DIPLOMATURA EN MACHINE LEARNING CON PYTHON

Reglas de Asociación

Reglas de Asociación

- Instalación y uso del paquete
- Ejemplo en Python
- Problema concreto
- Principales parámetros de ajuste y control

Introducción:

Utilizamos reglas de asociación cuando queremos descubrir las relaciones que hay entre múltiples instancias de un conjunto de atributos.

El problema inspirador de la búsqueda de reglas de asociación ha sido el análisis del carrito de compras del supermercado.

Se trata de investigar si la compra de un grupo de artículos está correlacionada con la de otro conjunto.

Naturalmente no hay pérdida de generalidad si consideramos que el segundo conjunto tiene un único elemento. Si conseguimos resolver este caso, podemos aplicarlo tantas veces como sea preciso para terminar componiendo una solución para cada uno de los miembros del segundo conjunto.

Estamos entonces buscando cosas de la pinta "si compra pañales entonces compra cerveza" lo que constituiría un ejemplo de uso de las reglas de asociación para un aprendizaje no supervisado.

Si, por el contrario, nos interesa un resultado en particular entonces podemos limitarnos a las reglas cuyo consecuente es ese y, entonces utilizarlas para tareas predictivas. Sería un uso de las reglas para realizar un aprendizaje supervisado.

Formalmente hablando tenemos entonces vectores de N componentes. Cada componente puede ser 0 o 1. Un 1 significa que el atributo en cuestión se encuentra presente y un 0 significa que no se encuentra.

La cantidad de componentes es, por supuesto, la cantidad de atributos que vamos a considerar. Las reglas que buscamos serán entonces de la forma:

$$(1,0,1,0,0) \rightarrow \beta$$

Por ejemplo si el atributo 1 es pañales y el atributo 3 es champú infantil β representa la cerveza.

La regla entonces se lee: pañales y champú infantil implican cerveza.

Al vector de atributos lo llamamos el antecedente, en nuestro ejemplo, pañales y champú infantil.

Al segundo término lo llamamos consecuente, en nuestro caso, la cerveza.

Vamos a definir para la regla dos medidas:

- Cobertura: es la proporción de casos que se dan donde aparece el antecedente. En nuestro ejemplo es la cantidad de carritos de compra donde aparecen pañales y champú infantil dividida por el número total de casos.
- Confianza: es la cantidad de casos en que se verifica la regla dividida por la cobertura. En nuestro ejemplo es la cantidad de carritos en la que aparecen pañales, champú infantil y cerveza dividida por la cantidad en que aparecen pañales y champú infantil.

Formalmente hablando tendremos entonces un conjunto de M vectores de N posiciones cada uno de la pinta:

$$V_{i,j} = 0 \text{ ó } 1 \; con \; 1 \leq i \leq N \; y \; 1 \leq j \leq M$$

Nos interesa ahora encontrar algoritmos para calcular las reglas. El primero que vamos a considerar se llama *apriori.*

Apriori necesita una cobertura mínima k. Esto significa que vamos a producir sólo reglas que representen k casos.

Empezamos por todos los conjuntos de atributos que tengan un elemento y nos quedamos con todos los que tengan la cobertura mínima. Para cada uno de ellos calculamos la confianza.

Luego, para todos los conjuntos de un atributo que cumplan la condición de cobertura consideramos todos los posibles segundos atributos que cumplan la condición de cobertura y, para cada uno, calculamos la confianza.

Así vamos agregando atributos hasta que no es posible hacerlo más porque nunca se cumple la condición de cobertura.

Hemos obtenido un gran conjunto de reglas, cada una con su cobertura y su confianza. Podemos ahora ordenarlas por la confianza en forma descendente y tomar así las que tengan un nivel de confianza que nos interese.

Este algoritmo parece terriblemente costoso. Sin embargo, en muchas situaciones de interés, la distribución de atributos es tal que se ejecuta razonablemente rápido.

Ejercicio 12.1:

Tomar los datos contenidos en la tabla casos1 en el campus virtual y encontrar todas las reglas de asociación que tengan más de 10 casos y una confianza superior al 80%

Reglas de Dependencia:

Un problema con las reglas de asociación es que no da cuenta de las coincidencias casuales de los atributos. Esto se vuelve relevante para los atributos de muy alta cobertura.

