Homework 11: Showtime!

36-462/662, Spring 2022

Due Thursday, 21 April 2022 at 6 pm

Agenda: Implementing matrix factorization, and making movie recommendations.

Remember[1] that in **matrix factorization**, we have an $n \times p$ matrix $\mathbf{x}$, and we want to approximate it as the product of two matrices,

$$\mathbf{x} \approx \mathbf{f}\mathbf{g}^T$$

where $\mathbf{f}$ is $n \times q$ and $\mathbf{g}$ is $p \times q$, and $q \ll n$, $q \ll p$. This is "factorization" because we have (approximately) "factored" $\mathbf{x}$ into $\mathbf{f}$ and $\mathbf{g}$. Here q is the number of factor variables.

In recommendation systems, $\mathbf{x}$ is the matrix of ratings, with n users and p items. If every user had rated every item, we could do matrix factorization using PCA. Since most users have not rated most items, straightforward PCA won't work so well. One approach is to start with guesses about $\mathbf{f}$ and $\mathbf{g}$, and iteratively improve: holding $\mathbf{f}$ fixed, adjust $\mathbf{g}$ to fit the ratings we do have, and then, holding $\mathbf{g}$ fixed, optimize $\mathbf{f}$.

In a little more detail, say that the rating of user i for item j is x_{ij}. Then we want

$$x_{ij} \approx \sum_{k=1}^{q} f_{ik} g_{jk} \ .$$

Mean squared error tells us how well the combination of user scores $\mathbf{f}$ and item scores $\mathbf{g}$ fit the ratings $\mathbf{x}$:

$$MSE = \frac{1}{|C|} \sum_{(i,j) \in C} \left(x_{ij} - \sum_{k=1}^{q} f_{ik} g_{jk} \right)^2 ,$$

C being the set of pairs where user i has rated item j, and $|C|$ the number of ratings. We can re-write this as

$$MSE = \frac{1}{|C|} \sum_{i=1}^{n} \sum_{j \in I(i)} \left(x_{ij} - \sum_{k=1}^{q} f_{ik} g_{jk} \right)^2 \text{ or as} \quad (1)$$

$$= \frac{1}{|C|} \sum_{j=1}^{p} \sum_{i \in U(j)} \left(x_{ij} - \sum_{k=1}^{q} f_{ik} g_{jk} \right)^2 , \quad (2)$$

where $I(i)$ are the items which user i has rated, and $U(j)$ are the users who have rated item j.

In **alternating least squares**, we first hold $\mathbf{g}$ fixed, and minimize equation (1) to find $\mathbf{f}$. Then we hold that new $\mathbf{f}$ fixed, and minimize equation (2) to find $\mathbf{g}$. Then we iterate to convergence.

There is one last bit of math. Notice that if $\mathbf{d}$ is a diagonal $q \times q$ matrix, then

$$\mathbf{f}\mathbf{g}^T = \mathbf{f}\mathbf{d}\mathbf{d}^{-1}\mathbf{g}^T = \mathbf{f}\mathbf{d}(\mathbf{g}\mathbf{d}^{-1})^T \ ,$$

so matrix factorizations aren't unique. (Find one, and we can find another just as good by making compensating changes to the user and the item columns.) We can fix this by insisting that every column of $\mathbf{g}$ have norm 1.

[1] In the notes, I wrote this as $\mathbf{x} \approx \mathbf{f}\mathbf{g}$ and said $\mathbf{g}$ should be $q \times p$. This is just a notational difference, but the way I'm writing it here is a bit more common when people use matrix factorization for recommendation systems, whereas the way I wrote it in the notes is more common in factor analysis in the social sciences.

The file `hw-11.R` on the class website contains all the *provided* code for this assignment. (You will have to write more code of your own.) Questions below refer to functions in this code by name. The names are deliberately meaningless (which you should not imitate in your own code).

Some of the calculations in Q6 take multiple minutes on my computer. Use caching!

