CSE P5211 Applied Algorithms (Spring, 2017)
Jorge Eugenio Aguirre Gonzalez (UW StudentId 1572586, NetId jorgeag, jorgeag@uw.edu)

CSEP 546: Data Mining (Spring 2017)

Assignment 2: Collaborative filtering and Bayesian Networks

https://github.com/jogonzal/MachineLearningHw2

Problem 1: Collaborative filtering on Netflix ratings

1.0: High level description of code

*The code can be run with visual studio community 2015 – it is a normal C# console app. The only requirement is to *update the path of the training and test data files in Program.cs, or put the training data and test data files in the root directory of the repository (the console agent gives proper error messages when it can't find these files)*. I use the the CsvHelper Nuget package, which will download automatically at compile time.

The code was written in C# - here is a high-level overview: The "Main" method calls into utility classes to parse the training data and test data. It initializes a "UserCache" which internally manages all the movies and ratings with dictionaries for efficient processing. It then uses the "PearsonCoefficientCalculator" and the "MovieScorePredictor" classes to make the predictions, as well as the "MeanAbsoluteError" and the "RootMeanSquareError" classes to calculate error.

Note: To make the code efficient, I parallelized the portion that makes the predictions by user – by splitting by user, I am able to iterate the data more efficiently (making all the predictions needed for a user at once).

```
static void Main(string[] args)
{
        Console.WriteLine("Parsing the input files...");
        UserCache trainingSetCache = UserCache.BuildUserCache(UserRatingsTrainingPath);
        UserCache testingSetCache = UserCache.BuildUserCache(UserRatingsTestPath);

        Console.WriteLine("Initializing predictors...");
        PearsonCoefficientCalculator pearsonCalculator = new PearsonCoefficientCalculator(trainingSetCache);
        MovieScorePredictor predictor = new MovieScorePredictor(pearsonCalculator);

        // Get the list of users to make predictions on
        IReadOnlyDictionary<int, UserCache.UserRatingsCache> listOfUsersToPredictScoresOn =
testingSetCache.GetAllUsersAndMovieRatings();

        Console.WriteLine("Making predictions...");
        // Predict ratings for all the users in parallel
        List<MoviePrediction> predictions = listOfUsersToPredictScoresOn.AsParallel().Select(l =>
        {
                var returnValue = predictor.PredictAllScores(l.Key, l.Value.GetMovieRatings());
                return returnValue;
        }).SelectMany(s => s.Values).ToList();

        Console.WriteLine("Calculating errors...");
        var rootMeanSquareError = RootMeanSquareError.Calculate(predictions);
        var meanAbsoluteError = MeanAbsoluteError.Calculate(predictions);

        Console.WriteLine("========================================");
        Console.WriteLine("Root mean square error: {0}", rootMeanSquareError);
        Console.WriteLine("Mean absolute error: {0}", meanAbsoluteError);
}
```

1.1: Accuracies obtained

After I used only the top 10 nearest neighbors 9instead of using the entire dataset), here are the numbers I obtained (the algorithm took 20.5 minutes to run.

Root mean square error: 0.893080506610725

Mean absolute error: 0.693885980135035

Extra credit

I went ahead and added my ratings – and modified the algorithm (see function "MakeMovieRecommendationsForUser999999InTrainingSet"). To run this, simply add these recommendations (with user 999999) to the training set.

```
17622, 999999, 5.0
10362, 999999, 5.0
5448, 999999, 5.0
14, 999999, 4.0
21, 999999, 1.0
69, 999999, 2.0
68, 999999, 4.0
209, 999999, 1.0
215, 999999, 5.0
731, 999999, 1.0
6205, 999999, 5.0
12293, 999999, 5.0
12311, 999999, 1.0
12954, 999999, 1.0
11726, 999999, 5.0
13501, 999999, 5.0
13595, 999999, 1.0
13617, 999999, 1.0
13638, 999999, 2.0
7624, 999999, 4.0
```

It's worth noting that I entered the movie "Scarface" of the criminal genre, as well as "That's 70's show" of the comedy genre, and my algorithm recommended "The Godfather", as well as friends, which are definitely similar to those, among other criminal/comedy movies. I will be checking out some of those after writing the report.

```
0: Id:'14283', The Best of Friends: Vol. 3
1: Id:'1256', The Best of Friends: Vol. 4
2: Id:'10947', The Incredibles
3: Id:'5760', The Sopranos: Season 3
4: Id:'14648', Finding Nemo (Full-screen)
5: Id:'16147', The Sopranos: Season 1
6: Id:'634', Christmas with The Simpsons
7: Id:'1744', Beverly Hills Cop
8: Id:'3290', The Godfather
9: Id:'8915', Terminator 2: Extreme Edition
```

What are the shortcomings of collaborative filtering and how might you augment it?