Vamos entonces a tratar de concentrarnos en las reglas que rescaten las correlaciones que se dan por encima de las coincidencias casuales.

Consideremos un par de atributos i y l para el conjunto sobre el que trabajamos las reglas de asociación:

$$V_{i,j} = 0 \text{ ó } 1 \; con \; 1 \leq i \leq N \; y \; 1 \leq j \leq M$$

Vamos a definir:

$$E(i,l) = \sum_{j=1}^{M} V_{i,j} V_{l,j}$$

E(i,l) es la cantidad de veces que el atributo i y el atributo l aparecen simultáneamente en el vector.

Análogamente tendremos:

$$E(\sim i,l) = \sum_{j=1}^{M} (1 - V_{i,j}) V_{l,j}$$

$$E(i,\sim l) = \sum_{j=1}^{M} V_{i,j} (1 - V_{l,j})$$

$$E(\sim i,\sim l) = \sum_{j=1}^{M} (1 - V_{i,j})(1 - V_{l,j})$$

Si los atributos i y l fueran independientes debería valer que:

$$P(i,l) = P(i)\, P(l)$$

Pero:

$$P(i) = \frac{E(i,l) + E(i,\sim l)}{E(i,l) + E(i,\sim l) + E(\sim i,l) + E(\sim i,\sim l)}$$

Y

$$P(l) = \frac{E(i,l) + E(\sim i,l)}{E(i,l) + E(i,\sim l) + E(\sim i,l) + E(\sim i,\sim l)}$$

Por otra parte :

$$P(i,l) = \frac{E(i,l)}{E(i,l) + E(i,\sim l) + E(\sim i,l) + E(\sim i,\sim l)}$$

Si considero la variable aleatoria:

$$\left(\frac{P(i,l) - P(i)P(l)}{P(i,l)} + \frac{P(\sim i,l) - P(\sim i)P(l)}{P(\sim i,l)} + \frac{P(i,\sim l) - P(i)P(\sim l)}{P(i,\sim l)} + \frac{P(\sim i,\sim l) - P(\sim i)P(\sim l)}{P(\sim i,\sim l)}\right)^2$$

Debe seguir una distribución χ^2 para 1 grado de libertad.

El valor así obtenido a partir de los datos del vector se compara con el valor de la distribución ${\chi_1}^2$ acumulada.

Si da muy grande o muy cercana a cero la probabilidad resultará pequeña y corresponderá rechazar la hipótesis de independencia.

El valor de la variable aleatoria así definida resulta dependiente del tamaño de la muestra. Sin embargo, dentro de la misma base de datos podemos usarla como una medida de la dependencia.

Podemos así elegir las reglas que tengan la mayor medida de dependencia.

Así podemos construir primero las reglas de asociación para un nivel de cobertura y confianza límites y luego ordenarlas por su nivel de dependencia.

Ejercicio 12.2:

Tomar las reglas obtenidas en el ejercicio 7.1 y ordenarlas desde las más dependientes a las más independientes.

Matriz de Interés:

El interés entre un par de atributos se define como:

$$I(i,l) = \frac{P(i,l)}{P(i)P(l)}$$

Así definido, cuando I es mayor a 1 entonces existe una fuerte dependencia positiva. Cuando es menor a 1 la dependencia es negativa. Alrededor de 1 significa que es compatible con la independencia. (Si se quieren asignar niveles de confianza esto también sigue la distribución χ^2 para 1 grado de libertad)

Podemos entonces construir una matriz con:

$$\begin{bmatrix} I(i,l) & I(\sim i,l) \\ I(i,\sim l) & I(\sim i,\sim l) \end{bmatrix}$$

Así podemos advertir las posibles dependencias entre los atributos y sus complementos.

Ejercicio 12.3
Encontrar i y l para los datos del ejercicio 12.1 tales que obtenga el máximo valor dentro de alguna matriz de interés.

