CCP Web Analytics Challenge

Overview

Given two months of logfiles from Cloudera Movies, solve three challenges. Firstly, predict whether the website users are of type adult or kid. Secondly, cluster web sessions into similar groups and thirdly, predict the preference ratings of a list of specific user item pairs. The logfiles provided are from two servers named heckle and jeckle and are in JSON format.

Approach

The first stage in solving the solutions to the above three challenges was to first gain an understanding of the data. I spent the first few days trying to understand what was in the data and what was a signal and what was noise. The logfiles were in a format with a "payload" object embedded in a "type" object. Not all '"type" objects had a "payload" object (e.g. Login).

Payload contained information related to movies, from what movies to display on the Home page to what movie a user was watching (a "Play" action). However, before I could start to understand the data, I had to clean up the logfiles to make them consistent, correct some formatting issues and also standardise the timezone. This led me to my first task which I call Task0.

Task1 I realised was a binomial generalised logistic regression problem and chose to solve this using R and packages (sparse) matrix and glmnet. A series of distributed MapReduce and Hive jobs extract the data as input for a glm predictive model coded in Task1Solution.R.

Task2 was a difficult problem to solve. Once I understood that item ids in the logfiles which contained an "e" referred to a TV episode (confirmed by Floyd Bush, Cloudera Movies), then it seemed reasonable to sessionise the logfiles and see whether any clusters formed between those sessions which watched TV (and how many episodes), those sessions which watched movies (and how many movies) and those sessions which did something else. To solve this problem I chose to use the Cloudera Machine Learning toolset and specifically the Kmeans++ algorithm.

Task3 involved making predictions of user item pairs and was obviously a collaborative filtering type problem, but my research revealed that this again was a difficult problem to solve because of the lack of data available in the logfiles. After extracting all available data, the user item matrix was only 7% complete which introduced the problem of sparseness and I quickly learned that traditional user based and item based methods of predicting preference were failing. For this problem I used the Mahout implementation of Alternating Least Squares with Weighted Lambda-Regularization.

Task 0 – Preparation

Task0 is a MapReduce job which takes as input the raw logfiles, uploaded to a HDFS directory structure as follows:

`~ds-shorsman` – root directory

`ds-shorsman/incoming` – directory for files which are to be input to MapReduce jobs

`ds-shorsman/outgoing` – directory for files outputted from MapReduce jobs.

The heckle and jeckle logfiles are saved into the local "~/ds-shorsman/data" directory. A workflow script called "~/ds-shorsman/bin/task0_workflow.sh" uploads the files to HDFS and a MapReduce job cleans the logfile. The logfiles had 3 distinct problems:

1. The JSON was corrupt. An extra " mark appeared at itemId.
2. Multiple names were used to describe the same value, such as "recs" and "recommendedItems".
3. The timezone was either in Zulu time (UTC) or PST (GMT-08:00).

The output of “~/ds-shorsman/bin/task0_workflow.sh” is a single log file, cleansed and formatted, stored in the “ds-shorsman/incoming” HDFS directory.

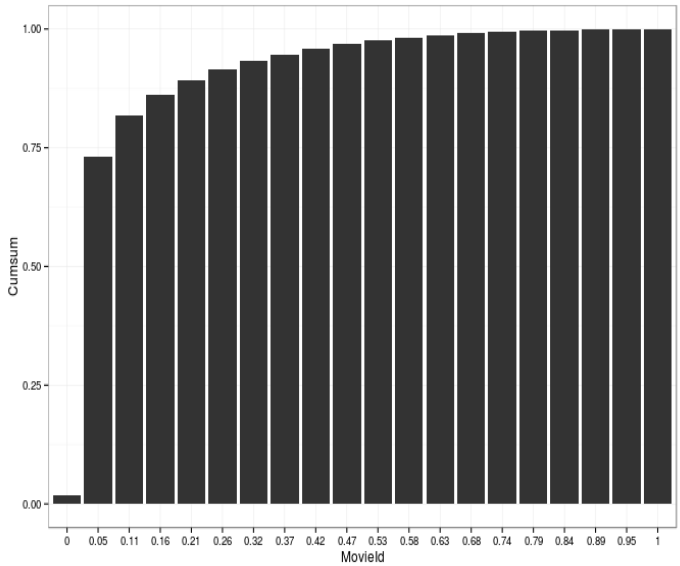
Task0 – Analysis

The first task I wanted to understand was how many object types were distributed throughout the dataset. This was an easy task of creating a MapReduce job which just counts each action and then aggregates. The results can be seen below:

It is obvious (and expected) that the Play action dominates the activity in the logfiles.

