k-split based approach to predict movie rating frequency

Hariprasad Bommaganti
Yahoo Inc.
Santa Clara, CA
hari@yahoo-inc.com

Anand Nagarajan
Symbram LLC
Sunnyvale, CA
anand@esymbram.com

ABSTRACT

In this paper, we describe a regression-based approach to predict movie-rating frequency using Netflix user-movie rating dataset as part of one of the challenges in KDD cup 2007. The task required predicting the additional number of user ratings that each movie gets in 2006, given the history of 100 million user ratings for 17770 movies for 1998 – 2005. In our approach, the user ratings were considered as a time series. For varying *k*, we split the ratings into *k* equally spaced time intervals and computed user-rating based features for each interval. Model generation consisted of two phases. In the first phase, for varying *k*, we regress models that predict movie ratings for 2005 using input features generated from 1998-2004. In phase2, cross validation RMSE obtained for various models in phase 1 was used to choose the best predictor and best *k*. This was used to predict movie rating frequency for 2006 on the test set using input features generated from 1998-2005. We trained three different regression methods – SVM, Random Forest and Gradient Boosting and found that Random Forest performed better than other two regression methods with the lowest RMSE of 0.5532 on test set for *k*=10.

Categories and Subject Descriptors

G.3 [Probability and Statistics]: Correlation and Regression Analysis, Time Series Analysis.

General Terms

Algorithms, Measurement, Design, Experimentation.

Keywords

time series, forecasting, feature extraction, svm, random forest, gradient boosting, predictive modeling, regression.

1. INTRODUCTION

In many real world problems, situations arise where a sequence of observed data points are spread out over a time interval. Such a dataset can be viewed as a time series. Of particular interest is the study determining the underlying distribution that generates the time series and prediction of future data points based on the existing data. [1,3].

Time series are generally sequences of data points of one or more variables of an underlying system, whose state changes with time as given by [4]:

$$\frac{du(t)}{dt} = F(u(t)) \qquad \text{[E1]}$$

where, u(t) is the current state vector.

Stock and weather data are good examples of such time series data. Likewise frequency of users who rate a movie varies over time and can be considered as a time series. The second task in KDD Cup 2007 constitutes predicting the additional number of user ratings that a movie gets in 2006 given the history of 100 million user ratings for 17770 movies from Netflix during 1998-2005 [5,6]. To model this problem, we find features that reflect the growth or staleness (inactivity) in user rating behavior over time, thereby transforming it into a regression task. Upon training set generation, we created regression models to predict user rating frequency in 2005 based on user-rating data from 1998-2004 that formed the training set. The best model was selected based on cross validation RMSE and the test set was generated using user-ratings from 1998-2005. The best model selected during the training phase was applied on the test set to predict rating frequency for 2006. We compared the performance of different regression methods such as random forest, SVM and gradient boosting. We showed that random forest performed better overall.

The paper is further organized as follows: In section 2, we discuss how we explored the user-rating data to generate features that reflect user-rating frequency trend over time. We also introduce our *k*-split approach for the problem. Section 3 describes our experiments and observations with Netflix user-rating data centric to predicting the rating frequency in 2005 and 2006 using the *k*-split approach. In Section 4, we discuss our results and conclude with remarks on future work that can be done to improve upon this simple approach.

2. OUR APPROACH

In this section, we discuss the approach we used for movie ratings forecasting task. In this task, we were provided with Netflix user-rating dataset that contained a history of 100 million user ratings for 17770 movies between 1998 and 2005 [6]. Each user-movie rating was also accompanied with the date of rating. This task required predicting the additional number of user ratings that each movie got in 2006, from the original user base (users who exist in the given 100M ratings). We considered user ratings along with date information for each movie as a time series. We used date information to split the user-ratings across equally spaced time intervals. Features were explored within and across these time intervals.

In the time series literature, the approach of splitting time series data across time intervals and considering a sliding window based method to compute features is a well established procedure [4]. In sliding window approaches, time series data is split into *n* (pre-defined) equal intervals and a window of particular size is used to move across these intervals. For a fixed window size *w*, we have *(n–w+1)* training examples for each sample (in our case, each movie) and a regression model on these training examples is trained. Figure 1 illustrates how training examples are generated in a sliding window approach. Heuristic methods to guesstimate the right window size exist in literature [2].