Aprendizaje supervisado y no supervisado:

El **aprendizaje** con **reglas de asociación** lo vemos aplicado principalmente en los sistemas de recomendación, como en el caso donde se nos muestra que las personas que compraron este producto, también compraron este otro o quienes vieron tal película también recomiendan estas otras, etc.

Para ello, el algoritmo **a priori** es uno de los más utilizados en este tema y permite encontrar de forma eficiente conjuntos de **ítems frecuentes***,* los cuales sirven de base para generar **reglas de asociación** entre los ítems.
Primero identifica los **ítems** individuales frecuentes dentro del conjunto de datos para luego extenderlo a un conjunto de mayor tamaño siempre y cuando esos conjuntos de datos aparezcan constantemente y de manera frecuente de acuerdo con un **umbral** establecido.

El algoritmo se aplica principalmente en el análisis de transacciones comerciales y en los problemas de predicción. Es por ello que el algoritmo está diseñado para trabajar con bases de datos que contienen transacciones como los productos o artículos comprados por consumidores, o detalles sobre las visitas a un sitio web, etc.
La forma de generar las **reglas de asociación** consta de dos pasos:

- **Generación de combinaciones frecuentes**: cuyo objetivo es encontrar aquellos conjuntos que sean frecuentes en la base de datos. Para determinar la frecuencia se establece un umbral.

- **Generación de reglas**: A partir de los conjuntos frecuentes se crean las reglas en base al ordenamiento de un índice que establece los grupos de ítems o productos frecuentes.

El índice para la generación de combinaciones se llama **soporte** y el índice para la generación de reglas se llama **confianza**.

Algoritmo

- **Paso 1**. Se establecen los valores mínimos para el soporte y la confianza
- **Paso 2**. Se toman todos los subconjuntos de transacciones que tienen un soporte mayor al valor del soporte mínimo.
- **Paso 3**. Tomar todas las reglas de estos subconjuntos que tengan una confianza mayor al valor de la confianza mínima.
- **Paso 4.** Ordenar las reglas de forma decreciente en base al valor del lift.

Antes de entrar en los detalles del algoritmo, conviene definir una serie de conceptos:

- Soporte: El soporte del item o itemset X es el número de transacciones que contienen X dividido entre el total de transacciones.
- Confianza: La confianza de una regla "Si X entonces Y" se define acorde a la ecuación

Confianza(X=>Y)=Soporte(unión(X,Y)) /soporte(X)
Donde unión (XY) es el itemset que contienen todos los ítems de X y de Y. La confianza se interpreta como la probabilidad P(Y|X), es decir, la probabilidad de que una transacción que contiene los ítems de X, también contenga los ítems de Y.

Evaluación de las reglas

Además de la confianza y el soporte, existen otras métricas que permiten cuantificar la calidad de las reglas y la probabilidad de que reflejen relaciones reales. Algunas de las más empleadas son:

- Lift*:* el estadístico lift compara la frecuencia observada de una regla con la frecuencia esperada simplemente por azar (si la regla no existe realmente). El valor lift de una regla *"*si X, entonces Y*"* se obtiene acorde a la siguiente ecuación:

soporte(union(X,Y)) / soporte(X) * soporte(Y)
Cuanto más se aleje el valor de lift de 1, más evidencias de que la regla no se debe a un artefacto aleatorio, es decir, mayor la evidencia de que la regla representa un patrón real.

- Coverage: es el soporte de la parte izquierda de la regla (antecedente). Se interpreta como la frecuencia con la que el antecedente aparece en el conjunto de transacciones.

- Fisher exact test*:* devuelve el p-value asociado a la probabilidad de observar la regla solo por azar.

A continuación veremos un ejemplo sencillo con 5 transacciones para entender conceptualmente la aplicación de las reglas de asociación para luego seleccionar una lista más grande desde un archivo csv y aplicarlo en Python

Si tenemos un conjunto de 5 transacciones con diversos productos en cada una de ellas de acuerdo con la siguiente tabla

1	Pan, leche, pañales
2	Pan, pañales, cerveza, huevo
3	Leche, pañales, cerveza, Gaseosa, café
4	Pan, leche, pañales, cerveza
5	Pan, Gaseosa, leche, pañales

El primer paso es generar las combinaciones frecuentes, y, si queremos un soporte superior al 50%, entonces contamos la frecuencia de cada uno de los artículos, es decir, en cuantas transacciones aparecen cada uno de los artículos.

