en los materiales de la asignatura, con similitud euclídea y correlación de Pearson.

Como el recomendador devuelve una lista con las recomendaciones con una ponderación asociada pero en este apartado no interesa la ponderación, esta se elimina del resultado.

También se leen los descubrimientos de los usuarios para averiguar en qué posición de los restaurantes recomendados se encuentran. Los resultados son los siguientes:

Similitud	**Posición mínima**	**Posición máxima**	**Posición media**
Euclídea	0	19	5,07
Pearson	0	4	1,59

Se puede ver cómo Pearson da mejores resultados, acertando más las preferencias de los usuarios. Esto es lógico porque evita el sesgo asociado a cada usuario: unos usuarios tienden a puntuar mejor en general, i otros peor. La correlación de Pearson es capaz de captar las coincidencias aunque los usuarios usen niveles medios de valoración diferentes. La similitud euclídea sólo tiene en cuenta los valores absolutos.

En el fichero *apartado2.py* se encuentra el código para ejecutar y evaluar los recomendadores.

Actividad 3

Haga un agrupamiento jerárquico aglomerados de los restaurantes según su posición geográfica (archivo *restaurantes.data*). Tome como resultado el agrupamiento en el que hay exactamente cuatro grupos, que son grupos de restaurante por zonas.

A continuación, calcula el precio medio y la calidad media de los restaurantes de cada zona. Piensa que hay relación entre la zona a la que pertenece el restaurante y su precio? Y con la calidad?

Pruebe con los métodos de enlace simple y completo. Comparar los resultados.

Como las valoraciones de los restaurantes están organizadas por usuarios, en primer lugar conviene reorganizarlas y calcular la valoración media de cada restaurante como **{restaurante : valoración}**. Con esta estructura de datos se puede usar la misma función (*valorMedioGrupo*) para calcular los precios y las valoraciones medias de cada grupo, pasando como parámetro el diccionario de precios o el de valoraciones.

En la tabla siguiente se resumen los resultados obtenidos con el criterio **simple** de enlace:

Grupo	**1**	**2**	**3**	**4**
Número de elementos	10	24	1	15
Precio medio	38,93	23,21	40,12	14,034
Valoración media	7,013	7,16	6,49	7,10

Puede comprobarse que hay diferencias claras en los precios medios entre zonas, pero en las valoraciones la diferencia no es tan clara, teniendo en cuenta que la única valoración claramente inferior (grupo 3) corresponde a un grupo “degenerado”, con un único elemento, lo que hace que sus valores medios sean poco significativos.

Respecto al criterio de enlace completo, los resultados son:

Grupo	**1**	**2**	**3**	**4**
Número de elementos	11	23	8	8
Precio medio	39,04	23,54	13,41	14,86
Valoración media	6,96	7,14	7,015	7,25

En primer lugar, hay que recordar que los números de grupo no tienen ninguna relación entre un agrupamiento y otro, no hay que intentar identificarlos directamente, sino teniendo en cuenta sus características.

La diferencia más evidente entre los resultados de los dos criterios de enlace es el número de elementos de cada grupo; con enlace completo los grupos están más equilibrados, no hay grupos de un elemento como en el caso anterior, lo que en general es bueno (si bien puede haber valores “outliers” en los datos, que sí deberían estar separados).

Aparentemente, el grupo 4 del enlace simple corresponde a los grupos 3 y 4 del completo, y los grupos 1 y 2 tienen valores parecidos y probablemente corresponden a los mismos restaurantes. Se puede comprobar consultando los restaurantes de cada grupo.

Respecto a los precios, hay un grupo con precios altos, otro con precios intermedios y dos grupos con precios bajos. Las diferencias son bastante

claras y se puede asegurar que hay relación entre el precio y la zona geográfica.

Aunque hay diferencias de valoración, no son tan claras y no se puede asegurar que estén fuertemente relacionadas con la localización.

Los resultados obtenidos en este apartado indican que hay una relación clara entre la situación geográfica y el precio de los restaurantes. La relación entre valoración y situación no es clara porque, al fin y al cabo, las valoraciones de los usuarios son subjetivas y cada usuario puede valorar de manera diferente cada restaurante. Aunque puede haber una relación entre calidad y situación geográfica, con los datos disponibles no hay un valor absoluto de calidad y por tanto es imposible encontrar esa relación.

Finalmente, parece más adecuado usar el enlace completo para obtener grupos mejor formados y posiblemente más útiles para analizar las diferencias entre restaurantes.

En el fichero *apartado3.py* tenéis el código correspondiente a la solución.

Sugerencia: se puede valorar si las diferencias entre grupos son significativas con un test estadístico como el que se describe en http://www.stat.yale.edu/Courses/1997-98/101/meancomp.htm.

Actividad 4

Medir la similitud entre cada par de usuarios con la correlación de Pearson aplicada a sus valoraciones de calidad del fichero *valoraciones.data*. Con estas similitudes, aplique el algoritmo k-means (lo tenéis al código 4.4 de los materiales) para generar k grupos de usuarios, supuestamente con preferencias similares. Para simplificar esta actividad, no tendremos en cuenta el precio de los restaurantes, sólo la valoración de su calidad.

De cada uno de los k grupos obtiene un centroide, es decir un usuario representativo de cada grupo, las preferencias del cual se supone que coinciden bastante con las del grupo. Con esta información generaremos un recomendador basado en modelos.

Concretamente, por cada centroide se calculará la lista de restaurantes ordenada por valoración (primero los mejor valorados y al final los peores).

Entonces, por cada usuario debería identificar a qué grupo pertenece y tomar como sugerencia el restaurante preferido por centroide del grupo.

