Sean Ventrella
CSE 446/ Domingos/ Assignment #2

Problem 1: Collaborative Filtering on Netflix Ratings

Code Description:

- My code contains several Classes for storing data. Movie.java is for information read in from movie_titles.txt (currently not being used). Rating.java holds all the the data stored in the actual rating datasets. User.java is to store the userId, mean for that user, and a Map from movie id to the rating the user gave.

  The first step in my code is to parse the ratings data for both training and test using the function “parseRatings” which accepts the map to put the ratings into and the filename. Once we have our data parsed I pre-compute the mean values for each user in the training set by calling “calculateMean” which accepts a map of ratings and returns the mean rating.

  Finally we get to the actual prediction computational work. For each user in the test ratings map, I loop over each movie id in the ratings map for that user. Then I calculate the error (difference between actual and predicted) by calling a function called “calculateWeightedSum”, which returns the predicted rating that I then subtract from the actual and store appropriately to be used for the MAE and RMS computations. The function “calculatedWeightedSum” accepts the training data map, the test User Object, and the movie id as parameters. It then loops over each user in the training data, and if they have a rating for the movie id we call “calculateWeight” to retrieve the weight value and perform the necessary math to compute the prediction. The normalization factor (k) is 1/sum(abs(mean)) for the movie we are looking at. The function “calculateWeight” calculates the Pearson Coefficient for a given test user and training user pair and returns the weight value.

  After all predictions are made, the program outputs to the console the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMS) and then exits.

  Usage Notes
  To run the program simply run the Collaborative.java file. Be aware that since the dataset is so large, the runtime takes roughly around 5-6 hours to complete.

Accuracy:

- Mean Absolute Error = 0.69
- Root Mean Squared Error = 0.88

Accuracy Explanation:

- Both errors appear to be quite low so there does not appear to be any failures in the implementation of the collaborative filtering algorithm.

Shortcomings:

- One shortcoming is the runtime of this algorithm. Since it is an O(n^3) complexity after pre-computing the means, a large dataset such as ours requires an extreme runtime delay before the results complete. To fix this we can notice that the algorithms involved are food candidates for parallelizing of the code, which should be able to reduce the runtime significantly.
- Another shortcoming with the overall algorithm is that it has trouble handling predictions when the data is sparse. In a case where the test data computes a weight of zero or simply their is no data besides the user being predicted for, the value predicted becomes just the mean for the active user. One suggestion would be to provide a default cluster of recommendations for new users or require users to fill out additional ratings (that hopefully vary between likes and dislikes) so that we can better approximate future predictions.

Problem 2: Bayesian Networks

2.1.1)

Let: F = Focus

$P(\text{Olinguito} \mid \text{Furry}) = P(\text{Furry} \mid \text{Olinguito}) * P(\text{Olinguito}) / P(\text{Furry})$

$$\begin{aligned} P(\text{Furry} \mid \text{Olinguito}) &= P(\text{Furry} \mid \text{Olinguito} \wedge {\sim}F)*P({\sim}F) \\ &\quad + P(\text{Furry} \mid \text{Olinguito} \wedge F)*P(F) \\ &= (0.9)(0.4) + (1)(0.6) = 0.96 \end{aligned}$$

$$\begin{aligned} P(\text{Furry} \mid \text{Olingo}) &= P(\text{Furry} \mid \text{Olingo} \wedge {\sim}F)*P({\sim}F) \\ &\quad + P(\text{Furry} \mid \text{Olingo} \wedge F)*P(F) \\ &= (0.9)(0.4) + (0)(.6) = 0.36 \end{aligned}$$

$$\begin{aligned} P(\text{Furry}) &= P(\text{Furry} \mid \text{Olinguito})*P(\text{Olinguito}) + P(\text{Furry} \mid \text{Olingo})*P(\text{Olingo}) \\ &= (0.96)(0.1) + (0.36)(.9) = 0.42 \end{aligned}$$

$P(\text{Olinguito} \mid \text{Furry}) = (0.96)(0.1) / (0.42) = .22857 = 22.86\%$

2.1.2)

$P(\text{Olinguito} \mid \text{Furry}_1 , \text{Furry}_2 , \text{Furry}_3 , \text{Furry}_4)$

$$= \frac{P(\text{Furry}_1 , \text{Furry}_2 , \text{Furry}_3 , \text{Furry}_4 \mid \text{Olinguito}) * P(\text{Olinguito})}{P(\text{Furry}_1 , \text{Furry}_2 , \text{Furry}_3 , \text{Furry}_4)}$$

$P(\text{Olinguito} \mid \text{Furry}_1 , \text{Furry}_2 , \text{Furry}_3 , \text{Furry}_4) = (0.85) * (0.1) / (0.1) = .85 = 85\%$

2.2)

- No, D is conditionally independent of E given C.
- No, A is conditionally dependent of B given C because knowing either A or B immediately decreases the likelihood of the other.
- Yes because C is already true, so B being true no longer adds anything to the likelihood of E occurring. B and E are no longer d-connected.

- No, A is conditionally dependent of B given D because knowing D increases the likelihood of C and that in turn increases the likelihood of A and B.
- No, there exists a path through C from E to D so it is connected.

2.3)

Initialization:

$P(A) = 0.66$, $P(B | A) = 0.33$, $P(B | \sim A) = 1$, $P(C | B) = 0.66$
$P(C | \sim B) = 0.5$, $P(D | B) = 0.33$, $P(D | \sim B) = 0.5$

E-Step:

$P(? = 1) = P(B | A, \sim C, D) = P(A, B, \sim C, D) / P(A, \sim C, D)$

$$= \frac{P(A) * P(B|A) * P(\sim C|B) * P(D|B)}{P(A,B,\sim C,D) + P(A,\sim B,\sim C,D)}$$

$= 0.15$

M-Step:

$P(A) = 0.66$, $P(B | A) = 0.2875$, $P(B | \sim A) = 1$, $P(C | B) = 0.63$
$P(C | \sim B) = 0.35$, $P(D | B) = 0.63$, $P(D | \sim B) = 0.35$