INF391 Reconocimiento de Patrones en Minería de Datos
Tarea 2

Universidad Técnica Federico Santa María, Campus San Joaquín
Departamento de Informática

8 de septiembre de 2015

Profesor Marcelo Mendoza

Juan Pablo Escalona
juan.escalonag@alumnos.usm.cl
201073515-k

Rafik Masad
rafik.masad@alumnos.usm.cl
201073519-2

Gianfranco Valentino
gianfranco.valentino@alumnos.usm.cl
2860574-9

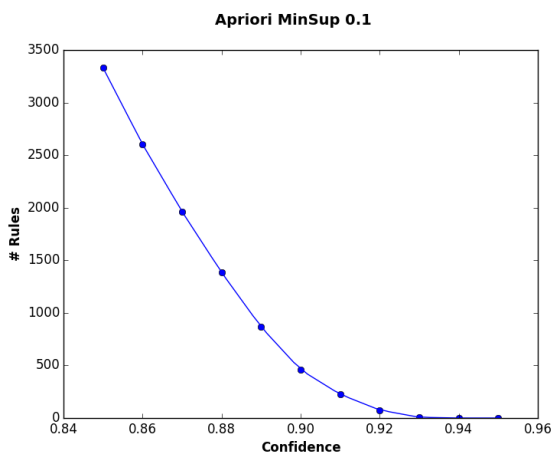
1. Introducción

En el presente informe se comparan los algoritmos Apriori y FP-growth buscando reglas de asociación sobre el dataset de *supermarket.arff* utilizando el software Weka[1]. La idea es comparar los comportamientos de ambos algoritmos al variar los parámetros de soporte mínimo y confianza.

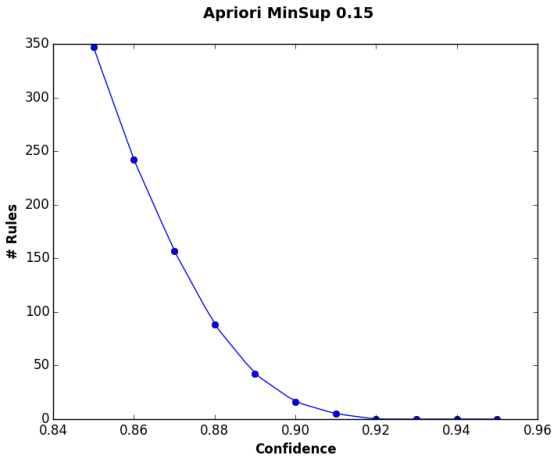
Para el desarrollo de la experiencia se escribió un script en python capaz de ejecutar los diferentes algoritmos con los parámetros a probar utilizando Weka. Este script genera gráficos interesantes de salida, los cuales se analizan en el informe. Todos los scripts creados para esta experiencia se encuentran en 5.

2. Algoritmo Apriori

En este experimento se gráfica la confianza entre 0.85 y 0.95 utilizando un soporte mínimo de 0.1 y 0.15. En el eje Y se presenta el número de reglas encontradas para cada una de las confianzas.



(a) Algoritmo Apriori: Confianza vs Número de reglas encontradas con soporte mínimo 0.1



(b) Algoritmo Apriori: Confianza vs Número de reglas encontradas con soporte mínimo 0.15

Es importante destacar la diferencia entre los ejes y de ambos gráficos, hay una diferencia en magnitud de orden 10.

2.1. Reglas interesantes

2.1.1. El monopolio del pan y pasteles

En un alto porcentaje de las reglas encontradas, sobre todo las de más alto porcentaje de confianza, encontramos que una combinación de compras implica la compra de pan o pasteles. Lo anterior nos da a inferir que estos productos son altamente consumidos y un porcentaje alto de los consumidores al comprar algún producto llevan además pan o pasteles ya que es algo que es necesario constantemente en el hogar.

2.1.2. ¿Vegetales?

En una situación similar pero en menor medida a la de los panes y pasteles existen múltiples relaciones entre distintas combinaciones de compras y vegetales. En particular se repite bastante combinaciones que incluyen instrumentos de cocina. Al igual que los productos anteriores nos da a inferir que estos productos son altamente consumidos y un porcentaje alto de los consumidores al comprar algún producto llevan además vegetales.

[1] http://www.cs.waikato.ac.nz/ml/weka/

2.1.3. Desayuno de campeones

Con una confianza del 87 % podemos determinar que la gente que compra comida para el desayuno, comida congelada y margarina o pañuelos de papel, compran galletas. Lo anterior lo podemos asociar a un comportamiento cultural propio del país de origen de los datos, Estados Unidos, donde la gente que va a comprar comida para el desayuno y congelados, compran galletas.

