Homework 2 + Final Project

Let's build some recommender systems.

Overall deliverable guidelines

The main deliverable is a GitHub repository with the following:

- A README file outlining the repository contents
- A requirements file with all software/package requirements to run your code
- A top level directory for Part I
    - Include a notebook or markdown with your approach and basic results
- A top level directory for Part II
    - Include a notebook or markdown with your approach and basic results

Projects should be completed in Python (preferred) or R.

Data Scientists are more than statisticians and more than engineers. We solve real-life business problems by leveraging data and are usually brought in to brainstorm and frame a problem from day one. You will probably be expected to interface directly with other departments in the business, where your trust will be gained by sharing your insights into the core business problem at hand, and by translating these problems into a technical solution. This also means that you will be required to *communicate* your solution to key stake holders that are committed to solving the problem, but may not have your technical background.

Imagine a future employer looking at this repository. They would evaluate your project based on the solution's accuracy and your technical prowess, but they would also look for coherence and creativity. An employer would look for thoughtfulness and thoroughness; for example, how does the solution scale, were the hyper parameters tuned, will this solution discover novel recommendations, etc. Are the major contributions of your work clear? Did you call out important caveats?

These are the same standards on which you will be graded for these projects in this class.

Instructions for groups

You are organized into groups. Each team member will bring their own strengths and have their own opportunity areas for development. It's ok for teams to divide and conquer for divisible tasks, but *every team member should have a complete understanding of the entire solution, end to end*. I also recommend that before any piece solution is implemented by an individual, the entire team should brainstorm possible approaches, and should agree on the framework, what to try, and what the deliverable will look like.

Data sets

Data sets:

- Many are available:
    - https://github.com/caserec/Datasets-for-Recommneder-Systems/blob/master/README.md
- The "Full" data set for education and development (27M ratings; 265 MB) from MovieLens is recommended if there is no specific reason to choose another data set
    - https://grouplens.org/datasets/movielens/

Homework 2

Due November 7th

1. Objective

Build two very simple collaborative filtering models. You may use published packages or methods (e.g. Implicit: https://github.com/benfred/implicit) - the goal of this exercise is to gain a practical intuition for how these types of common models work, and to develop methods to test and explore them. Treat this as a case study and think about it from a business perspective. What is your objective? What are you trying to optimizing and what are you willing to sacrifice?

2. Choose a data set

See above. You may want to choose a data set in a domain aligned with your personal interests. Data sets that have additional information beyond ratings on users or items will probably prove more useful if you choose to use this data set for the final project as well.

For this exercise, develop with a small data set (e.g. < 10000 users / < 100 items). If you need to choose data from a larger dataset be thoughtful about how you sample your data.

3. Report

- Build two brute-force collaborative filtering algorithms to recommend items to users:
    1. Neighborhood-based (either item *or* user)
    2. Model-based (you can pick one, but vanilla Matrix Factorization is a good default)
- Develop evaluation methods:
    1. Cross-validation setup
    2. Accuracy on training and test data (choose a primary accuracy metric and a secondary accuracy metric)
    3. Coverage on training and test data

- Systematically try a range of hyper parameters for your models (e.g. neighborhood size or number of latent dimensions). Record *and explain* in your markdown how your evaluation metrics change as a function of these parameters. *Include plots!*
- After seeing these results, what other design choices might you consider?
- How do the evaluation metrics change as a function of your model size? Systematically sample your data from a small size to a large size
    1. Does overall accuracy change?
    2. How does run-time scale with data size?
- Referencing your case study set up from above, how do these methods meet your hypothetical objectives? Would you feel comfortable putting these solutions into production at a real company? What would be the potential watch outs?

4. **Evaluation**

[10%] Case study framework

- How well was this envisioned as an actual business problem? Is it clear what is being solved, and what acceptance criteria would be?

[30%] Technical correctness

- Is the code complete and does it run?
- Were appropriate methods applied in the right place? Were hyper-parameters used in a reasonable manner?

[10%] Evaluation methods

- Correct implementation

[25%] Model exploration

- How well was each model explored?
    - Hyper-parameters
    - Data sample size
    - (Optional) Anything else?
- Are plots clearly legible, and do they make sense?

[25%] Write up

- If I were your manager at work, or a key stake holder, and I needed to understand what you did and why you did it in order to make a business decision (e.g. "go / no-go"), is this write up enough for me to go on? Would I have any lingering questions that were not already pointed out as "potential watch outs"?
- Clarity matters
- Connection to the business problem matters
- Demonstrating an understanding of why the algorithms are performing or not, or *when* then do or do not, matters

Final project

Due December 14th

The final project should allow you to add a bullet point to your resume and add a section to your public project portfolio. The outcome of this project is intended to be hosted in a public code repository where others can see your work, and most importantly, see how you think about a business problem, design a solution, and communicate your results.