In this approach, we explore the *optimal* time interval *t* by splitting the given time series dataset into *k* equally spaced time slots. Figure 1 illustrates how training examples are generated in our *k*-split approach. In this case, if T_s denotes starting rating date and T_e denotes last rating date in the given dataset, we compute the time interval as follows:

$$t = (T_e - T_s) / k$$

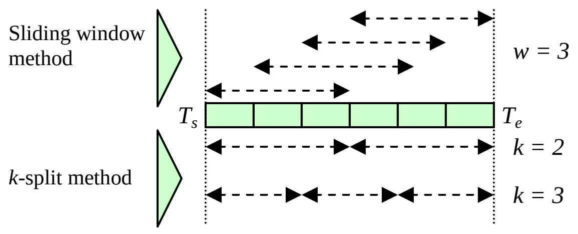

Figure 1: Sliding window versus k-split

Above transformation (as depicted in Figure 1) on the time series data generates a regression model input dataset. The main idea of our approach is to search for an *optimal* value of *k* that would produce least RMSE, where *k* is the number of different time intervals that we split the initial time series into. Root mean squared error (RMSE) is calculated between predicted rating frequency and actual rating frequency. For this problem, we normalize the target variable (rating frequency) used in regression by taking its logarithm. RMSE was computed as follows:

$$rmse = \sqrt{\frac{\sum (\log(x+1) - \log(y+1))^2}{n}} \qquad [E2]$$

Where, *n* is the total number of examples in our dataset. *x* is actual rating frequency and *y* is predicted rating frequency for each example.

2.1 Feature Exploration

We developed a variety of features that can be broadly categorized into two classes –

1. User Rating Features
2. Collaborative Features

User rating features can be loosely defined as user based rating statistics within and across various time intervals. We consider three main types of user rating features –

- **Overall User Ratings**: Total user ratings for a given movie within a time interval.
- **Seen User Ratings**: Frequency of user ratings within a time interval, where user had rated previously (in earlier time intervals).
- **Unseen User Ratings**: Frequency of new user ratings within a time interval.

The seen user ratings within a time interval provide information about ratings given to each movie by users who have rated at least one movie earlier. Frequency of seen user ratings over different time intervals can be useful to capture the basic rating *trend* for a given movie. The frequency of unseen user ratings on the other hand describes the growth of new users for each movie across time intervals. This is a very useful feature as it indicates how new users affect the overall rating frequency over time.

We further explored features such as percent of seen users and unseen users growth (or lack thereof) within a time interval. Additional aggregate features across all time intervals were added to capture the popularity or seasonality of movie ratings. Other features such as cumulative growth of seen users and unseen users were also computed. In computing the above features we normalize feature values by using log function. This was done to mitigate exponential growth factor associated with user ratings.

To consider the collaborative influence of how rating one movie by a user would have on another, we were interested in representing each movie in terms of users who watched it. Since number of unique users is quite large (~500K), a reduced representation was required to encode users who watched a movie. In essence, we wanted to associate the affinity of some large clusters of similar users to each movie. We used SVD [7,8] for this purpose as follows:

Let *A* be the given user-movie rating u X m matrix where *u* is the number of users and *m* is the number of movies. Singular value decomposition (SVD) of *A* is the factorization $A = U \Sigma M^T$, where *U* and *M* are orthogonal, and $\Sigma = diag(\sigma_1, ..., \sigma_r)$, *r* = min (*u*, *m*), with $\sigma_1 \geq ... \geq \sigma_r \geq 0$. σ_i are called the singular values, the first *r* columns of *M* are the right singular vectors and the first *r* columns of *U* are the left singular vectors. We used top ten features from each vector (row) in M corresponding to each movie. Intuitively, each of these feature values for a movie denotes the contribution of some large cluster of similar users.

2.2 Modeling

Training sets for our *k*-split approach were generated using ratings from 1998-2004 and the rating frequency in 2005 was used as the target variable. We generated models for different values of *k* (number of splits) ranging from 2 – 16. For each value of *k*, we generated a training set T_k across all movies. Input features for T_k were computed as described in Section 2.1. We trained three different regression methods in our experiment namely SVM, RandomForest (using Regression Trees) and Gradient Boosting Regression Trees. Four-fold cross validation RMSE (root mean squared error) as computed in equation E2 was used as the metric to pick the best model and best value of *k*. Figure 2 illustrates the k-split approach in detail.

