Practical 1: Book Ratings

Writeup due 23:59 on Friday 14 February 2014

Kaggle submission closes at 11:59am on Friday 14 February 2014

CS 181 Students: You will do this assignment in groups of three. You can seek partners via Piazza. Course staff can also help you find partners. Submit one writeup per team by the due date via the iSites dropbox.

CSCI E-181 Students: You will do this assignment on your own. Submit your writeup by the due date via the Extension School iSites dropbox.

Competing on Kaggle: You are expected to submit at least one set of predictions to the Kaggle competition online at

`http://inclass.kaggle.com/c/predicting-book-ratings`

CS 181 students should submit these as a team. There is a limit of four submissions per day. **Note that the Kaggle submission site closes 12 hours before the iSites dropbox. This is to ensure that you are able to write up any last-minute submissions.** You should be able to join the competition by registering with your `harvard.edu` or `mit.edu` email address. If you have trouble joining the contest, please email the staff list.

Warm-Up

To get warmed up for the practical, implement K-Means clustering from scratch.[1] It's a good chance to get your feet wet and try out one of the most basic and important machine learning algorithms. Go out and grab an image data set like:

- CIFAR-10 or CIFAR-100:
  `http://www.cs.toronto.edu/~kriz/cifar.html`
- MNIST Handwritten Digits:
  `http://yann.lecun.com/exdb/mnist/`
- Small NORB (toys):
  `http://www.cs.nyu.edu/~ylclab/data/norb-v1.0-small/`
- Street View Housing Numbers:
  `http://ufldl.stanford.edu/housenumbers/`

[1] That is, don't use a third-party machine learning implementation like `scikit-learn`; math libraries like `numpy` are fine.

- STL-10:
  `http://www.stanford.edu/~acoates//stl10/`
- Labeled Faces in the Wild:
  `http://vis-www.cs.umass.edu/lfw/`

Figure out how to load it into your environment and turn it into a set of vectors. Run K-Means on it for a few different *K* and show some results from the fit. What do the mean images look like? What are some representative images from each of the clusters? Are the results wildly different for different restarts and/or different *K*? Plot the K-Means objective function (distortion measure) as a function of iteration and verify that it never increases. If you're up for it, implement K-Means++ and see if that gives you more satisfying initializations. Turn in your code along with your writeup. You can write it in whatever language you want as long as you provide a README file with reasonable instructions on how we can run it in a Unix-like environment (e.g., Mac, Linux, Cygwin) and you don't require that we have to download a gigabyte of obscure software. If you use something high-level like Python or Matlab, make sure you vectorize your code or it will take forever. Running K-Means on data sets like CIFAR-10 or MNIST should only take a couple of minutes on a consumer laptop.

Predicting Book Ratings

For this practical, you are tasked with predicting people's tastes in literature. Specifically, you will be predicting the ratings (between 1 and 5) of a set of users, on a set of books. You will have some basic demographic information about the users, such as their self-reported location and age. You will also have information about the books: ISBN, title, author, publisher, and year. There are just under 13,000 users and about 130,000 books. You'll have access to 200,000 ratings (each a triplet of User ID, ISBN, and an integer 1, 2, 3, 4 or 5). Your objective is to predict the ratings associated with another 100,000 or so User/ISBN pairs.

Data Files

There are five files of interest, which can be downloaded from the course website and also from `http://inclass.kaggle.com/c/predicting-book-ratings`:

- `users.csv` – This file contains information about the 12,787 users who have provided ratings. There is a header row and then three columns in basic CSV format with comma delimiters and double-quote escaping where appropriate. The first few rows are:

```
User,Location,Age
3527,"Timmins, Ontario, Canada",0
6948,"Franktown, Colorado, USA",42
```

```
11942,"Ligonier, Pennsylvania, USA",57
7660,"Porto, Porto, Portugal",27
4886,"Neuffen, Baden-Wuerttemberg, Germany",37
1592,"Arden Hills, Minnesota, USA",0
...
```

The `User` column is a unique integer identifier. The `Location` column may contain `N/A` entries for missing data, and the `Age` column may be zero when it was unavailable.

