Homework 11: Symposium on Recommendation Systems

36-462/662, Spring 2020

Due at 10 pm on 9 April 2020

AGENDA: Consolidating ideas on recommendation engines.

The following problems all concern making a recommendation engine for wine. They are all hypothetical, but inspired by real issues that arose when some acquaintances of mine tried to build such a thing about ten years ago. (The venture never launched for reasons unrelated to its technical or commercial feasibility.) No knowledge of wine is required for solving these problems.

1. *Online questions* (10) are online.

2. *The wine club* Our recommendation system begins when a club of wine ~~snobs~~ enthusiasts gathers data on the wines its members like. They use the **box-of-wines** representation, so the data consists of a list of wine types for each member, like so:

```
Aligheri_D.: "L'Inferno 00", "Beata Beatrix 83", "Averno 01", "Ad Astra 02"
Maro_P.V,: "L'Inferno 00", "Averno 01", "Ad Astra 02", "Maecenas Estates n.d."
Montresor_E.: "Valdemar Revenant 03", "L'Inferno 00"
Fortunato_A.: "Pym Amontillado 02", "Ligeia 99"
Rumpole_J.: "Chateau Plonk 97", "Chateau Plonk 98", "Chateau Plonk 99",
           "Chateau Plonk 00", "Chateau Plonk 01", "Chateau Plonk 02"
```

   There will, however, in general be more than five members.

   The club wants to use this data to build a system to recommend new wines to members, based on the ones they already like.

   (a) (4) Explain how to convert this data into a data frame with one row for each member, and a binary value for each feature. Be clear about what the features are, how they relate to wines, and how you would determine the number of features from the data.

   (b) (5) Explain how to find the k wine-drinkers most similar to a given member. Be explicit about how you would calculate similarity.

(c) (6) Describe a procedure to recommend up to m wines to a customer based on the wines enjoyed by the k most-similar people in the data base. Explain how your procedure ensures that these wines are new to the customer, and how it selects m recommendations if it comes up with more than m candidates.

(d) (4) Supposing that the system began with only the five members listed above, what wines, if any, would your procedure with $k = 2$, $m = 2$ recommend to E. Montressor? Does this seem like it might be a problem?

(e) (4) Suppose that the system began with only the five members listed above, what wines, if any, would your procedure with $k = 2$, $m = 2$ recommend to J. Rumpole? Does this seem like it might be a problem?

(f) (4) How did Shardananad and Maes, in the "Social Information Filtering" paper, solve the "cold start" problem of making music recommendations to new users? What would be the analogous way of making recommendations to new users of the wine recommendation engine?

(g) (6) Explain how you would use cross-validation to evaluate how well the recommendation system was working. Be as specific as possible.

3. *Yuck*

(a) (2) Explain why the representation in the previous problem doesn't distinguish between someone who hasn't tried a wine, and someone who has tried a wine and didn't like it.

(b) (4) Explain how to modify the representation so that it distinguishes between (i) wines a user has tried and liked, (ii) wines a user has tried and doesn't like, (iii) wines a user has tried and doesn't feel strongly about, and (iv) wines user has not tried.

(c) (5) How (if at all) would you have to modify your procedure to work with this new representation? Explain your answer.

(d) (4) Explain how the new representation makes it easier for users to give feedback to the system about whether its predictions were accurate.

4. *Scaling up* The Pennsylvania Liquor Control Board (which runs all the state wine and spirits stores) sells about 18,000 distinct kinds of wine at any one time. Suppose it was interested in using the wine recommendation system originally developed by the club. Very few users will have ratings for even 1% of the available wines.

(a) (3) Explain how having 18,000 features would increase the amount of computational work needed to make recommendations, *and* not add a lot of information to the recommendations.

(b) (5) Suppose someone from the PLCB, who has just read read `https://www.wired.com/story/the-style-maven-astrophysicists-of-silicon-valley/`, suggests using "eigenvector decomposition, a concept from quantum mechanics, to tease apart the overlapping 'notes' in an individuals" wine preferences. What might that actually mean here? What would be a problem in doing so? *Hint:* lots of data are missing.

(c) (5) Explain how to use matrix factorization to come up with recommendations. How (if at all) does it get around the problem of having so many features?

(d) (3) The PLCB currently tracks four pieces of information about each wine: the year of origin (e.g., 2017), the country of origin (e.g., USA), the region within the country (e.g., the Willamette Valley in Oregon), and the variety or style of wine (e.g., Reiseling). Explain how this could be used to make a content-based recommendation system. *Hint:* How would you use this to find similar wines?

(e) (3) How (if at all) would your matrix-factorization system make recommendations for a new wine? Could the content-based system help out here? Could the Sahrdanand and Maes approach work?

(f) (3) How (if at all) would your matrix-factorization system make recommendations for a new user? Could the content-based system help out here? Could the Shardanand and Maes approach work?

5. *Incentives* The wine recommendation system now partners with a wine seller. Suppose that the system estimates the probability that a user i will like wine k is p_{ik}, and that the probability of a user buying a recommended wine (through the partner) is cp_{ik}, for some $c < 1$. The price of a bottle of wine k is r_k dollars. The wine seller pays the recommendation system a fixed fraction of the price on all sales it generates, say qr_k dollars on a sale of r_k dollars.

   (a) (2) Find an expression for the recommendation engine's expected income from recommending wine k to user i, in terms of p_{ik}, c, q and r_k.

   (b) (5) Consider two wines, k and l. Suppose $p_{ik} < p_{il}$, i.e., the system predicts user i is less likely to enjoy wine k than wine l. When will those running the system nonetheless prefer to recommend wine k? Express your answer algebraically, and simplify as much as possible.

   (c) (3) More realistically, the probability that a user buys a recommend wine will not just be a constant c times the probability of the user liking the wine p_{ik}; rather, c should be replaced by a function of the price r and the user's income y, say $d(r, y)$. Suppose that the system knows this function and the income of each user. Now, when will the system recommend wine k rather than wine l to user i, even though $p_{ik} < p_{il}$?

RUBRIC (10): The text is laid out cleanly, with clear divisions between problems and sub-problems. The writing itself is well-organized, free of grammatical and other mechanical errors, and easy to follow. All plots and tables are generated using code embedded in the document and automatically re-calculated from the data. Plots are carefully labeled, with informative and legible titles, axis labels, and (if called for) sub-titles and legends; they are placed near the text of the corresponding problem. All quantitative and mathematical claims are supported by appropriate derivations, included in the text, or calculations in code. Numerical results are reported to appropriate precision. Code is properly integrated with a tool like R Markdown or knitr, and both the knitted file and the source file are submitted. The code is indented, commented, and uses meaningful names. All code is relevant; there are no dangling or useless commands. All parts of all problems are answered with actual coherent sentences, and raw computer code or output are only shown when explicitly asked for.