I think the biggest shortcoming is its latency and cost at query time – to be able to answer a single query (predict a score for a user and a movie) we need to iterate through a big number of users, make similarity calculations, then calculate the predicted score. This can be mitigated by setting up a few "indexes" we could set up before running the algorithm, but runtime will still be significant.

Another disadvantage is that for users that have always rated everything with the same "5 star" rating, the vanilla version of the algorithm will have a hard time differentiating them – as the difference between the average rating and the rating of all movies for that user is 0. In order to make a recommendation for such users, one suggestion I have for augmenting the algorithm is taking into account not just the predicted score (which in the case of a user that predicted everything with 5 stars, will be 5 stars for all movies), but also the "trust" that we have in that score. We will trust the score more when there are other users that share scores to the movie, even if the score is the same as the average score. This can be done in a similar way we calculate the weight, but with a calculation that doesn't involve subtracting the average and the current rating and instead considers that if you rated everything 5 stars, you liked them.

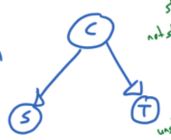
CSE P5211 Applied Algorithms (Spring, 2017)
Jorge Eugenio Aguirre Gonzalez (UW StudentId 1572586, NetId jorgeag, jorgeag@uw.edu)

Problem 2: Bayesian Networks

2.1 Car problem

2.1.a "You do a quick test drive and discover the car is unstable at turns but is not skidding. How likely is it to be tire or suspension problem?"

C==true ∴ Tires
C==false ∴ Suspension
Skidding: S
Unstable: T



		tires	suspension
		C	$\bar{C}$
skidding	S	0.5	0.2
not skidding	$\bar{S}$	0.5	0.8

		tires	suspension
		C	$\bar{C}$
unstable	T	0.3	0.7
not unstable	$\bar{T}$	0.7	0.3

$P(C)=0.3$
$P(\bar{C})=0.7$
$P(S|C)=0.5$
$P(\bar{S}|C)=0.5$
$P(S|\bar{C})=0.2$
$P(\bar{S}|\bar{C})=0.8$
$P(T|C)=0.3$
$P(\bar{T}|C)=0.7$
$P(T|\bar{C})=0.7$
$P(\bar{T}|\bar{C})=0.3$

$$P(C|\bar{S},T)=\frac{P(C\cdot\bar{S}\cdot T)}{P(\bar{S}\cdot T)}=\frac{P(C)\cdot P(\bar{S}|C)\cdot P(T|C)}{P(C\cdot\bar{S}\cdot T)+P(\bar{C}\cdot\bar{S}\cdot T)}=\frac{0.3\cdot 0.5\cdot 0.3}{0.3\cdot 0.5\cdot 0.3+P(\bar{C})\cdot P(\bar{S}|\bar{C})\cdot P(T|\bar{C})}$$

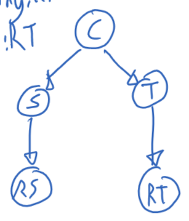
$$=\frac{0.3\cdot 0.5\cdot 0.3}{0.3\cdot 0.5\cdot 0.3+0.7\cdot 0.8\cdot 0.7}=\frac{.045}{.045+.392}=0.103$$

This means that the probability of this being a tire problem is 0.103 and the probability of this being a suspension problem is 0.897.

2.1.b What are the probabilities of the car condition given that the customer now claims that his car is skidding but otherwise stable?