Account	12410	Advance	237366	Hover	1653113	WriteReview	18801
AddToQueue	378869	Home	460204	ItemPage	18798	Search	112149
Login	89669	Logout	89521	Pause	387375	Stop	628718
Play	48636796	Position	35745	Queue	111454	VerifyPassword	9889
Rate	70199	Resume	137069			Recommendations	112433

The next analysis I performed was to count the number of times each movie was played and then display this information as a cumulative sum (Rscripts/Task0Analysis.R). The graph is below:



What this shows is that about 5% of the movies count for almost 75% of the total play count. This is important because for Task1, we will try and predict whether an account is an adult or child, based on the movies they watch and it is clear that some movies may be strong predictors of this identity. Here is top 5 movies by count and cumulative sum:

MovieId	Count	Percent	Cumsum
10375	1030246	0.02	0.02
11108	906677	0.02	0.04
26355	291983	0.01	0.05
27370	265523	0.01	0.05
36851	249710	0.01	0.06

Indeed, the most watched movie id, 10375, appears never have been watched by a known adult account and becomes a strong predictor that the account is a kid. We know that 10375 is a kids movie (because kids can't watch restricted movies due to the parental control). We can use this information for Task 1.

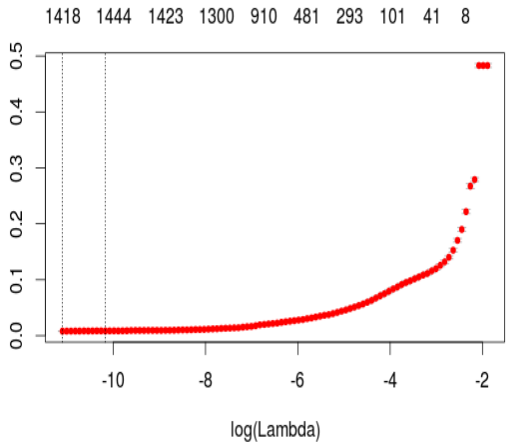
Task1 – Classification

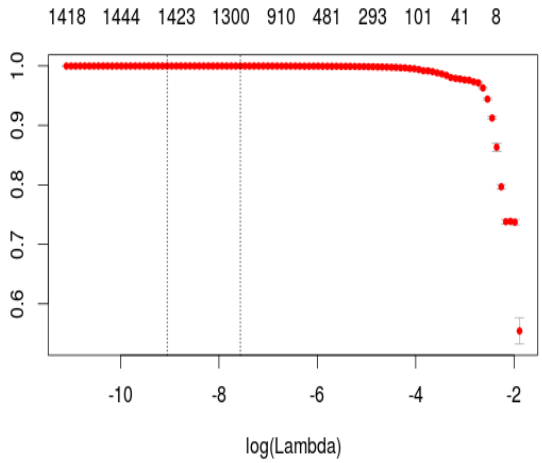
The method I used to solve this problem was based around two tables extracted from the logfiles. The first one contained all those accounts which had a parental control applied to the account, the account id and the action date. The date is important because it tells us that before the date, the account was an adult and after the date, the account became a kid. This should be reflected with a change in viewing pattern. The next table I created was a play table which contained a row for each play action, which gave me the user id, movie id and the date associated with this action. To build the training set, I simply joined these two tables on user id. I selected all accounts prior to the parental control and their viewing history for the adult users and likewise selected all ids post parental control and their viewing data. With this information I could train a predictive model and make predictions on the remaining user accounts found in the logfiles.

My training set was broken down as follows. I had 16984 user ids where the account had both been a kid and an adult. This broke down to 8781 kids and 8203 adults so the classes were roughly balanced. The differential is explained because those accounts with no adult play history are just simply new kid accounts. These accounts had watched some 4966 movies and after doing some cross validation, found that the movies above which contributed most to the play activity were included in my user item matrix.

To train the model, I split the training set between training (70%), test (20%) and validation(10%) and ran a variety of test scenarios. The glmnet R package also has cross validation testing functionality and I used this to cross check the values for lambda and deviance. The plots I produced from the cross validations and confusion matrices were very interesting and the results suggested I had a highly accurate model (99% accurate). The reader can run the analysis (Rscripts/Task1Analysis.R).