1. Choose a personal/group objective:

You will extend a recommendation system beyond the typical collaborative filtering setups that are common in coursework to something that is more typical of problems you would need to solve in industry.

You are free to propose an idea for approval, or choose one of these pre-approved options:

1. Association Rules
    1. Association Rules are an old idea for recommendation systems, but they are not common in industry because they are hard to scale. That is, until Spark ML added a pattern mining algorithm into it's library. Use Spark to build out an Association Rules-based recommendation engine. Thoroughly explore the results, and answer questions such as:
        1. What kind of patterns does the algorithm find?
        2. What kind of patterns are most useful for prediction?
        3. How sensitive are patterns to initial conditions?
        4. Can you use the found patterns for other prediction tasks?
2. Multi-entity representation in matrix factorization
    1. Matrix Factorization methods are linear in either items or users. This means, for example, that a user with preferences for two very different types of items will be described by a user vector that averages between the two item clusters; due to this cluster splitting effect, the user vector may not be close to either item cluster. Instead, the user vector may be small and near the origin, which will result in bad recommendations for the user. One potential way to solve this problem may be to allow users or items to have multiple vectors, so that a user could appear twice - once near each disparate item cluster. Define a framework to split items or users into one or more vectors and develop a system to learn and predict with this model. You should be able to answer a few important questions such as:
        1. What is the criteria for splitting a user or an item into more than one vector? Is this criteria a heuristic, or optimized?
        2. What rules are used to make a prediction for a user and/or an item when there is more than one vector?
        3. What implications does your approach have on speed and memory?
        4. What implications does your approach have on accuracy?
            1. Compared to a non-split version?
            2. On split vectors vs non-split vectors?
            3. For users or items that have eclectic membership?
3. Factorization Machines
    1. Factorization Machines have been described as state of the art for many recommendation systems. Yet, experience has shown these models to suffer from slow

    training and local minima. Use a large(ish) dataset and characterize where FMs are easy to fit and accurate and where they are not.
    1. Start with models that have no side information, and are only user and item ratings. Specifically, subsample datasets from small to large, and subsample users/items from sparsely-populated to well-populated, and train and test FMs. Where do they work the best? Where do they fail? Can you set good rules of thumbs for their training and use?
    2. Next use side information about users or items. Answer the same questions as above.
4. Approximate Nearest Neighbors
    1. Most Collaborative Filtering algorithms (both Memory and Model-based) suffer from the complexity of nearest neighbor lookup at industry-scale data. Use Spark's Locality Sensitive Hashing methods to design collaborative filtering algorithms that scale. Assuming that you are using a single machine (i.e. your laptop, not a cluster), how does the LSH approach compare to naive brute-force approaches in speed and accuracy?
5. Missing Not At Random
    1. Steck's "Missing Not At Random" paper (2010) shows that Collaborative Filtering can be vastly improved when optimized with an objective function that recognizes that missing data is likely different than rated data. Yet, there is no common Python package that implements this algorithm. Implement and test it against standard CF algorithms.

2. Choose a data set

This could be the same or different than what you used in part I. Make sure that the data will support your project. You will need to do some minimum amount of ETL and exploration to make this determination.

3. Report

Along with the code, document your project with a notebook or markdown.

1. Clearly outline your objectives
2. Provide benchmarks. These should include a baseline bias model at the very least
3. Explore your model. Does it work equally well for all users and items? What about for less popular items or less prolific users? Think carefully about a business or technical framework to segment your data for users, items, or more, and test accuracy separately for these segments.
4. Devise methods to test for quality beyond just accuracy metrics. Consider testing coverage, novelty, serendipity etc
5. Extend your model. Can you think of any way to make your model better, like changing the objective, or adding in side information? It's ok if you can't make it better, but document your efforts and try to explain the results.

4. Evaluation

[30%] Technical correctness (see above)
[30%] Creativity

- Defining your model
- Optimizing your model
- Exploring your model

[40%] Presentation (see *writeup* above)

- This is from the perspective of a potential employer. Most interviewers instinctively try to evaluate if the prospective employee will help them solve an outstanding or upcoming business problem, and will look for evidence that the candidate has already solved similar problems before. Does this project demonstrate that the author(s) would be able to frame and solve a sophisticated personalization problem at the hiring manager's company?