Predicción de Compradores Habituales de una Promoción de Ventas

Javier Lara Padron: 93343 Mail: jlara91@gmail.com

Di Tomaso Nicolas Padron: 88408 Mail: ditomaso.nh@gmail.com

Vazquez Diego Padron: 78709 Mail: vazquez.diego@gmail.com

Facultad de Ingeniería, Universidad de Buenos Aires, Buenos Aires, Argentina

**Abstract.** En el presente trabajo, se mostrará una manera de poder predecir compradores habituales luego de una promoción de ventas. Se explicará la solución propuesta en base a redes neuronales, utilizando una librería del lenguaje de prog,ramación Python.

Introducción

Las ofertas en los precios suelen servir como publicidad para un comerciante.

A su vez los clientes viven buscando y comparando precios de eso que quieren comprar y esperando el momento oportuno para poder adquirirlo.

Al cliente, este sistema de promociones por supuesto que le conviene. Entonces, ¿qué beneficio tiene el vendedor a la hora de ofrecer sus productos a un precio menor? Uno de los objetivos principales es poder absorber nuevos clientes, dándose a conocer a través de las ofertas.

Sin embargo, muchos clientes son oportunistas. Aprovechan estas ofertas y no compran nunca más.

Por este motivo, cualquier vendedor desea tener la posibilidad de predecir cuál cliente se convertirá luego en habitual.

Problema

Luego de una promoción de ventas (ya sea por navidad o el famoso “Black Friday”). El objetivo de estas ofertas es atraer nuevos clientes.

Sin embargo, muchos clientes son oportunistas. Aprovechan estas ofertas y no

compran nunca más. Por supuesto que al ser precios más bajos, no impacta en absoluto en las ventas.

Es por eso que para los vendedores, sería muy importante identificar los clientes que se convierten en habituales. De esta manera, podrían reducir el costo de la promoción y mejorar el retorno sobre la inversión (ROI).

Sabiendo que es muy difícil esta tarea, se obtuvo una base de datos de Tmall.com, donde se poseen los datos de muchas compras realizadas en este sitio. En particular, son las compras realizadas por nuevos compradores durante la promoción del día “Double 11”, una promoción famosa en los Estados Unidos.

El objetivo es poder predecir cuáles de estos nuevos compradores, se convertirán en compradores habituales en el futuro.

Solución Propuesta

Para poder solucionar el problema, se utilizará una red neuronal que prediga los compradores habituales.

Se decidió utilizar el lenguaje de programación Python, ya que posee una librería llamada Pybrain[[1]](#footnote-1), la cual facilita el armado y el manejo de redes neuronales.

Datos

El set de datos brindado por Tmall.com consta de aproximadamente 7 millones de registros, los cuales poseen el siguiente formato:

**Table 1.** Formato de datos

|  |  |
| --- | --- |
| Nombre del campo | Definición |
| USER\_ID | Identificación única del usuario |
| AGE\_RANGE | Rango de edad   * 1: menor a 18 * 2: entre 18 y 24 * 3: entre 25 y 29 * 4: entre 30 y 34 * 5: entre 35 y 39 * 6: entre 40 y 49 * 7 y 8: mayor o igual a 50 * 0: desconocido |
| GENDER | Sexo   * 0: femenino * 1: masculino * 2: desconocido |
| MERCHANT\_ID | Identificación única del vendedor |
| LABEL | Clasificación entre habituales o no   * 1: habitual * 0: no habitual * -1: desconocido |
| ACTIVITY\_LOG | Interacciones entre el usuario y el vendedor |

Preparación de los datos

Para poder tener un mejor funcionamiento de la red, se decidió optimizar este set de datos, generando uno nuevo más ¨limpio¨ .

Se generó un script de preprocesamiento de los datos el cual elimina el ¨user\_id¨ y el ¨activity\_log¨ ya que no aportan información.

Además, se quitaron completos todos los registros que tengan algún campo con el valor desconocido, ya que dificultan la predicción.

De esta manera, los campos de entrada de la red serán: AGE\_RANGE, USER\_ID y GENDER y de salida: LABEL.

Para que la predicción sea equitativa, se decidió tomar la misma cantidad de casos positivos que de negativos. Es por eso que de la muestra, quedaron 12600 de cada uno. Todos estos tomados aleatoriamente de los resultantes del proceso anterior.

Estos son ordenados aleatoriamente, de manera que no sean consecutivos los distintos registros clasificados iguales.

Posteriormente, se toma como datos de entrenamiento la mitad del resultado de este proceso, y como datos de prueba, la otra mitad.