Para poder evaluar el rendimiento de este recomendador, tomaremos los restaurantes elegidos por los usuarios en el archivo *descubrimientos.data* y miraremos a qué posición de la lista recomendada por su grupo se encuentra. Cuanto más abajo (posición más baja) en la lista, más acertada se supone

que es la recomendación. La posición media de los descubrimientos reflejará la calidad media de la recomendación.

Nota: como los usuarios no han visitado todos los restaurantes, los centroides tampoco lo han hecho. Por este motivo los descubrimientos de algunos usuarios no aparecerán en las listas de preferencia de los centroides. Puede descartar estos casos (no contar a la hora de hacer los cálculos).

Pruebe este recomendador con k = 2, 3, ..., 6. Con qué valor se obtiene mejores resultados? ¿Por qué pensáis que puede ser eso?

Dado que el código de k-means que hay en los materiales de la asignatura utiliza la distancia euclídea y espera listas con las valoraciones (no permite diccionarios), es necesario cambiarlo para que que se pueda resolver este apartado (fichero *kmeans_dictio.py*).

El código de *kmeans_dictio* devuelve la asignación de cada usuario a un grupo determinado y las valoraciones medias del grupo para cada restaurante (centroides). Tal y como sugiere el enunciado, se puede tomar un usuario de cada grupo como representante (el usuario más cercano al centroide que devuelve k-means). Los restaurantes mejor valorados por ese usuario serán los que se sugieran a los usuarios del grupo.

Por tanto, para cada grupo 1..k se han de obtener los restaurantes mejor valorados por el usuario representante y compararlos con los descubrimientos de los usuarios que pertenecen a ese grupo. Como se hizo en el apartado 2, la posición media de los nuevos restaurantes mejor valorados (fichero *descubrimientos.data*) dará idea de la calidad de las recomendaciones.

Los resultados de una ejecución con diferentes valores de k son los siguientes:

K	Posición mínima	Posición máxima	Posición media
2	0	28	9,16
3	0	26	7,71
4	0	29	8,37
5	0	29	8,71
6	0	29	8,14
7	0	29	8,0
8	0	29	9,28
9	0	26	5,87
10	0	29	6,11

En general los valores son peores que los obtenidos con el recomendador ponderado, con cualquiera de las dos medidas de similitud. ¿Por qué puede ocurrir eso? El motivo principal es que el recomendador ponderado utiliza

todas las valoraciones de todos los usuarios para hacer una recomendación, mientras que el basado en modelos sólo guarda una lista de recomendaciones para cada uno de los k grupos. La inforrmación que se utiliza es mucho más reducida, y a eso se añade el error introducido al clasificar un usuario en un grupo u otro.

Las ventajas de este recomendador son que ocupa mucha menos memoria, es más rápido y permite extraer un cierto conocimiento de los datos porque indica qué usuarios son más parecidos y qué grupos puede haber.

Finalmente hay que recordar que k-means elige k elementos al azar para comenzar, por lo que los resultados pueden variar de una ejecución a otra. De hecho, la variación en los resultados con diferentes valores de k se puede explicar por la influencia del azar más que por el efecto de k. Eso se puede comprobar ejecutando el programa muchas veces. En todo caso habría que ejecutarlo repetidamente y analizar si los valores medios muestran alguna tendencia.

En el fichero *apartado4.py* está el código que resuelve este apartado.

Nota: también se podrían tomar los valores de los centroides directamente como orientación para el recomendador, porque cada centroide en este ejemplo es un diccionario de valoraciones medias de restaurantes.

Recursos

Este PAC requiere de los siguientes recursos:

Básicos: Ficheros de datos adjuntos al enunciado

Complementarios: Manual de teoría de la asignatura. En especial las tablas de código 2.2-2.7 y 4.4.

Criterios de valoración

Los ejercicios tendrán la siguiente valoración asociada:

Actividad 1: 1 punto

Actividad 2: 3 puntos

Actividad 3: 2.5 puntos

Actividad 4: 3.5 puntos

Es necesario razonar las respuestas en todos los ejercicios. Las respuestas sin justificación no recibirán puntuación.

Formato y fecha de entrega

La PAC debe entregarse antes del **próximo 4 de Abril** (antes de las 24h).

La solución a entregar consiste en un informe en formato PDF usando la plantilla colgada en el tablón de la asignatura más los archivos de código (*. Py) que usó para resolver la prueba. Estos archivos deben comprimir en un archivo ZIP.

Adjuntar el fichero a un mensaje en el apartado de **Entrega y Registro de AC (RAC)**. El nombre del archivo debe ser ApellidosNombre_IA_PAC1 con la extensión. zip.

Para dudas y aclaraciones sobre el enunciado, diríjase al consultor responsable de su aula.

Nota: **Propiedad intelectual**
A menudo es inevitable, al producir una obra multimedia, hacer uso de recursos creados por terceras personas. Es por tanto comprensible hacerlo en el marco de una práctica de los estudios del Grado en Informática, siempre y esto se documente claramente y no suponga plagio en la práctica.
Por lo tanto, al presentar una práctica que haga uso de recursos ajenos, se presentará junto con ella un documento en el que se detallen todos ellos, especificando el nombre de cada recurso, su autor, el lugar donde se obtuvo y el su estatus legal: si la obra está protegida por copyright o se acoge a alguna otra licencia de uso (Creative Commons, licencia GNU, GPL ...). El estudiante deberá asegurarse de que la licencia que sea no impide específicamente su uso en el marco de la práctica. En caso de no encontrar la información correspondiente deberá asumir que la obra está protegida por copyright.
Deberán, además, adjuntar los archivos originales cuando las obras utilizadas sean digitales, y su código fuente si corresponde.