---

1. **Reading questions** are from Kearns and Roth's *The Ethical Algorithm* (=TEA), chapter 2 (available electronically through the library, if you haven't bought a copy already). This looks ahead to HW 12.
   a. (2) On p. 68, TEA says that "it has become virtually impossible to enforce notions of fairness that work by trying to restrict the inputs given to a machine learning or algorithmic decision-making process". What are two specific reasons the book gives to justify this assertion?
   b. (1) On p. 69, TEA explains a notion of fairness they call "statistical parity". Define "statistical parity" in your own words.
   c. (1) On pp. 69–71, TEA lays out several objections to statistical parity. Explain the objection the book regards as most serious.
   d. (2) On p. 75, TEA claims that "even... the most accurate model may badly violate fairness". What is the reason given for this (in your own words)?
   e. (1) On p. 77, TEA considers using different models for different groups. It gives one legal/ethical objection to doing so, and one technical objection. Explain them both.
   f. (1) What is being shown in the figure on p. 80? Specifically, what are the axes, what are the dots, and what is the curve?
   g. (1) Why, in the figures on p. 83, are there no points above and to the right of the curves, as there are in the figure on p. 80?
   h. (1) On pp. 81–82, TEA describes how an algorithm for optimizing a *single* criterion (like accuracy) can be used to map the trade-off between minimizing *two* distinct criteria (like accuracy and fairness). Explain the procedure, in your own words. We have seen a similar, if not identical, idea earlier in the course — what is it?
2. **Understanding the Math**
   a. (3) Explain how the following statement relates to equation (1) above: "Holding the item scores constant, we can find each user's scores on all factors by linearly regressing that user's ratings on the item scores." Why should the intercept in this regression be 0? How many items must each user have rated, in terms of the number of factors q?
   b. (3) Explain how the following statement relates to equation (2) above: "Holding the user scores constant, we can find each item's scores on all factors by linearly regressing that item's ratings on the user scores." Should the intercept in this regression also be 0? How many users must have rated each item, in terms of q?
3. **Linking Code and Math** Explain your answers, don't just assert.
   a. (3) What part of the math does the function `foo()` implement? That is, what is `foo()` trying to do, how does it try to do it, and how can you tell?
   b. (3) What part of the math does `bar()` implement? What is the point of part of the code between the call to `apply()` and the final `return()`?
   c. (1) How do the `foo()` and `bar()` functions handle the possibility that some users didn't rate some items, so there are `NA` entries in the ratings matrix?
   d. (4) What part of the math does `baz()` implement? Why is this an inefficient implementation when many ratings are missing? (You do not have to suggest a better one.)
   e. (4) What part of the math does `qux()` implement?
   f. (5) What part of the math does `alice()` implement? Why does it set `old.MSE` to infinity before it's calculated anything?
4. **Try It Out Where We Know the Answers**
   a. (3) What does the `babur()` function try to do, and how does it do it?
   b. (3) Run `babur()` to produce a simulated data set with 50 users, 10 items, and two factors. Run `qux()` on the rating matrix, again with 2 factors. Plot (i) the estimated user scores as a

function of the true user scores, (ii) the estimated item scores as a function of the true item scores, and (iii) the predicted ratings (found using the estimated scores) as a function of the true ratings. (You may find it helpful to convert these objects to vectors before plotting.) You should find that (iii) gives a much better match than either (i) or (ii) — why is that?

c. (3) What does the `chandra()` function do, and how does it work? Run it on the demo ratings matrix you just made in Q4b, and verify that (i) this function creates the right over-all number of NAs, (ii) it creates the right number of NAs per row, and (iii) it doesn't change any *other* entries in the matrix.

d. (3) Use the `chandra()` function to create a matrix from your earlier demo ratings matrix where two ratings per user are replaced by NAs. Call the resulting matrix `obscured`. Run `qux()` on this new matrix with NAs. Check that the new initial scores are *not* the same as the initial scores that you got on the full matrix in Q3b. Using these initial scores for items and users, calculate predicted ratings for all items and users. Make a plot of true ratings against predicted ratings, using different plotting symbols (or colors, etc.) to indicate whether the true rating was obscured or not. Do these initial estimates do a better job of matching the un-obscured ratings? How can you tell?

e. (3) Run `alice()` on the matrix `obscured` that you made in the previous problem. Using those estimated scores, calculate predicted ratings for all combinations of users and items. Plot the true ratings against the estimated ratings, as in the previous question. Do these predictions better match the un-obscured ratings than the initial predictions? Do they better match the obscured ratings than the initial predictions?

f. (5) What is the point of Q4d and Q4e? That is, what do you learn from doing them, why is that an important thing to know about the code, and how do Q4a–Q4c contribute to it?

5. **Keep Asking It Questions Whose Answers We Know**

a. (5) Write a function, `calc.rmse()`, which takes 3 arguments: (i) `sub`, a ratings matrix with some obscured entries; (ii) `super`, the ratings matrix which `sub` was derived from; and (iii) `mdl`, a model of the kind returned by `alice()`. Your function should calculate predictions for all ratings, using `mdl`, and then calculate the RMSE for the ratings which are visible in `super` but were `NA` in `sub`. Your code should allow for the possibility that `super` has some `NA` entries as well, but you can assume that every entry that is `NA` in `super` will also be an `NA` in `sub`. (That is, `sub`'s missing values are a superset of `super`'s.)

