CSEP546 - HW2 Answers

Name: Abhijith Jagannath
Student ID: 1776317

Autumn 2017
11 / 05 / 2017

1.1. Code description:

- Language used: python3
- Usage :
  - $ python3 main.py TrainingSet.txt TestingSet.txt
- Main.py :

  main()

  Reads and loads both training and test files
  It also measures each sub functions and timings
  We call the `CF.train` from here which loads the data

- collaborativeFiltering.py

  `organize_data`

  We copy the data into 2 dicts.

  1. Per user data dictionary
     a. `user_votes[user]['items'][item] = vote`
     b. `user_votes[user]['sum_votes'] += vote`
     c. `user_votes[user]['num_votes'] += 1`
  2. Per item data dictionary
     a. `item_users[item]['users'].append(user)`
     b. `item_users[item]['sum_votes'] += vote`
     c. `item_users[item]['num_votes'] += 1`

  `predicted_vote`

  Here we calculate the mean vote per user. This also calls the correlation function which calculates the correlation

  `correlation`

  This calculates the correlation between users

  output.csv has all the predicted values

1.2. Result:

Mean Absolute Error: 0.695
Root Mean Squared Error: 0.892

1.3 Trying to improve accuracies:

a. Tried not using users who's votes are not usable (users who always vote 1 or 5)
   This improved RMSE to 0.887
b. Using average ratings of an item from all users instead of just mean_user_vote when correlation is 0.
   This did not improve anything!

1.4 Shortcomings of the collaborative filtering:

a. Scalability : It's just not feasible implement the algorithm as is on a real world data set. To make it more practical, we will have to implement some kind of attribute selection which may affect the accuracy.

b. System most likely not perform well with the new users a.k.a cold start problem. Because, in order to predict something meaningful system needs to know in prior, the preferences of user at hand. Some kind of survey to start with may help with this problem but not completely curable.

c. Odd-user : If a user ratings are very random and erratic and he/she has rated a considerable number of items, the information will not be useful. More harmful when there are more such users. Hence, we need come up with some filtering techniques to take these odd-ones out.

d. Susceptible to attacks. A product can be made popular overnight just by creating lot of accounts and rating the product highly and the competator's product lowly. So, there needs to be some kind of protection and filtering about who gets the account and rate.

1.5: Extra Credit:

Added some ratings to training set.

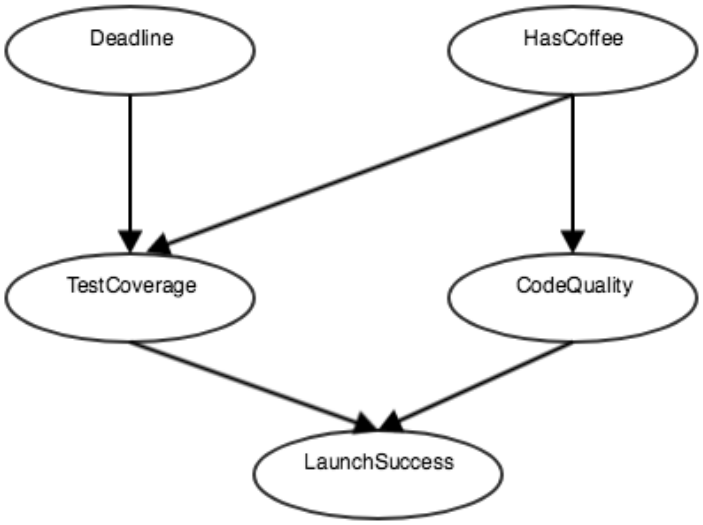
Method `def predicted_movies(user) -> does the predictions`

Use `$python3 predict_my_ratings.py UpdatedTrainingSet.txt` to get the results

`my_suggestions.json` is the file which has predictions

2.1

I. Bayesian network



II. Intern writing tests will depend on the HasCoffee and/or Deadline. If we want to naively use the Laplace estimate, there is a 50% chance that there was no coffee if intern did not write test. Hence code qaulity is affected too.

III. CodeQuality and Deadline are linked via TestCoverage, If we assume there was enough coffee, then it now depends on LauchSuccess. If TestCoverage was less and LauchSucess was in fact failure, then CodeQuality is likely to be bad.

IV. Given LauchSuccess is successful, then Deadline and CodeQualiy are (conditionally) independent. LaunchSuccess indicates there was sufficient TestCoverage AND better CodeQuality. TightDeadline may have reduced the TestCoverage but not very much. Also, all that coffee helped to pull the night outs.

V. CodeQuality and Deadline do not become independent with given HasCofee. Because HasCoffee affects the outcome of CodeQuality and TestCoverage direclty. So, given HasCoffee do not tell anything about Deadline. It all depends on LauchSuccess. Given no-cofee, but LauchSuccess is in fact successful, then it's more likely that the deadline was very very relaxed. I.e CodeQuality and Deadline are still dependent.

2.2

I. P(d) = 0.5644

$$= \sum_a \sum_b \sum_c P(a,b,c,d)$$

$$= \sum_a \sum_b \sum_c P(d \mid c) * P(c \mid b) * P(b \mid a) * P(a)$$

II. 7 Sums.
Yes we can do better by ordering the terms inside sums.

III. Best case we will need : $(n-1) * (n-1)$ sums

IV. Worst case we need $2^n - 1$ sums

2.3

a	b	c	d
0	1	1	1
1	1	0	0
1	0	0	0
1	0	1	1
1	**?**	0	1

Initialization:

$P(a) = 4/5$

$P(b \mid a) = 1/3 \qquad P(b \mid \neg a) = 1$

$P(c \mid b) = 1/2 \qquad P(c \mid \neg b) = 1/2$

$P(d \mid c) = 1 \qquad P(d \mid \neg c) = 1/3$

To calculate $P(\neg c \mid a)$

$$
\begin{aligned}
P(\neg c \mid a) &= \sum_{b} P(\neg c\ ,\ b \mid a\ ) \\
&= P(\neg c\ ,\ b \mid a\ ) + P(\neg c\ ,\ \neg b \mid a\ ) \\
&= P(\neg c\ \mid b\ ) * P(b \mid a\ ) + P(\neg c\ \mid \neg b\ ) * P(\neg b \mid a\ ) \\
&= 1/2 * 1/3 + 1/2 * 2/3 \\
&= 1/2
\end{aligned}
$$

E - Step:

$$
\begin{aligned}
P(b =?) &= P(b \mid a, \neg c, d) = P(b \mid a, \neg c) \\
&= P(a, b, \neg c) \ /\ P(a, \neg c) \\
&= P(\neg c \mid b) * P(b \mid a) \ /\ P(\neg c \mid a) \\
&= 1/2 * 1/3 \ /\ 1/2 \\
&= 1/3
\end{aligned}
$$

M- Step:

$P(a) = 4/5$

$P(b \mid a) = 1/3 \qquad P(b \mid \neg a) = 1$

$P(c \mid b) = 3/7 \qquad P(c \mid \neg b) = 3/8$

$P(d \mid c) = 1 \qquad P(d \mid \neg c) = 1/3$