- `books.csv` – This file contains information about the 131,378 books that have been rated in these data. There is a header row and then five columns. The first several rows are:

```
ISBN,Title,Author,Publisher,Year
0002005018,"Clara Callan","Richard Bruce Wright", \
   "HarperFlamingo Canada",2001
0060973129,"Decision in Normandy","Carlo D'Este", \
   HarperPerennial,1991
0374157065,"Flu: The Story of the Great Influenza Pandemic \
   of 1918 and the Search for the Virus That Caused It", \
   "Gina Bari Kolata","Farrar Straus Giroux",1999
0399135782,"The Kitchen God's Wife","Amy Tan","Putnam Pub \
   Group",1991
0425176428,"What If?: The World's Foremost Military \
   Historians Imagine What Might Have Been","Robert \
   Cowley","Berkley Publishing Group",2000
...
```

  Some of the `Year` columns are zero and there may be typos in some of the text fields. The `ISBN` column is unique, but note that it should be treated as a string as some of them have alphabetical letters.

- `ratings-train.csv` – This file contains the training data, which are 200,000 ratings of books by users. It is a standard CSV file with comma delimiters and a header row. The first column is unique `Id` for the user/book pair, along with the actual `User` and `ISBN`. The final column is the rating, which is an integer between 1 and 5, inclusive. Bigger is a better rating. Example rows:

```
Id,User,ISBN,Rating
247128,2178,0449911004,3
197566,943,0618129022,4
287153,1417,0930289595,5
255840,6665,0312960808,4
98408,575,0671867172,3
```

```
...
```

- `ratings-test.csv` – This file is a CSV file which contains users and books, but without the `Rating` column. Your objective is to predict these values and create a prediction file, which is described below. As in the training data, each user/-book pair has a distinct `Id`; you'll need to match this to your predictions. There are 102,226 ratings to predict. About half of these are used to compute the visible leaderboard. The other half are used to compute the true results. This separation is to prevent overfitting to the leaderboard, and is standard for these kinds of prediction contests. The first couple of rows of the test file are:

```
Id,User,ISBN
268752,3389,0446610038
80629,304,0345306880
189135,546,0440224764
270511,5153,0451524551
179535,599,0425170349
...
```

- **ratings-sample.csv** – This is an example of how you submit predictions. It is a standard comma-delimited CSV file with two columns. The `Id` column corresponds to entries in the **ratings-test.csv** file above, i.e., specific user/book pairs. The `Prediction` column is where you specify your best guess. Although the true ratings are integers, you can produce floating point numbers in your predictions. An example is below:

```
Id,Prediction
268752,3.5
80629,3.5
189135,3.5
270511,3.5
179535,3.5
86367,3.5
39319,3.5
...
```

Evaluation

After you upload your predictions to Kaggle (which you can do at most four times per day), they will be compared to the held-out true ratings. The score is computed via root mean squared error (lower is better). If there are N test data, where your prediction is $\hat{x}_n$

and the truth is x_n, then the RMSE is

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{x}_n - x_n)^2}.$$

Sample Code

Four Python files are available from the course website. The files `global_mean.py`, `book_mean.py`, and `user_mean.py` implement rudimentary predictions based on simple ideas. The file `util.py` provides some file reading and writing utilities that you might find useful.

- `global_mean.py` – Implements an incredibly simple prediction scheme, in which it computes the mean of all the ratings in the training set and then writes that out for each prediction in the test set. Achieves RMSE of 0.86606 on the leaderboard.
- `book_mean.py` – Makes predictions on the test set by computing the book-specific mean from the training set. If there are no training examples for that book, uses the global mean. Achieves RMSE of 0.92797 on the leaderboard – worse than global mean!
- `user_mean.py` – Makes predictions on the test set by computing the user-specific mean from the training set. If there are no training examples for that user, it uses the global mean. Achieves RMSE of 0.78346 on the leaderboard.

Questions and Answers

What should I turn in via the dropbox? The main deliverable of this practical is a three-to-four page typewritten document in PDF format that describes the work you did. The warmup asks you to implement a basic algorithm and turn in code, but the bulk of your writeup should be about how you tackled the practical prediction task. This may include figures, tables, math, references, or whatever else is necessary for you to communicate to us how you worked through the problem. Concretely, you should turn in via the dropbox:

- A 3-4 page PDF writeup that shows results from your warm-up and explains your approach to the book ratings challenge. Make sure to include the name of the team and the names of all partners.
- A zipped or gzipped folder containing your warm-up code and a README file explaining how to run it.