Let,

M	Set of movies given
R_1	Set of user ratings in 1998 - 2004
R_2	Set of user ratings in 1998 - 2005
f(m,k,R)	Function to transform ratings (R) into input features for given movie 'm' and number of splits 'k'
L	Set of learning models

Training Phase

```
For k = 2, 4,…, 16:
  Train(k) = { f( m, k, $R_1$) | m $\epsilon$ M }
  Train learning model $l_k$ using Train(k)
  Add $l_k$ to L

$B_m$ = $l_k$, best = k, where $l_k$ $\epsilon$ L and RMSE($l_k$) is minimum
```

Testing Phase

```
Test = { f( m, best, $R_2$ ) | m $\epsilon$ M }
return predicted ratings for Test using $B_m$
```

Figure 2: k-split methodology used in training and test phases

Test set generation was slightly modified to take into account that we had to predict rating frequency for 2006. For this scenario, input features were generated using ratings from 1998-2005 (instead of 1998-2004, which was used for training set). The best value of *k* obtained during cross validation on training set was used to split user ratings and generate input features for test set.

The best model selected in the training phase was used to predict the rating frequency in the test set for 2006.

3. EXPERIMENTS AND OBSERVATIONS

We compared SVM, Random Forest (RF) and Gradient Boosting Trees (GBM) on our *k*-split approach using cross validation RMSE. Figure 3 illustrates a plot of RMSE for different values of *k*.

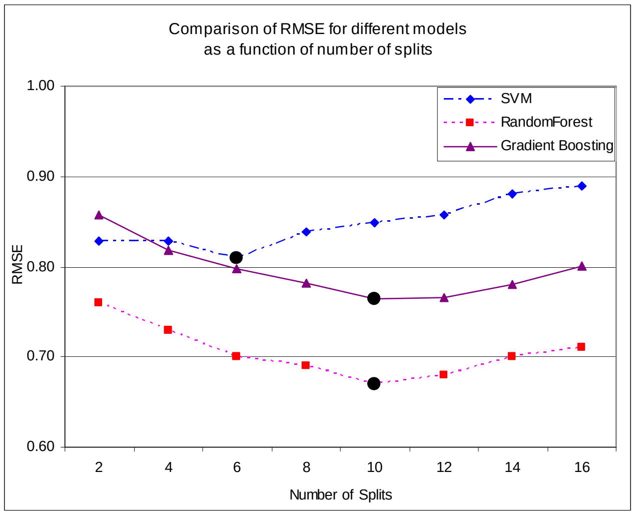

Figure 3: RMSE for different models for various values of k (2, 4, … , 16)

We made the following observations –

For small values of *k*, RMSE of all models decreases as *k* increases. But for higher values of *k*, RMSE starts to increase again. This can be justified intuitively since with increasing *k*, we generate more input features in our training set. These additional features seem to provide more information relevant to the regression process. However for larger values of *k* (esp., $k > 10$) this information about time series in the additional splits tends to become irrelevant (or rather redundant). We note similar observations made in literature for using large window sizes in sliding window techniques [4], i.e., larger window sizes tend to add noise to training set.

Of the three learning methods we trained, random forest performed relatively much better than SVM and GBM. We also noted that optimal value of *k* for SVM was different from that for random forest and GBM. Best values of *k* for SVM, RF and GBM were 6, 10 and 10 respectively.

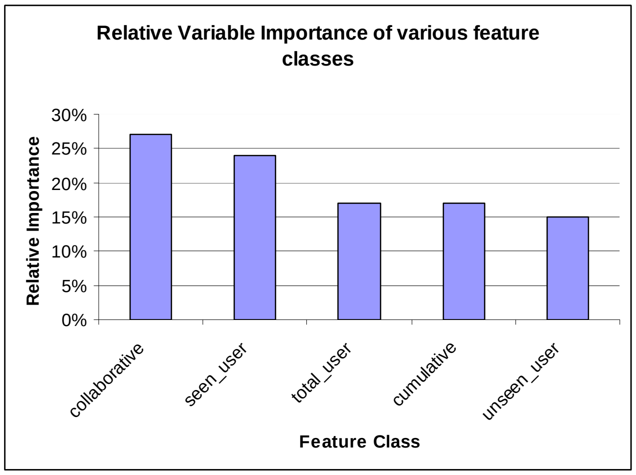

Figure 4: Relative variable importance of various feature classes

Figure 4 illustrates the relative variable importance of different feature classes used in training random forest for *k*=10. Variable importance is measured as described in [3]. We combined the variable importance measure of features belonging to the same class to plot the relative importance of different feature classes. Feature classes compared include collaborative features, seen users, unseen users, cumulative frequencies and total users in different time intervals. Interestingly, we observe that across all time intervals, collaborative features play an important role in predicting user frequency. Other highly used classes of features include seen users and total users. We also note that features of recent time intervals are more important than older time intervals (this is usually expected in a time series).