Artículo	**Transacciones**
Cerveza	3
Pan	4
Gaseosa	2
Pañales	5
Leche	4
Huevo	1
Café	1

Para calcular el soporte de cada artículo dividimos la cantidad de transacciones de cada artículo, entre el total de transacciones. Es decir, para cerveza tenemos que aparece en 3 de las 5 transacciones, entonces es 3/5 = 0.6 que representa el 60%. Para el resto de los artículos tenemos lo siguiente:

Artículo	**Soporte**
Cerveza	60%
Pan	80%
Gaseosa	40%
Pañales	100%
Leche	80%
Huevo	20%
Café	20%

Como se requiere un soporte superior al 50% entonces eliminamos todos los artículos que están por abajo de este umbral: Gaseosa, huevo y café.
El siguiente paso es generar las combinaciones con los productos que quedaron para iterar primero con combinaciones de dos, calculamos el soporte y después con combinaciones de 3 y así sucesivamente.

Conjuntos	**Frecuencia**	**Soporte**
Cerveza, Pan	2	40%
Cerveza, Pañales	3	60%
Cerveza, Leche	2	40%
Pan, Pañales	4	80%
Pan, Leche	3	60%
Pañales, Leche	4	80%

Eliminamos los que están por abajo del 50% y nos quedamos con los primeros conjuntos frecuentes cuyo soporte es superior al 50%

- Cerveza, Pañales
- Pan, Pañales
- Pan, leche
- Pañales, Leche

A partir de los conjuntos generados, creamos conjuntos de tres artículos y calculamos su soporte

Conjuntos	**Frecuencia**	**Soporte**
Cerveza, Pañales, Pan	2	40%
Cerveza, Pañales, Leche	2	40%
Pan, Pañales, Leche	3	60%
Pan, Leche, Cerveza	1	20%

En estas combinaciones de tres, únicamente nos quedamos con el conjunto formado por Pan, Pañales y Leche, el cual utilizamos para hacer combinaciones de 4 artículos, sin embargo para este caso, tienen soporte de 20% por lo cual, aquí termina el algoritmo.

El resultado arrojo un elemento de 3 artículos y cuatro de 2 artículos:

Pan, Pañales, Leche
Cerveza, Pañales
Pan, Pañales
Pan, Leche
Pañales, Leche

A partir de estos 5 conjuntos obtenemos la reglas de asociación, para lo cual, establecemos que también queremos un índice superior 50%. Este índice es la confianza y lo calculamos dividiendo las repeticiones de las observaciones del conjunto entre las repeticiones de la regla:
Tomando el primer conjunto de Pan, Pañales, Leche, las reglas posibles son:

- Pan => Pañales, Leche
- Pañales => Pan, Leche
- Leche = Pan, Pañales
- Pan, Pañales => Leche
- Pan, Leche => Pañales
- Leche, Pañales => Pan

Si tomamos la primer regla: Pan => Pañales, Leche observamos que en las transacciones originales que Pan, Pañales, Leche aparece en 3 transacciones y la regla Pan aparece en 4 transacciones, entonces la confianza es 3/4 = 0.75 que es el 75%
Para la regla formada por: Pan, Pañales => Leche tenemos que la combinación Pan, Pañales, Leche aparece en 3 transacciones y la regla Pañales, Leche en 4 transacciones por lo que su confianza es del 75% también, es decir 3/4 = 0.75
Una vez que calculamos las confianza de todas las reglas, las ordenamos de mayor a menor en base a esa confianza calculada y obtenemos las reglas de asociación para todo el conjunto, que es cómo funciona el algoritmo **A Priori**.