2.1.4. Comida de solteros

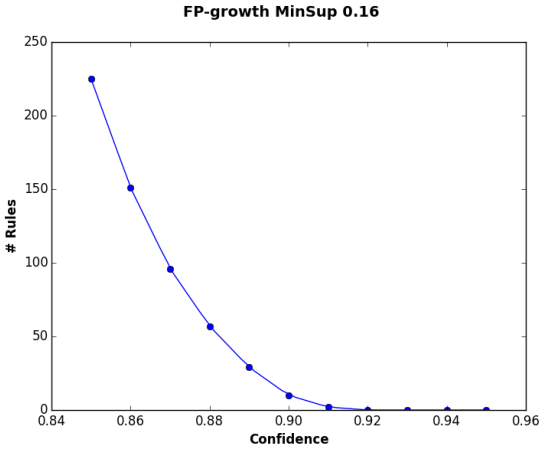
Con una confianza del 88 % los que compran galletas y comida preparada también compran comida congelada. Lo anterior lo podemos asociar a que los que compran comida preparada compran por la misma razón comida congelada.

2.1.5. Asado... y galletas

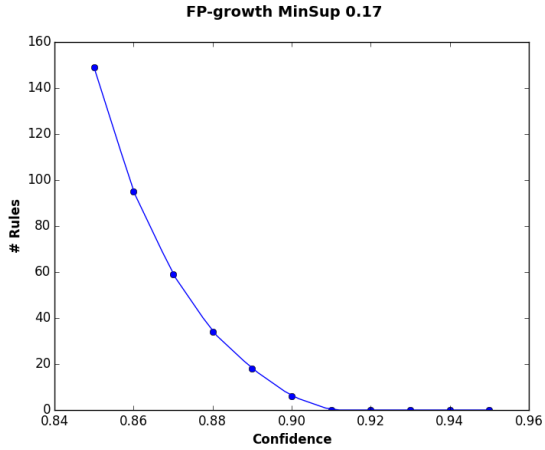
Con una confianza del 87 % los que compran galletas, carne y herramientas para cocinar o vegetales, compran, además, comida congelada. Lo anterior lo podemos asociar a que cuando compran carne y implementos para el asado generalmente lo acompañan con alguna comida congelada fácil de cocinar.

3. Algoritmo FP-Growth

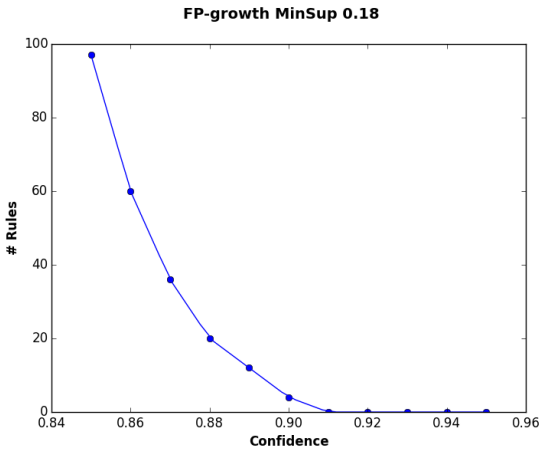
Para este experimento se hace variar la confianza mínima entre 0.85 a 0.95 para cada uno de los soportes mínimos entre 0.16 y 0.25. Se generan entonces 10 gráficos con diferentes soportes mínimos, en el eje X se presenta la confianza entre 0.85 y 0.95. El eje Y representa el número de reglas encontradas.



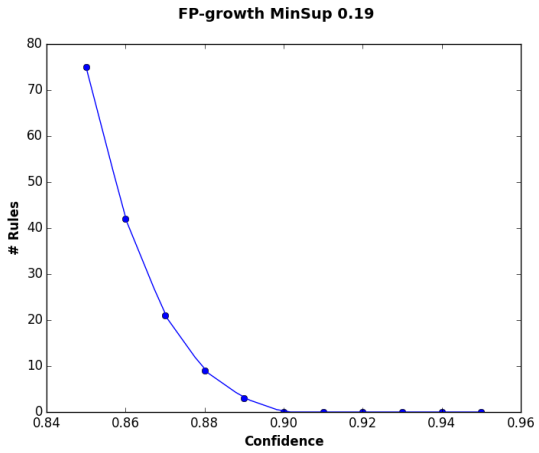
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.16