Using your `calc.rmse()` and your `obscure` matrix, calculate, and plot, the RMSE on the hidden ratings for $q \in 1:5$ (i.e., from 1 to 5 factors inclusive). Why is 2 factors the best? (If it is not the best, you have a mistake somewhere.)

b. (5) What is the point of Q5a? What do you learn from doing it, why is that an important thing to know about the code, and why do I ask you to do this *before* using the code on real data?

c. (5) Why is your `calc.rmse()` giving us an estimate of predictiion error? What kind of estimate is it (empirical risk, leave-one-out CV, k-fold CV, empirical risk plus optimism penalty, etc.)? For full credit, explain your answers, rather than just making assertions.

6. **Try It on Something Real** The `MovieLens` project was an early movie recommendation system, running during the 1997–1998 academic year. It had 943 users who gave 1–5 ratings on 1664 movies, with about 100,000 ratings in the data set. The data is part of the `recommenderlab` package in R, as `MovieLense` (sic), but in a slightly annoying format, so I have saved it as a matrix in `MovieLense.csv`. Load the data and make sure it has the right dimensions.

a. (4) As you saw in Q2a and Q2b above, matrix factorization has trouble if there are users who've rated too few items, or items rated by too few users, compared to the number of factors. Create two histograms, one of the number of ratings per user, and the other of the number of ratings per item. Report the number of users who have rated < 6 items, and the number of items rated by < 6 users. Remove those rows and columns from the data set. (If necessary, iterate.) Report the dimensions of the data set you are left with. Use this pruned data in all further problems.

b. (4) Using the `chandra()` function, create a version of the data set which hides 1 extra ratings per

user. How do you know that this has worked properly?

c. (5) Do matrix factorization for $q \in 1:5$ on the obscured data set. Use your `calc.rmse()` to calculate, and plot, the RMSE of these models. What number of factors works best?

d. (5) Repeat what you did in Q6b and Q6c nine more times, and average the RMSEs you get from each run. (Make sure that you are obscuring different ratings each time.) Plot the results. What number of factors works best here on average?

7. **Timing** (1) How long, roughly, did you spend on this problem set?

Presentation 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. 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 plots and tables are generated by code included in the R Markdown file. 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. All parts of all problems are answered with actual coherent sentences, and raw computer code or output are only shown when explicitly asked for. Text from the homework assignment, including this rubric, is included only when relevant, not blindly copied.

Extra Credit

These are all independent of each other.

a. **Heteroscedasticity** (3) When the variance isn't constant, we say that a process is "heteroscedastic" (or "heteroskedastic"), as opposed to "homoscedastic". For the final model you selected in Q6, what is the over-all MSE? Call this $\hat{\sigma}^2$. Calculate the MSE for each movie; call this $\hat{s}_j^2$ for movie j. Display a histogram of the $\hat{s}_j^2$. Does this suggest heteroscedasticity or homoscedasticity? For comparison, do a simulation where, for movie j which was rated n_j times, you generate n_j $\mathcal{N}(0, \hat{\sigma}^2)$ random variables and find their sample variance (so you end up with one simulated sample variance per movie). Explain whether this simulation is homoscedastic or heteroscedastic. Plot the histogram from the simulation and compare the distributions. Does this comparison suggest homoscedasticity or heteroscedasticity in the data?

b. **Interpretation** (5) The `recommenderlab` package contains, along with the `MovieLense` data frame of ratings, a `MovieLenseMeta` data frame of meta-data about the movies themselves, such as title, year, and genre. Using the movie scores you estimated with $q = 2$ latent factors, try to use this meta-data to interpret the factors for non-statisticians. That is, say in ordinary language how movies with high scores on each factor differ from those with low scores on that factor. (If you think it's not possible to give such an interpretation, explain why not.) *Hint*: You're trimming the data set in Q6a, so keep track of which movies you're dropping, and trim rows from the meta-data accordingly, or this will be *very* hard to do.

c. **Regularized recommendations** (5) The reason we need at least q rating per user or per item is that otherwise we are trying to estimate more coefficients than there are observations. Ordinary least squares can't do this, but methods for high-dimensional, "$p > n$" regression are perfectly happy to do so. Modify the code provided to use the lasso; include a way to pick the lasso penalty. Does it change any of the results for MovieLens? In particular, does it give a better fit on obscured movies, and/or pick a different number of factors?

d. **Fair recommendations** (5) The dataframe `MovieLenseUsers` gives information about the users who provided the ratings, including their sex and occupation. Compute the RMSE of the ratings when $q = 2$ for both sexes, and for each occupational group. Are the RMSEs equal for different groups? Is there a relationship between the size of the group and how accurate the predictions are? How big is the difference in RMSEs between the sexes, compared to the difference in RMSEs across occupations?