Construyendo el código Python – Instalación y uso de paquetes

Para la aplicación en Python vamos a seguir con cesta de compra pero en este caso vamos a tomar un lote más grande tomando los datos de un archivo tipo csv ***'Market_Basket_Optimisation.csv'***

- Numpy
- Pandas
- Matplotlib.pyplot
- Apriori, Apyori

```
# Cargamos las librerias necesarias
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Importamos el archivo de transacciones
dataset = pd.read_csv('Market_Basket_Optimisation.csv', header = None)
transactions = []
for i in range(0, 7501):
  transactions.append([str(dataset.values[i,j]) for j in range(0, 20)])

# Training Apriori on the dataset
from apyori import apriori
rules = apriori(transactions, min_support = 0.003, min_confidence = 0.2, min_lift = 3, min_length = 2)

# Resultados
results = list(rules)
def inspect(results):
  rh        = [tuple(result[2][0][0]) for result in results]
  lh        = [tuple(result[2][0][1]) for result in results]
  supports    = [result[1] for result in results]
  confidences = [result[2][0][2] for result in results]
  lifts      = [result[2][0][3] for result in results]
  return list(zip(rh, lh, supports, confidences, lifts))

# Este comamdo crea un frame para ver los datos resultados
resultDataFrame=pd.DataFrame(inspect(results),
          columns=['rhs','lhs','support','confidence','lift'])

#Imprimimos el frame con las reglas
print (resultDataFrame)
```

Antes de ejecutar el archivo con el Script desarrollado en Sublime debemos instalar el paquete apyori, luego invocamos al script.

```
C:\Users\susan\OneDrive\Documentos\Python>pip install apyori
Collecting apyori
  Downloading https://files.pythonhosted.org/packages/5e/62/5ffde5c473ea4b033490617ec5caa80d59804875ad3c3c57c0976533a21a/apyori-1.1.2.tar.gz
Installing collected packages: apyori
  Running setup.py install for apyori ... done
Successfully installed apyori-1.1.2
WARNING: You are using pip version 19.2.3, however version 20.0.2 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\susan\OneDrive\Documentos\Python>python reglasdeasociacion.Py
```

Obtenemos el siguiente Resultado:

```
C:\Users\susan\OneDrive\Documentos\Python>python reglasdeasociacion.Py
                              rhs                                lhs   support  confidence      lift
0                    (light cream,)                      (chicken,)  0.004533    0.290598  4.843951
1            (mushroom cream sauce,)                     (escalope,)  0.005733    0.300699  3.790833
2                          (pasta,)                     (escalope,)  0.005866    0.372881  4.700812
3                  (fromage blanc,)                       (honey,)  0.003333    0.245098  5.164271
4                  (herb & pepper,)                  (ground beef,)  0.015998    0.323450  3.291994
..                              ...                             ...       ...         ...       ...
155         (olive oil, ground beef)  (mineral water, nan, spaghetti)  0.003066    0.216981  3.632981
156          (pancakes, ground beef)  (mineral water, nan, spaghetti)  0.003066    0.211009  3.532991
157          (tomatoes, ground beef)  (mineral water, nan, spaghetti)  0.003066    0.261364  4.376091
158  (mineral water, milk, spaghetti)               (olive oil, nan)  0.003333    0.211864  3.223519
159                 (milk, tomatoes)  (mineral water, nan, spaghetti)  0.003333    0.238095  3.986501

[160 rows x 5 columns]
```

Observamos las reglas resultantes con 2, 3 o más artículos que implican otro grupo de productos y tenemos también el soporte, confianza y el lift.

Interpretación de los resultados

Los resultados que se han obtenido dicen cosas interesantes. Analizando la primera regla se puede observar que los clientes que han comprado crema light también aparece pollo con un soporte de 0,045 y una confianza de 0,29. Esto indica que en el 4,5% de las transacciones contienen ambas referencias. Además, la confianza indica que en el 29% de los casos que se compra crema light también aparece pollo.
La mejora de la confianza (lift) de 4,84 indica que cuando aparece crema light aparece la otra referencia 4 veces más de lo que se podría esperar por azar.
Recordando una de las definiciones arriba decíamos que “Cuanto más se aleje el valor de *lift* de 1, más evidencias de que la regla no se debe a un artefacto aleatorio, es decir, mayor la evidencia de que la regla representa un patrón real.”

Enunciado Ejercicio 12.4

Buscar un ejemplo propio en el que se pueda aplicar el concepto de Regla de Asociación en Python.