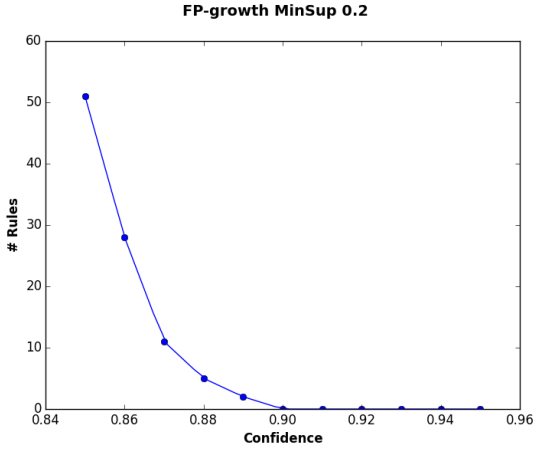
(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.17



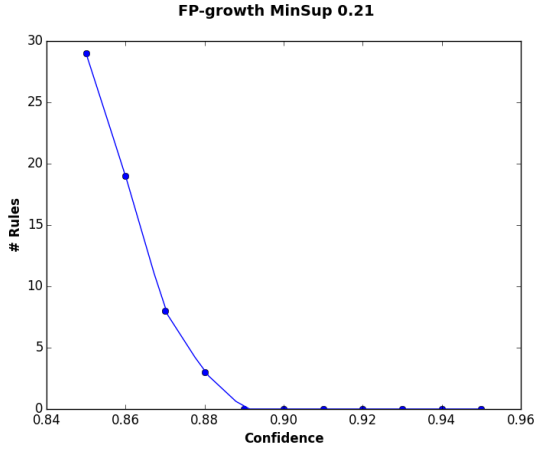
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.18



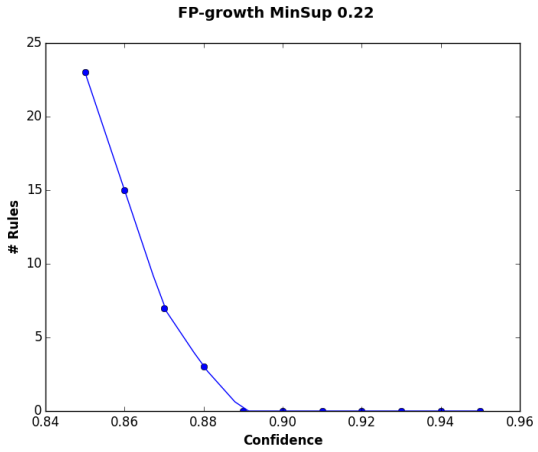
(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.19



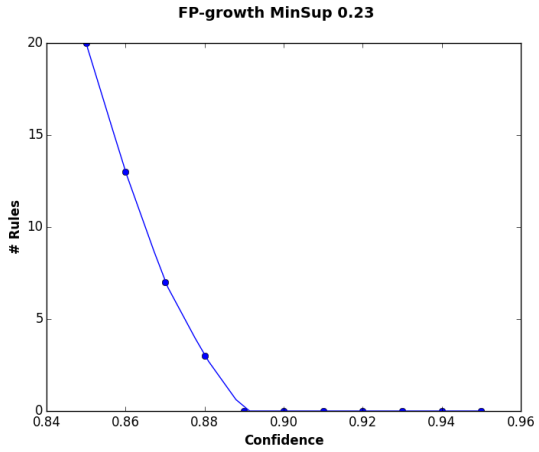
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.20



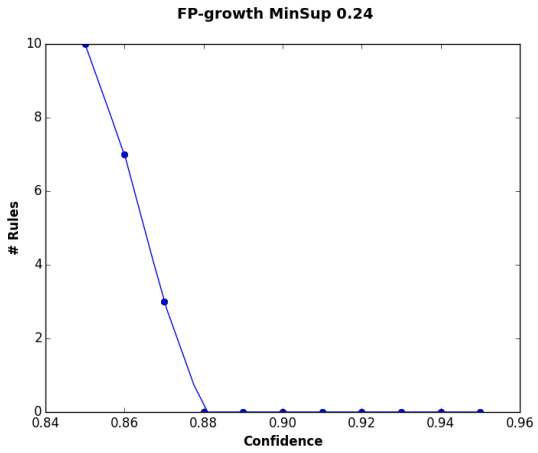
(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.21