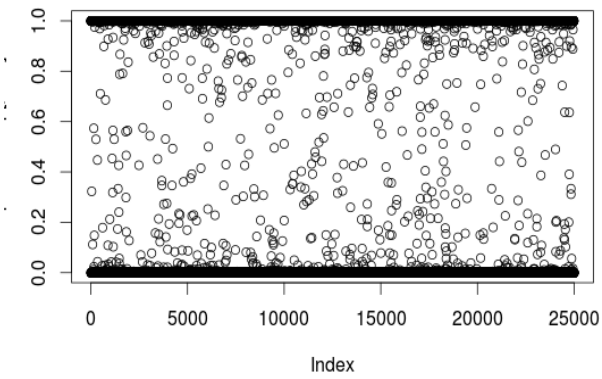
The graphs below show the misclassification error (left) and AUC (right) of the glmnet model. The model





converges at a value for lambda and the number of variables (movies) by co-ordinate descent.

The final plot to the right shows the predictions made by the model. The decision boundary is between being an adult or kid is obviously at 0.5. Misclassification can occur in the scenario where an account switches from adult to kid, but then the account is barely used by the kid, say they just watch 1 movie and this movie also shows up as viewed by adults, given our adult population is free to watch any movies. Likewise, we may classify adult accounts as kids if the adult just watches kid movies. Both scenarios will likely happen, but will be outliers. My final prediction yields 10740 adults and 14260 kids, 8923 which are known from parental controls.



Task2 – Clustering

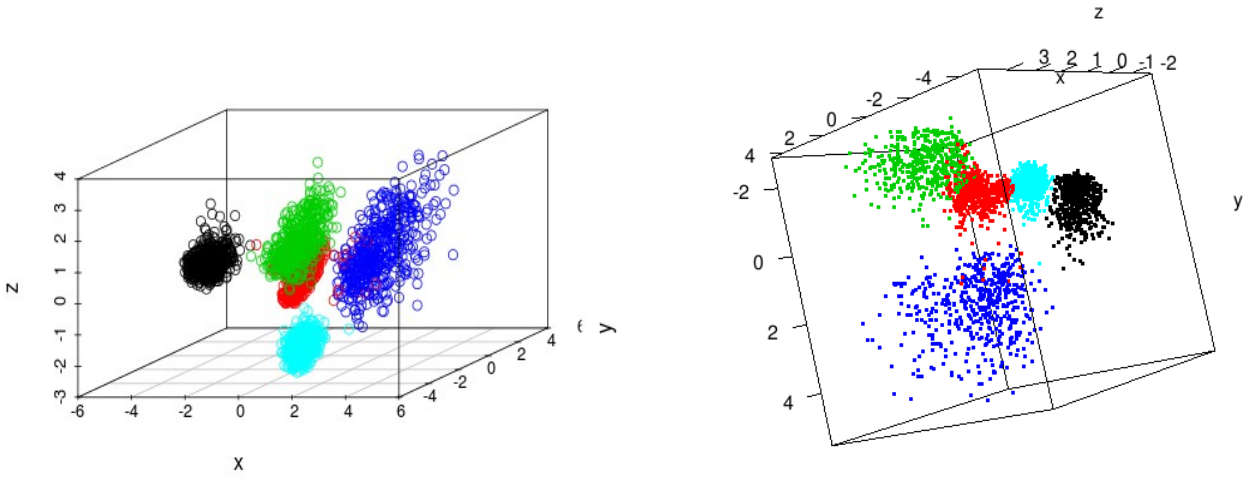
The second challenge was to cluster sessions based on certain attributes into similar groups. One could imagine that a typical session would involve logging on, browsing the home page, possibly recommendations, selecting a movie to watch, playing that movie, stopping the movie and then logging off. Another typical session might be if the user was watching a TV series then this might involve watching multiple TV episodes. Other sessions might update account information, or search for items ready for viewing in a future session. If we can sessionise the data, we can profile our customers better into those that watch predominantly TV compared to those that watch predominantly movies. We can also sum the session duration and find our top customers which spend the most amount of time on the website, because these users are our most important and we might want to target them with some sort of loyalty based marketing programs. It is hoped that the clustering algorithm find this data in our logfiles.

