Practical 2: Predicting Movie Opening Weekend Revenues

Writeup due 23:59 on Friday 28 February 2014
Kaggle submission closes at 11:59am on Friday 28 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/cs181-practical-2-predicting-opening-weekend-movie-revenues`

CS 181 students should submit these as a team. You should be a part of exactly one team. Do not make submissions separately from your team, and please use your real name for your user identity so that we can identify your results. There is a limit of four submissions per day, where "day" is determined by UTC. (Your deadlines are still in eastern time.) **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 started, you should take a stab at implementing basis function linear regression yourself, using some data about the g-forces associated with crash helmet impacts. Download the file `motorcycle.csv` from the course website. It has two columns. The first one is the number of milliseconds since impact and the second is the g-force on the head. The data file looks like this:

```
"time since impact (ms)","g force"
2.4,0
2.6,-1.3
3.2,-2.7
3.6,0
4,-2.7
6.2,-2.7
6.6,-2.7
```

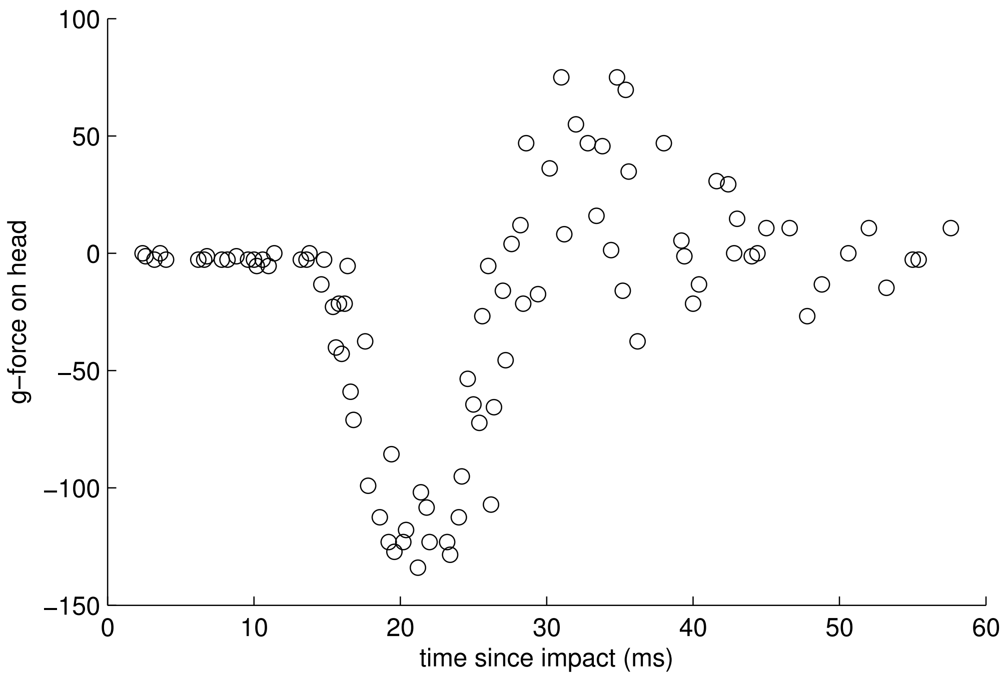

Figure 1: Motorcycle crash helmet data. The horizontal axis is time since impact and the vertical axis is force on the head.

```
6.8,-1.3
...
```

and you can see a plot of the data in Figure 1.

For the warmup, you should implement basis function regression, using both maximum likelihood and Bayesian linear regression, as described in the Bishop book. Use Gaussian noise for both of them, with fixed variance.[1] Pick a set of basis functions that seem reasonable to you. Plot the predictive mean that you find with each method, and plot predictive bars around it. For the MLE version, these predictions will only use the variance. For the Bayesian version, it will be the combination of variance and posterior uncertainty. Describe what basis functions you used and what prior you set for the weights and variance/precision. Show us your plots. Turn in your code as a zip file.

Predicting Movie Opening Weekend Revenues

In this practical, you will predict the opening weekend revenues of major films released in 2008 and 2009. To accomplish this, you will train on data from 1147 films released between 2005 and 2007. Your predictions on 317 films released in 2008 will appear on

[1]Note that the data clearly don't have fixed variance! There is obviously less variance on the left of the plot. Modeling such *heteroscedastic* data is beyond the scope of the course.

the public leaderboard, and your predictions on 254 films released in 2009 will be used in computing the final team ranking. You will have access to a significant amount of metadata to help make these predictions. Historical records for the test set are, of course, publicly available. It is expected that you will use only the data available in the files we distribute to you.

Data Files

There are three files of interest, which can be downloaded from the course website and also from Kaggle:

- `train.xml` and `testcases.xml` – These file contains information about the 1147 movies in the training set, and the 571 movies in the test set, respectively. They are XML files with root element `<instances>`. Each movie is contained within an `<instance>` element that has an attribute `id`, a unique string for each movie. The contents of each instance are:

  `regy`: Has one attribute `yvalue`, which is the first-weekend revenue in USD. This is the thing you are trying to predict and so is only present in the training data. In the test data, it has value -1.