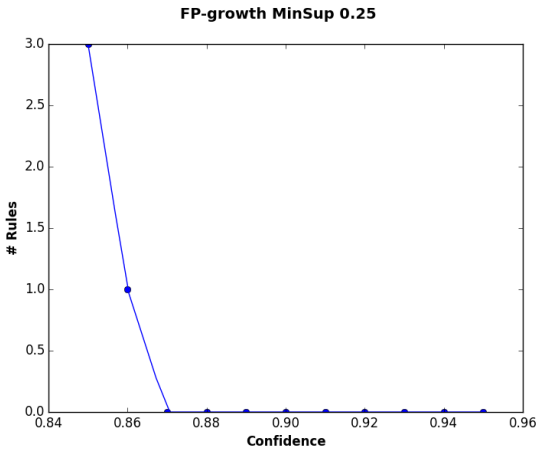
(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.22



(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.23



(a) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.24



(b) Algoritmo FP-growth: Confianza vs Número de reglas encontradas con soporte mínimo 0.25

4. Conclusiones

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

5. Anexo

Script 1: Generador de gráficos para todos los parámetros

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-

import matplotlib.pyplot as plt
import subprocess
from scipy.interpolate import interp1d
import numpy as np
import re
import os.path



def plotter(x, y, filename = 'foo.png', interp_points = 30, title = 'Alg wtih ↙
    ↳ Min Sup = minsup', x_title = 'Confidence', y_title = '# Rules'):
    x_interp  = np.linspace(min(x), max(x), interp_points)
    y_interp = np.interp(x_interp, x, y)

    fig, ax1 = plt.subplots()
    ax1.set_xlabel(x_title, fontweight = 'bold')
    ax1.set_ylabel(y_title, fontweight = 'bold')
    fig.suptitle(title, fontsize = 14, fontweight = 'bold')
    # ax1.set_ylim([0, 250])
    ax1.plot(x, y, 'bo', x_interp, y_interp, 'b-')
```

```
23
24      plt.savefig(filename)
25
26
27  INPUT = "supermarket.arff"
28  RUN = "./run.sh"
29  FILTER = "./filter.sh"
30  MATCHER = re.compile(r'.*?([\d]+)\. .*')
31  filtered = "supermarket-filtered.arff"
32  filter_indexes = map(lambda x: str(x) ,
33                           [
34                               # 13, # bread and cake
35                               # 86  # vegetables
36                           ])
37  if filter_indexes:
38      cmd = [FILTER, INPUT, filtered,  ",".join(filter_indexes)]
39      subprocess.call(cmd)
40  else:
41      filtered = INPUT
42  c_float = map(lambda x: x/100.0, range(85, 96))
43  c = map(lambda x: str(x/100.0), range(85, 96))
44  alg_minsup = {
45      "Apriori": ['0.1', '0.15'],
46      "FP-growth": map(lambda x: str(x/100.0), range(16, 26))
47  }
48
49  alg_type = {"Apriori":"1", "FP-growth":"2"}
50
51  for alg, minsups in alg_minsup.items():
52      for minsup in minsups:
53          filename = None
54          ys = []
55          for conf in c:
56              filename = 'outs/'+'-'.join([alg,minsup,conf])+"-out.txt"
57
58              if not os.path.isfile(filename):
59                  cmd = [RUN, filtered, filename, conf, minsup, alg_type[alg]];
60                  subprocess.call(cmd)
61
62              weka_file = open(filename)
63              last = 0
64              for line in weka_file:
65                  mt = MATCHER.match(line)
66
67                  if mt:
68                      last = mt.groups()[-1]
69              weka_file.close()
70              ys.append(last)
71          graph_name =  alg+"-"+minsup
72          plotter(c_float, ys, filename='img/'+graph_name+".png",  title=alg+" ↙
                ↳ MinSup "+minsup)
```

Script 2: Ejecuta Weka dado los parámetros indicados

```
1  #!/bin/bash
2
3  WEKA_PATH=.
4
5  CP="$CLASSPATH:$WEKA_PATH/weka.jar"
```

```
if [[ "$5" -eq 1  ]]; then
    java -cp $CP weka.associations.Apriori -N 5000 -T 0 -C $3 -D 0.05 -U 1.0 -M ↙
        ↳ $4 -S -1.0 -c -1 -t $1 > $2
else
    java -cp $CP weka.associations.FPGrowth -P 2 -I -1 -N 5000 -T 0 -C $3 -D ↙
        ↳ 0.05 -U 1.0 -M $4 -t $1 > $2
fi
```

Script 3: Filtra elementos específicos del dataset

```
#!/bin/bash

WEKA_PATH=.

CP="$CLASSPATH:$WEKA_PATH/weka.jar"

 java -cp $CP  weka.filters.unsupervised.attribute.Remove -R $3 -i $1 -o $2
```