4. RESULTS

Cross validation RMSE for the three learning methods generated using training set is listed in Table 1. Training set was generated using ratings from 1998-2004, with target variable being rating frequency in 2005. Lowest RMSE of 0.67 was obtained for random forest (RF) for *k*=10 upon cross validation using training set.

Table 1. Cross Validation RMSE on Training set on SVM, RF, GBM for various values of k

Learning Model	Number of splits (*k*)							
	2	**4**	**6**	**8**	**10**	**12**	**14**	**16**
SVM	0.83	0.83	0.81	0.84	0.85	0.86	0.88	0.89
RF	0.76	0.73	0.70	0.69	**0.67**	0.68	0.70	0.71
GBM	0.86	0.82	0.80	0.78	0.76	0.77	0.78	0.80

Based on cross fold validation RMSE, random forest was chosen as our best model for test set. In Table 2, we illustrate the performance of random forest on test set for different values of *k*. We noted that selection of best split size (*k*=10 for random forest) was justified in test set as RMSE on test set was found to be minimal at *k*=10. We also noted that the RMSE starts increasing for *k* > 10 in the test set as well. However, at the time of actual submission for KDD cup 2007 task 2, we had evaluated for *k*=2, 4, 6 only and submitted results we had obtained for *k*=6 (RMSE=0.607).

Table 2. RMSE for random forest model on test set for various values of k

Learning Model	Number of splits (*k*)							
	2	**4**	**6**	**8**	**10**	**12**	**14**	**16**
RF	0.691	0.668	0.607	0.579	**0.553**	0.564	0.585	0.601

5. CONCLUSION AND FUTURE WORK

In this paper, we presented an approach to modeling times series regression task for KDDCUP 2007. In this method, we split the time series dataset into *k* equal intervals and apply regression methods to empirically pick the best model and estimate an optimal value of *k* based on RMSE. This technique is a variation of sliding window method that works particularly well on time series datasets. Our approach performed very well for predicting movie rating frequency in 2006 using historical user ratings from 1998-2005. *k*-split methodology explored in this paper is simple and can be extended in future to build ensembles of mixed regression models which use different values of *k* for each individual model. We also plan to explore further comparisons with sliding window methods.

6. ACKNOWLEDGEMENTS

We would like to thank the KDD 2007 organizing committee for conducting KDD cup and providing us with an opportunity to participate in it. We also would like to thank the Netflix team for providing us with a valuable dataset to analyze user-movie ratings.

7. REFERENCES

[1] Ababarnel, H. D. I., Brown, R., Sidorowich, J. L., and Tsimring, L. S., The analysis of observed chaotic data in *physical systems. Reviews of Modern Physics*, Vol. 65, No. 4 (1993), 1331-1392.

[2] Bowman, B., Debray, S. K., and Peterson, L. L. Reasoning about naming systems. *ACM Trans. Program. Lang. Syst., 15,* 5 (Nov. 1993), 795-825.

[3] Breiman, L., Random Forests. *Machine Learning, 45,* (2001), 5–32

[4] Frank, R. J., Davey, N., and Hunt, S. P. Time Series Prediction and Neural Networks. http://www.smartquant.com/references/NeuralNetworks/neural30.pdf

[5] KDDCUP 2007, http://www.cs.uic.edu/~liub/Netflix-KDD-Cup-2007.html

[6] Netflix Prize, http://www.netflixprize.com

[7] Timely Development Netflix Prize Demo, http://www.timelydevelopment.com/Demos/NetflixPrize.htm

[8] Wall, Michael E., Andreas Rechtsteiner, Luis M. Rocha. Singular value decomposition and principal component analysis in *A Practical Approach to Microarray Data Analysis*. Berrar, D. P., Dubitzky, W., and Granzow, M. eds., Kluwer: Norwell, MA (2003), 91-109.