  `name`: Name of the movie.
  Example: `Harry Potter and the Goblet of Fire`.

  `company`: Name of the production company.
  Example: `Warner Bros`.

  `release_date`: Date of film release.
  Example: `November 18, 2005`.

  `running_time`: Duration of movie, in minutes.
  Example: `157`.

  `number_of_screens`: Number of theater screens on which it was shown.
  Example: `3,858`.

  `rating`: MPAA audience rating, e.g., `G`, `PG`, `PG-13`, `R`, and `NC-17`.

  `production_budget`: Movie production budget, in USD.
  Example: `$150,000,000`

  `oscar_winning_directors_present`: If this empty element is in the `instance`, then one or more of the directors have won Oscars for previous films. In this case, the `instance` element also includes the element `num_oscar_winning_directors`, and one or more `oscar_winning_director` elements containing their names.

  `highest_grossing_actors_present`: If this empty element is in the `instance`, then one or more of the actors are among the 50 highest grossing actors. In this case, the `instance` element also includes the element `num_highest_grossing_actors`, and one or more `highest_grossing_actor` elements containing their names.

`oscar_winning_actors_present`: If this empty element is in the `instance`, then one or more of the actors have won Oscars for previous films. In this case, the `instance` element also includes the element `num_oscar_winning_actors`, and one or more `oscar_winning_actor` elements containing their names.

`summer_release`: Indicates whether the movie was a summer release, `true` or `false`.

`christmas_release`: Indicates whether the movie was a winter holiday release, `true` or `false`.

`memorial_release`: Indicates whether the movie was a Memorial Day weekend release, `true` or `false`.

`independence_day`: Indicates whether the movie was a July 4th holiday release, `true` or `false`.

`labor_release`: Indicates whether the movie was a Labor Day weekend release, `true` or `false`.

`genres`: Contains a set of `genre` elements, each with a genre for the movie.
Example: `Adventure`

`authors`: Contains a set of `author` elements, each with one of the screenwriter names.
Example: `Steven Kloves`

`directors`: Contains a set of `director` elements, each with one of the director names.
Example: `Mike Newell`

`origins`: Contains a set of `origin` elements, each with a concatenation of country names.
Example: `FranceIsraelUSA`

`actors`: Contains a set of `actor` elements, each with an actor name.
Example: `Rupert Grint`

`text`: These are text elements containing movie reviews. The attribute `tlabel` indicates the source of the review. `AC`, `BO`, `CL`, `EW`, `NY`, `VA`, and `VV` indicate that the review came from The Austin Chronicle, The Boston Globe, The LA Times, Entertainment Weekly, The New York Times, Variety, and The Village Voice, respectively.

- `p2-reg-sample-sub.csv` – A sample submission file. You will produce a similar file. The format is comma-delimited, with the first column being the film ID and the second being your prediction of the first weekend revenue.

```
Id,Prediction
10000BC,7054026.0279
21,7054026.0279
```

```
27dresses,7054026.0279
88minutes,7054026.0279
Defiance,7054026.0279
Haroldandkumarescape,7054026.0279
airibreathe,7054026.0279
...
```

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 revenues. The score is computed via mean absolute 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 MAE is

$$\text{MAE} = \frac{1}{N}\sum_{n=1}^{N} |\hat{x}_n - x_n|.$$

Sample Code

Two Python files are available from the course website. The file `regression_starter.py` and `util.py` are meant to help you get going. You definitely don't have to use them, but they provide some potentially-useful tools in which you can fill in the gaps. Specifically, it helps you write some functions that can generate features from the movie data. The file has lots of comments, so hopefully you can figure out how it works. Also included is a file `english.stop` which identifies common *stop words* that are not indicative of content in the reviews. Thanks to Sam Wiseman for putting this together!

Solution Ideas

You have a lot of flexibility in what you might do. You could focus on feature engineering, i.e., coming up with fancy inputs for your method, or you could focus on fancy regression techniques that use the features. Here are some ideas to get you started:

- **Basic regression on meta-data features:** A good place to start is to turn the metadata into a vectorial feature representation, and use a linear regression technique like ridge regression. You could use quantitative features such as production budget, binary holiday release variables, and number of Oscar-winning actors.
- **Extract features from the reviews:** You could use word counts from the reviews as features, or use something fancier like a topic model.
- **Use a sparse regression technique:** If you have a lot of features, it might be sensible to use a sparse regression method like lasso for variable selection.

- **Use a neural network:** If you think there isn't enough flexibility, you could go ahead and learn about multi-layer perceptrons and how to train them with backpropagation.
- **Use a support vector machine:** If you prefer your objectives convex, you could jump the gun and learn about regression with support vector machines.
- **Go totally Bayesian:** Worried that you're not accounting for uncertainty? You could take a fully Bayesian approach to linear regression and marginalize out your uncertainty. Intrigued by the infinite-dimensional stuff I mentioned in lecture? Check out Gaussian process regression.

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 movie revenue 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 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** – 14 February 2014 at 23:59pm