(Adding on to the tree and tables above)

Reported skidding: RS
Reported unstable: RT



$P(RS|S)=0.75$
$P(RS|\bar{S})=0.25$
$P(\overline{RS}|S)=0.25$
$P(\overline{RS}|\bar{S})=0.75$
$P(RT|T)=0.75$
$P(RT|\bar{T})=0.25$
$P(\overline{RT}|T)=0.25$
$P(\overline{RT}|\bar{T})=0.75$

$$P(C|RS,\overline{RT})=\frac{P(C\cdot RS\cdot\overline{RT})}{P(RS\cdot\overline{RT})}=\frac{P(C)\cdot P(RS|C)\cdot P(\overline{RT}|C)}{P(C\cdot RS\cdot\overline{RT})+P(\bar{C}\cdot RS\cdot\overline{RT})}=\frac{P(C)\cdot P(RS|C)\cdot P(\overline{RT}|C)}{P(C)\cdot P(RS|C)\cdot P(\overline{RT}|C)+P(\bar{C})\cdot P(RS|\bar{C})\cdot P(\overline{RT}|\bar{C})}$$

$P(RS|C)=P(RS\cdot S|C)+P(RS\cdot\bar{S}|C)=P(S|C)P(RS|S)+P(\bar{S}|C)\cdot P(RS|\bar{S})=0.5\cdot 0.75+0.5\cdot 0.25=0.5$

$P(\overline{RT}|C)=P(\overline{RT}\cdot T|C)+P(\overline{RT}\cdot\bar{T}|C)=P(T|C)\cdot P(\overline{RT}|T)+P(\bar{T}|C)\cdot P(\overline{RT}|\bar{T})=0.3\cdot 0.25+0.7\cdot 0.75=0.6$

$P(RS|\bar{C})=P(RS\cdot S|\bar{C})+P(RS\cdot\bar{S}|\bar{C})=P(S|\bar{C})\cdot P(RS|S)+P(\bar{S}|\bar{C})\cdot P(RS|\bar{S})=0.2\cdot 0.75+0.8\cdot 0.25=0.35$

$P(\overline{RT}|\bar{C})=P(\overline{RT}\cdot T|\bar{C})+P(\overline{RT}\,\bar{T}|\bar{C})=P(T|\bar{C})\cdot P(\overline{RT}|T)+P(\bar{T}|\bar{C})\cdot P(\overline{RT}|\bar{T})=0.7\cdot 0.25+0.3\cdot 0.75=0.4$

$$P(C|RS,\overline{RT})=\frac{0.3\cdot 0.5\cdot 0.6}{0.3\cdot 0.5\cdot 0.6+0.7\cdot 0.35\cdot 0.4}=\frac{0.09}{0.09+0.098}=0.479$$

This means that the probability of this being a tire problem is 0.479 and the probability of this being a suspension problem is 0.521.

2.2 Evaluate statements given Bayes graph

1. P(F | E, D, A) = P(F | E)

This is **true** because "each node is conditionally independent of its nondescendants, given its parents". D and A are nondescendants of F, and E is given, which means the statement above is true.

2. P(G, H | A, C) = P(H | A, C) ∗ P(G | A, C)

This is trying to make the assumption that G and H are conditionally independent given A and C. Given the graph, this is **false**, because A and C still leaves B (among other nodes) making the conditional independence false.

3. P(B | A = a, C = c) = P(B)

This is **false**. Knowing the values of A and C does impact our knowledge on B.

4. P(H, I | E, F) = P(H | E, F) ∗ P(I | E, F)

This is assuming conditional independence of H and I. This is **true,** because "each node is conditionally independent of its nondescendants, given its parents"- in this case, all the parents are given.

5. Is F independent of H given B and G?

This is **false** because not all parents of F are given – knowing H would give us more information about F.

2.3 EM algorithm

Initial values are:

R	A	T
1	1	0
1	1	1
1	1	1
0	1	1
0	1	?
0	0	0
0	0	?

E step 1:

P(T | ¬R, A) = 1/1 = 1 (only one sample)
P(T | ¬R, ¬A) = 0/1 = 0 (only one sample)

M step 1:

R	A	T
1	1	0
1	1	1
1	1	1
0	1	1
0	1	1
0	0	0
0	0	0

E step 2

P(T | ¬R, A) = 2/2 = 1 (from two samples)
P(T | ¬R, ¬A) = 0/2 = 0 (from two samples)

The probabilities are stabilized and we're done.