El script de preprocesamiento es el siguiente.

preprocessData.py

import csv

from random import shuffle

fin = open('data\_format2/train\_format2.csv','r')

fout = open('train\_format2\_processed.csv','w')

writer = csv.writer(fout, delimiter=',')

# Copio header en nuevos datos

line = fin.readline()

writer.writerow(line.strip().split(',')[1:5])

# Preprocesamiento: se sacan registros con valores vacios o nulos y el activity\_log y el user\_id

repeatBuyers = []

nonRepeatBuyers = []

print "Starting"

for line in fin.readlines():

data = [x for x in line.strip().split(',')[1:5] if x != '']

# Se sacan los registros que tengan valores que esten marcados como desconocidos segun el formato de cada atributo

if len(data) < 4 or data[0] == '0' or data[1] == '2' or data[3] == '-1':

continue

if data[3] == '1':

repeatBuyers.append(data)

elif data[3] == '0':

nonRepeatBuyers.append(data);

# Tomo aleatoriamente la misma cantidad de registros que no se repitan

shuffle(nonRepeatBuyers)

nonRepeatBuyers = nonRepeatBuyers[:len(repeatBuyers)]

# Los ordeno aleatoriamente y se escriben en nuevo archivo de datos

toWrite = repeatBuyers + nonRepeatBuyers

shuffle(toWrite)

writer.writerows(toWrite)

print "Repeat: " + str(len(repeatBuyers)) + " No Repeat: " + str(len(nonRepeatBuyers))

print "Finished"

Red Neuronal

Se creó una red neuronal recurrente con una configuración de 3-3-2-1, es decir, una capa de entrada de 3 neuronas, una capa oculta de 3 neuronas, otra capa oculta de dos neuronas y una capa de salida de 1 neurona.

Luego la red es entrenada con un algoritmo de Backpropagation, con un máximo de 30 ciclos de entrenamiento.

Código

El código en python es el siguiente

main.py

from pybrain.tools.shortcuts import buildNetwork

from pybrain.datasets import SupervisedDataSet

from pybrain.supervised.trainers import BackpropTrainer

from pybrain.structure import TanhLayer

import csv

import numpy as np

ds = SupervisedDataSet(3, 1)

fin = open('train\_format2\_processed.csv','r')

num\_lines = sum(1 for line in open('train\_format2\_processed.csv','r'))

fin.readline()

length = (num\_lines - 1) / 2

print "Reading data"

i = 0

testData = []

for line in fin.readlines():

if i < length: # tomo una muestra de datos para el entrenamiento

trainRow = [float(x) for x in line.strip().split(',')]

ds.addSample(tuple(trainRow[:3]),tuple(trainRow[3:]))

i = i + 1

elif i < length \* 2: # tomo una muestra de datos para las pruebas

testRow = [float(x) for x in line.strip().split(',')]

testData.append(testRow)

i = i + 1

else:

break

# Normalizo los atributos

i = np.array([d[0] for d in ds])

i /= np.max(np.abs(i),axis=0)

o = np.array([d[1] for d in ds])

o /= np.max(np.abs(o),axis=0)

test = np.array([row for row in testData])

test /= np.max(np.abs(test), axis=0)

nds = SupervisedDataSet(3, 1)

for ix in range(len(ds)):

nds.addSample( i[ix], o[ix])

net = buildNetwork(nds.indim,3,2,nds.outdim, recurrent=True)

t = BackpropTrainer(net,verbose=True)

print "Training"

t.trainUntilConvergence(nds,verbose=True, maxEpochs=30)

print "Testing"

fout = open('results.csv','w')

writer = csv.writer(fout, delimiter=',')

correct = 0

total = 0

for testRow in test:

predicted = net.activate(testRow[:3])[0]

roundedPredicted = round(predicted)

writer.writerow(np.append(testRow, [roundedPredicted,predicted]))

if roundedPredicted == testRow[3]:

correct = correct + 1

total = total + 1

print "Precission: " + str(float(correct)/float(total))

Resultados

Los resultados obtenidos son guardados en un archivo csv, para luego poder ser consultados. En la mayoría de los casos (se generaron los datos, la red y el entrenamiento varias veces) la precisión obtenida por la red promedia el 52%.

Conclusiones

Si bien el porcentaje de precisión no es muy elevado, sirve como un comienzo para poder predecir los posibles clientes. Con poder reconocer la mitad de los clientes que van a volver a comprar, según los cálculos de precisión obtenidos, es un avance enorme para los vendedores. Además debido a que se obtuvo una precisión de solo 0,52 lo único que podemos afirmar es que los campos Age\_Range, Gender y Merchant\_Id no son suficientes para predecir que un cliente será un cliente reiterativo.

Este trabajo puede servir como un comienzo de un análisis más profundo de este fenómeno, a simple vista impredecible, de las acciones de los consumidores.

Con más datos precisos y un análisis más exhaustivo de la configuración de la red, seguramente el nivel de precisión puede llegar a ser muy elevado.

1. <http://pybrain.org/> [↑](#footnote-ref-1)