How will my work be assessed? This practical is intended to be a realistic representation of what it is like to tackle a problem in the real world with machine learning. As such, there is no single correct answer and you will be expected to think critically about how to solve it, execute and iterate your approach, and describe your solution. The upshot of this open-endedness is that you will have a lot of flexibility in how you tackle the problem. You can focus on methods that we discuss in class, or you can use this as an opportunity to learn about approaches for which we do not have time or scope. Except for the warm-up, you are welcome to use whatever tools and implementations help you get the job done. Note, however, that you will be expected to *understand* everything you do, even if you do not implement the low-level code yourself. It is your responsibility to make it clear in your writeup that you did not simply download and run code that you found somewhere online.

You will be assessed on a scale of 25 points, divided evenly into five categories:

1. **Warm-Up:** In the warmup, you'll be expected to implement a simple algorithm from scratch, run it on some simple data, and turn in your code. You'll be graded on correctness of the implementation.

2. **Effort:** Did you thoughtfully tackle the problem? Did you iterate through methods and ideas to find a solution? Did you explore several methods, perhaps going beyond those we discussed in class? Did you think hard about your approach, or just try random things?

3. **Technical Approach:** Did you make tuning and configuration decisions using quantitative assessment? Did you compare your approach to reasonable baselines? Did you dive deeply into the methods or just try off-the-shelf tools with default settings?

4. **Explanation:** Do you explain not just what you did, but your thought process for your approach? Do you present evidence for your conclusions in the form of figures and tables? Do you provide references to resources your used? Do you clearly explain and label the figures in your report?

5. **Execution:** Did you create and submit a set of predictions? Did your methods give reasonable performance? Don't worry, you will not be graded in proportion to your ranking; we'll be using the ranking to help calibrate how difficult the task was and to award bonus points to those who go above and beyond.

Bonus Points for CS 181 Students: The top three teams among CS 181 (Harvard College and local cross-registrations) students will be eligible for extra credit. The first place team will receive an extra five points on the practical, conditioned on them giving a five-minute presentation to the class at the next lecture, in which they describe their approach. The second and third place teams will each receive three extra points, conditioned on them posting an explanation of their approach on Piazza.

Bonus Points for CSCI E-181 Students: The top three individuals among CSCI E-181 (Extension School) students will be eligible for extra credit. The first place team will receive an extra five points on the practical and the second and third place teams will each receive three extra points, all conditioned on posting an explanation of their approach on Piazza. Extension school students who choose to form teams will be pooled with the Harvard College teams for purposes of awarding bonus points.

What does this have to do with unsupervised learning? It's true that this contest is about producing labels, and that there are supervised ways to tackle this that you might consider. However, some of the most successful approaches to these kinds of collaborative filtering problems have looked a lot like principal components analysis and factor analysis. Examples:

- Goldberg, Ken, Theresa Roeder, Dhruv Gupta, and Chris Perkins. "Eigentaste: A constant time collaborative filtering algorithm." *Information Retrieval 4*, no. 2 (2001): 133-151.
- Mnih, Andriy, and Ruslan Salakhutdinov. "Probabilistic matrix factorization". In *Advances in Neural Information Processing Systems*, pp. 1257-1264. 2007.
- Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." *Computer 42*, no. 8 (2009): 30-37.

What language should I code in? You can code in whatever language you find most productive. We will provide some limited sample code in Python and can also provide some support for Matlab. You should not view the provided Python code as a required framework, but as hopefully-helpful examples.

Can I use {scikit-learn | pylearn | torch | shogun | other ML library}? You can use these tools, but not blindly. You are expected to show a deep understanding of the methods we study in the course, and your writeup will be where you demonstrate this. You should not use these tools for the warm-ups.

These practicals do not have conceptual questions. How will I get practice for the midterms? We will provide practice problems and solutions in section. You should work through these to help learn the material and prepare for the exams. They will not be a part of your grade.

Can I have an extension? There are no extensions to the Kaggle submission and your successful submission of predictions forms part of your grade. Your writeup can be turned in up to a week late for a 50% penalty. There are no exceptions, so plan ahead. Find your team early so that there are no misunderstandings in case someone drops the class.

Changelog

This format for assignments is somewhat experimental and so we may need to tweak things slightly over time. In order to be transparent about this, a changelog is provided below.

- **v1.0** – 29 January 2014 at 13:00