LAB 3 – SOCIAL RECOMENDATION

COMP-47270 COMPUTATIONAL NETWORK ANALYSIS AND MODELLING

Author :	*Paula Dwan* *paula.dwan@gmail.com*
Student ID :	*13208660*
Lecturer :	*Dr. Neil Hurley* *neil.hurley@ucd.ie*
Due Date :	*Sunday, 19 April 2015*

Table of Contents

LABORATORY 3 – BASIC REQUIREMENTS

Download Octave in order to run the collaborative filtering code CFsimple.m.

This is a user-based code for making predictions given a database of user item ratings.

Initially try some simple modications to this code :

- Modify the parameters. Can you change the similarity metric?
- How do these changes impact on performance?
- Can you change the code to calculate a top-N recommendation rather than rating prediction.

Ultimately, the goal is to augment this code with trust metric data in order to improve the recommendation.

1 : CHANGE VALUE OF K

2 : CHANGE SIMILARITY METRIC

3 : CHANGES IMPACT PERFORMANCE

AS PROVIDED

Source code – cfsimple.m (as edited)

```
function CollaborativeFiltering

% Y = load('../RatingsDataSets/ml.dat');                    % PD : uses ratings / trust for epinions
Y = load('../RatingsDataSets/epinions_rating_3.txt');      % PD : remove unneeded columns
p = randperm(length(Y));                                   % use random permutations of Y into p

Y(:,1) = Y(p,1);
Y(:,2) = Y(p,2);
Y(:,3) = Y(p,3);

numTrans = length(Y);                % PD : initialise varaibles
division=floor(numTrans/5);   % split into 5 sets; PD : see notes & changed to division
first=1;
last = division;
sumAbsErr = 0;
totalTrans = 0;

fprintf('\n\nTO DATE : \n\t MAE \t\t Total Translactions
   \n\t------------------------------------------------------ \n');

for I = 1:5,

   testY = Y(first:last,:);
   trainY = [Y(1:(first-1),:);Y((last+1):end,:)];

   first = first + division;  % PD : previously first = first + last
   last = last + division;           % PD : previously last = last + last

   trainSet = sparse(trainY(:,1),trainY(:,2),trainY(:,3));
   % PD : same data represented in different manner
   testSet = sparse(testY(:,1),testY(:,2),testY(:,3));

   for trans=1:length(testY),

      activeUser = testY(trans,1);
      activeItem = testY(trans,2);
      activeRating = testY(trans,3);
```

```
      sim = computeSimilarities(activeUser,trainSet);
      % PD : compute similarity between active user and all training users

      sim(activeUser) = -1;
      % PD : high numbers -- more similar / low numbers -- less similar
      % PD : activeUser = 692 say : here set to -1 to move to bottom of list -- least similar
      k = 10;
      % PD : arbitary - usually take k = 20, can use k=100, redo until best k
      mask = trainSet(:,activeItem) > 0;
      % PD : going through training set and checking what rating users have given to item,
      % PD : if =0 then no rating given
      sim(mask==0) = -1;
      % PD : again set similarity values of non-rating users to -1 --- move to bottom of list also
      [s,indx] = sort(sim,'descend');
      % PD : vector in sorted order, plus index array list of neighbors who have best
      % PD : similarity as descending order applied
      neighbours = indx(1:k);
      % PD : pick 20 neighbors who are most similar to activeUser
      mask = sim(neighbours) > 0;
      % PD : be careful there may not be 20 neighbors who have rated item, so remove anyone
      % PD : who has rating of zero / did not rate item
      neighbours = neighbours(mask); % PD : this is our neighbors rating listing

      if (length(neighbours) == 0)
      % PD : if no neighbors rated item - then use average / mid-point
         predictRating = 3;
      else
         predictRating = round(mean(trainSet(neighbours,activeItem)));
      end                              % PD : formula in notes : take mean average

      sumAbsErr = sumAbsErr+abs(predictRating-activeRating);
      % PD : can work out error in many ways : meanAbsError here
      totalTrans = totalTrans + 1;
      % PD : get mean abs error by dividing sumAbsErr by totalTrans once loop is done
      % PD : seems to converge on 0.75

      if (mod(totalTrans,10 ) == 0)
         fprintf('\t %e \t %d\n', sumAbsErr/totalTrans, totalTrans);
      end

   end

end

MAE = sumAbsErr/totalTrans;
fprintf('\nFINAL RESULTS : \n');
fprintf('\tMAE \t\t= %e\n', MAE);
fprintf('where \tk \t\t= %d \n\tactiveUser \t= %d \n\tactiveItem \t= %d \n\tactiveRating \t= %d \n',
   k, activeUser, activeItem, activeRating);


function sim = computeSimilarities(user,trainSet)

user_row = trainSet(user,:);
sim = trainSet*user_row';
```

Output : Console – cfsimple.m

Using small sample file for testing, having reduced contents columns to the three required : epinions_rating_3_short2.txt

```
TO DATE :
         MAE             Total Translactions
        ------------------------------------------------------
         1.400000e+00    10
         1.500000e+00    20
         1.500000e+00    30
         1.550000e+00    40
         1.520000e+00    50
         1.533333e+00    60
         1.514286e+00    70
         1.537500e+00    80
         1.466667e+00    90
         1.500000e+00    100
         1.454545e+00    110
         1.458333e+00    120
         1.430769e+00    130
         1.428571e+00    140
```

```
         1.440000e+00    150
         1.443750e+00    160
         1.417647e+00    170
         1.416667e+00    180
         1.426316e+00    190
         1.435000e+00    200

FINAL RESULTS :
        MAE             = 1.435000e+00
where   k               = 10
        activeUser      = 2
        activeItem      = 23
        activeRating    = 1
```

- Devise and evaluate an algorithm that combines ratings data and trust data to make a recommendation.

  Epinions data is provided (... other data can be downloaded from the web).
- Compare your algorithm with and without the trust data?
- Does the trust data data improve the recommendation?

DEVISE ALGORITHM

EVALUATE ALGORITHM

COMPARE ALGORITHM WITH TRUST DATE

COMPARE ALGORITHM WITHOUT TRUST DATE

GOOD OR BAD?