Solution Abstract Problem 2: Almost Famous

Cindy Lamm

Wednesday 21st January, 2015

1 Approach

I identified question 1 as classification problem (supervised or unsupervised) which I wanted to solve using Spark (after the performance problems I encountered for the classification in R with 7.3 million observations).

I opted for the following approach to solve the questions:

- analyse file structure & convert to csv
- explore data
- prepare data for modeling
- estimate and validate models for spam classification
- prepare the web data for prediction
- classify entries in web data as spam
- remove spam entries from web data
- calculate required metrics on spam free web data

1.1 Analyse file structure

I had a short look into the log data in json format via jq. Once I saw that the data is fairly regular (same names for keys, same number of keys per row even if different row "types") I converted the data to csv[1] using json2csv (which is pre-installed in the data science toolbox) to load it into R for exploratory analysis.

1.2 Explore data

To get an impression of the web and spam data I had a look at overall summary statistics but also at summary statistics aggregated over visits and visitors. Since I am not that familiar with web analytics I assume that a `uid` uniquely identifies a visitor (which I will use as a synonym for user).

From manual inspection it seems that visits that are generated via the bot network only contain the action `adclick` after they landed on the page. However if I would classify the web data according to this simple rule, then the click through rate for ads on the page would be zero - which seems too extreme.

[1] see `problem2_famous/data/convert2csv.sh`

1.3 Prepare data for modeling

In order to run a classification/clustering algorithm in Spark I need to code the categorical variables (all but `tstamp`) with integer levels (which means that I implicitely assume that it is okay if an algorithm treats these variables as continuous features, because the resulting model is still good enough). I converted the `tstamp` to epoch values (i.e. to seconds since 1970-01-01 00:00:00 GMT) to have a true continuous variable without loosing the time information.

For the supervised classification I added a variable `is_spam` with value 1 to the spam data and with value 0 to the web data (knowing that with this assumption some web data is falsely labeled). I pooled the spam data and different amounts of the web data into a training data sets - which I used as well for testing (see section 1.5).

1.4 Estimate and validate models for spam classification

I tried different classification and cluster algorithms: I started with unsupervised clustering using k-means on a training sample with all spam log data and (the top) 200.000 observations of the web log data[2]. I had the hope that by using $k = 2$ all spam data would end up in one cluster and all web data in the other, so that when I predict the complete web data on this clustering, all so far unknown spam entries go into the spam-cluster. However when I used the estimated cluster centers to see the division into clusters of the training data, the result was not as good as I would have expected:

TODO: add venn diagram here or error/accuracy values

1.5 Current Limitations Of Spam Classification

I used the same data for training, validation and prediction, which decreases the generalization performance of the estiamted model. It would be better to split the available spam and web log data into independent sets for training and validation.

Given that I relabeled the categorical variables with (rather small) integer labels it might be necessary to scale the epoch values, because they otherwise bias the differences. It might also be better to use 1-of-n encoding for the categorical variables instead of just giving them numerical labels.

Since the results are not as expected I would look for better input features, e.g. the time difference between the actions and/or the total time spent per visit. I also deem worth trying to cluster/classify on aggregated visit and/or user data.

1.6 Compute Required Metrics

Even if I was not able so far to properly classifiy the spam I prepared all the code to compute the required metrics.

Overall Click-Trough Rate

I assume the following formula for the overall click-through rate (CTR), that was described as "ad clicks per visit"

$$CTR = \frac{\#\{action = adclick\}}{\#\{unique\,visit\,ids\}}$$

Without having removed the spam visits from the web log data I counted 103896 ad clicks and 1482602 unique visit ids, which leads to a CTR of 7.01% - which is unusually high (if we believe in wikipedia[3]) and certainly prooves the necessity to exclude the spam entries from the calculation.

[2] Using random 200.000 observations from the web log data didn't seem to be a good idea given that entries are ordered by time.

[3] http://en.wikipedia.org/wiki/Click-through_rate

Mean Orders Per Visit Within Campaign-Query Combinations

Since the action "order" is only listed in log entries that don't have query and campaign information, the first step I did was to aggregate the web log per visit putting all actions in a list. I saved the aggregated visits data back to a log file, converted it to csv and worked on that in R to calculate the mean and standard deviation of number of orders within the groups formed by the campaign-query combination (using the ddply function of package plyr).

Note: Of the 50 possible campaign-query combinations only 10 have any entries in the web.log data. So I only wrote the standard deviations of the number of orders per campaign-query combination that have a non-zero value.

Newsletter Signup Rate

Before I compute the newsletter signup rate for each experiment I double-check the assumption that users see experiments independently by verifying that the distribution of experiments in the data is even[4].

I used the per visit aggregated web log data and computed in R the newsletter signup rate per experiment using the again the ddply function.

Possible Improvements

I wrote a separate Spark Python script for each of the metrics I computed with Spark, where I loaded the data from the file system in each script. This could be improved by calculating all metrics within one Spark session for which I would load the data only once from the file system (but need to refactor the code to make it reusable).

2 Used Software

In general I developed on a Macbook Pro with a 2.3 GHz Intel Core i7 Processor and 16GB Memory.

- git
- jq
- json2csv (pre-installed in the data science toolbox)
- LateX
- R in RStudio with packages knitr, plyr
- shell command line
- Spark with Python API and MLlib package

3 Time Spent

- 24h for data cleaning, preparing for and running the spam classification
- 12h for computing CTR and metrics for campaign-query combinations
- 1h for solution abstract

Summing up I spend xx hours on this problem.

[4] see `problem2_almost/src/q4a_newsletter_signup/newsletter_signup.pdf`