A series of MapReduce jobs sessionises the data. For this challenge, I chose to aggregate the session times into seconds per action and also sum the amount of times a unique movie or tv episode was seen by the user. So the output was in csv format where each row represents a session and each column time in seconds spent on each action.

The clusters that the kmeans algorithm found are as follows. (Analysis of results from ~/ds-shorsman/Rscripts/Task2Analysis.R).

Cluster	Description
0	All sessions which watch TV shows, grouped together into 1 cluster.
1	Sessions which watch movies, predominantly just 1 movie and which predominantly did not rate a movie.
2	Sessions which watch movies, predominantly just 1 movie and all sessions spent significant amounts of time rating a movie.
3	Sessions which watch movies, predominantly multiple movies in 1 session and which predominantly did not rate movies.
4	Sessions which watch movies, predominantly just 1 movie and completely ignored recommendations.

The two 3d plots below visualises our clusters (~/ds-shorsman/Rscripts/Task2Analysis.R).



The results were extremely interesting. Visiting the data again, the ratings and recommendations columns are related, in the fact that if users don't rate movies, this leads to sparse data which is a weak input into our recommendation engine. Clusters 1 and 3 are responsible for the formation of cluster 4. Cluster 1 contain 260807 sessions. It would be useful to understand why these sessions aren't rating the movie they've just seen. Maybe the actual web application is responsible. For example, an enhancement could be to prompt the user to rate the movie when the movie finishes or the user clicks “Stop”.

Task 3 – Recommendations

The final challenge was to predict preference ratings for a given set of user, item pairs. I'd previously used the recommenderLab package in R so was initially confident that I could solve this problem fairly easily. I'd spent some time studying and coding user based collaborative filtering solutions for a previous work task. However, after digging into the data a bit, I quickly realised that due to the sparseness of the data, using recommenderLab was no longer an option. It was taking somewhere in the region of 12 seconds to make a prediction, which given I had to make 10,000, would result in the job running for about 33 hours, which was unacceptable. Also, the UBCF model was also failing to predict some preferences due to data sparseness. I needed a new approach.

Even though I couldn't use recommenderLab to make predictions, it was useful for understand the distribution of ratings within the data I had. I calculated the mean user rating to be 4.112296 which if I'd just blindly made all predictions with this rating would have given me a RMSE of 1.101202. This was the benchmark for all other models to beat.

I recalled that a similar challenge was put out by Netflix, probably the most famous data science challenge to date so did some reading around this. I knew about Mahout but had never used it previously so after discovering that the Alternating Least Squares with Weighted Regularisation (ALS-WR) algorithm, described by an HP research paper into the Netflix challenge (~/ds-shorsman/docs/als-wr.pdf) had been implemented in Mahout, it seemed reasonable to use this. The Netflix data was also sparse. A single MapReduce job extracts the information from actions “Rate” and “WriteReview” and organises the data into “userid,itemid,rating” format. For user item pairs which have been rated more than once, the reduce code takes the last known rating made by the user. I created a new Mahout class called PredictorEvaluator.class. It was based on FactorizationEvaluator.class which calculated the RMSE of the ALS-WR model, so calculated the error between simulated and observed ratings. I copied this job, made some alterations and setup Mahout so that you can call a evalutePrediction Mahout job, pass in a user,item prediction file and the job will output the results. The final RMSE predicted by the ALS-WR model was ~0.988.

The next problem I had was that some items which required a prediction for the challenge, had no previous rating performed by any user, so was therefore not being outputted by the Mahout ALS-WR model. This posed a problem. I did some analysis in R. I tested a number of scenarios around using the mean value. I calculated a separate mean rating for kids and adults, theorising that these two demographics might rate differently. Based on my predictions from Task1, it slightly improved the (weighted) RMSE from 1.101202 to 1.094034. Given the tiny bump in RMSE, it wasn't worth the coding effort to introduce the predicted data. I also toyed with the idea of using the recommended items information and assigning a 5 rating (assuming that the existing recommender engine was performing top N ratings). However, my testing showed this introduced massive bias into the results and worsened RMSE. In the end, I settled for just using mean which was justified from my analysis. The final plot shows the RMSE testing I did to find the values for number of iterations and lambda which was used by the ALS-WR model. I ran a number of evaluation tests with varying iterations and lambda as parameters of the mode. A value of 0.065 for lambda and 25 iterations gave me the lowest RMSE. The red line represents the RMSE when using the mean value for predictions.