Project

Build some recommender systems.

Deliverables & 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

Imagine a future employer looking at this repository. They would evaluate your project based on technical correctness, but they would also look for coherence and creativity. An employer would look for thoroughness (how does the solution scale, were the hyper parameters tuned, will this solution discover novel recommendations, etc). As a Data Scientist you would also be expected to interface with other departments in the business. Are the major highlights in 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.

Projects should be completed in Python or R.

Datasets

Datasets:

- https://gist.github.com/entaroadun/1653794
- https://gab41.lab41.org/the-nine-must-have-datasets-for-investigating-recommender-systems-ce9421bf981c
- http://www.recsyswiki.com/wiki/Category:Datasezt
- iHR - stay tuned

Part I - fundamentals

Due November 7th

Build two very simple collaborative filtering models. You may use published packages or methods - the goal of this exercise is to gain a practical intuition for these types of common models, and to develop methods to test and explore them.

- Choose a dataset / recommendation domain

- 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?
- For this section, develop with a small dataset (< 10000 users / < 100 items)
- Build two brute-force collaborative filtering algorithms:
    1. Neighborhood-based (item or user)
    2. Model-based
- Develop evaluation methods:
    1. Cross-validation setup
    2. Accuracy on training and test data
    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 how your evaluation metrics change as a function of these parameters
- 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?

Part II - deep dive

Due December 12th

Think of this part of the project as adding a bullet point or two to your resume, or adding 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:

(A) Gain practical experience with modern personalization frameworks and algorithms

Build a recommender system that goes beyond the standard CF techniques that you built from Part I. It is preferred that you build these algorithms mostly from scratch. Using previously published solutions as examples is ok, but try to avoid using black-box solutions (e.g. libFM), *except* if you are using them as a comparison to your solution.

Some examples:

- Approximate nearest neighbors
- Content or hybrid-based
- Neural network based (wide & deep, RBM, VAE, etc)
- Recommender for novelty or serendipity (use iHR data)

Who should choose this option?

> If you have limited previous experience with machine learning, are new to scientific computing, or are new to Python or R, this option is recommended. Is is also recommended if you'd like to explore new types of personalization algorithms, or tie in knowledge that you've learned from other ML domains or classes to personalization.

(B) Gain practical experience with today's primary framework for distributed ML: Spark

Build a recommendation system using Spark. You can use PySpark, which can interact with a Spark context from within Python shell or instance, or Scala (less recommended - steep learning curve!).

Some examples:

- ALS collaborative filtering
  - Try different parameters, like regularization, non-negativity, or implicit vs. explicit
  - Grid search to find the best settings for your data
- Frequent pattern mining / Association rules
  - note: FPM is available for Python and for Data Frames, but Association rules is only available for Scala and Java, and only for RDD data types
- Locality Sensitive Hashing
  - Work on a large, sparse data set
    - Optimize the hyperparamters

For all options above, how do the settings vary by data set? What about for popular items vs unpopular items, or power users versus infrequent users? How do they scale as you sample the data set?

Who should choose this option?

> If you already have hands-on experience with algorithms beyond collaborative filtering this may be a valuable skill set to hone. Practical experience with Spark is currently a major selling point to hiring managers.
>
> However, this option does carry some risk. Spark has a learning curve, and even getting simple things to run the first time can be tricky. **This is not a Spark class**; we will not be able to provide deep hands on help with Spark itself.

2. Choose a dataset

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 benchmark. 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?
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.