Evaluating the effectiveness of regularized logistic regression for the Netflix movie rating prediction task

Adam Sadovsky
sadovsky@cs.stanford.edu

Xing Chen
xingchen@cs.stanford.edu

1 Introduction

Netflix Prize is a competition created by Netflix that challenges individuals to predict how a user will rate movies he or she has not seen, given the ratings for movies he or she has seen. The Computer Science Department at Stanford has built a team that aims to tackle this problem by developing and testing various machine learning approaches such as matrix factorization, clustering, various types of linear and logistic regression, etc. One of the most successful prediction methods thus far has been logistic regression with L2 regularization, which achieves approximately 0.93 root mean squared error (RMSE) on the probe set. However, because L2 regularization imposes a Gaussian prior on feature weights, it tends to assign non-zero weight to all features; for various reasons, L1 may outperform L2. Furthermore, because L1 regularization results in zero weight for a large majority of features, it can also help us construct more complex models (such as Markov networks with movie potentials).

2 Data Representation

Netflix provides contest entrants with several datasets: (1) a sparse matrix consisting of 17,770 movies and >400,000 users, with integer ratings (1 to 5) for select user-movie pairs, (2) a separate probe set (which includes ratings) on which to test prediction accuracy, and (3) a set of user-movie pairs for which predictions must be sent to Netflix for assessment. In uncompressed form, the training set size is approximately 1.5 GB. The sheer size of the training set makes it difficult to run queries and generate features from the raw data. Furthermore, because a database (SQL) representation of the training set wasn't available to us, we had to write our own Perl and Matlab scripts to parse the training set, extract ratings vectors for individual users, and construct generic feature matrices for individual movie classifiers.

3 Applying Logistic Regression

To apply logistic regression to the Netflix rating prediction problem, we consider each movie to be a separate problem requiring its own classifier. For any given movie m, our training data thus consists of all users who have rated m; more specifically, each training instance corresponds to a distinct user. This user's feature vector contains the ratings given by that user to every other movie in the dataset (0 if not seen). In addition, each movie rating is accompanied by a binary feature that indicates whether the user has seen the movie. Thus, supposing there are M movies in the database, a single feature vector contains (2M-1) features (including the intercept term), since we do not include the rating of the movie we are classifying as a feature. The corresponding label for each training instance is simply the rating given to movie m by this user. Additional details on feature definition, including normalization and scaling, are detailed in the section 5. First, however, we discuss the two types of logistic regression used in our work.

4 Regularized Logistic Regression Models

Unregularized logistic regression often overfits parameters to training data. Logistic regression finds the parameters θ that maximize the following expression:

$$\sum_{i=1}^{m} \log p(y^{(i)}|x^{(i)};\theta) - \alpha R(\theta),$$

where R(θ) is a regularization term (=0 for standard logistic regression). For L1 regularization, R(θ) is the sum of the norms of the components of theta; for L2, R(θ) is the sum of the squares of these components [1].

Because the L1 norm is not differentiable at zero [2], we cannot use simple gradient descent to optimize the L1-regularized objective function. Furthermore, because of the size of the Netflix training set, we cannot use Newton's method to run L2-regularized logistic regression (intractable computation of inverse Hessian), and gradient descent (implemented in Matlab) takes an extremely long time to converge. As a result, we eventually resorted to using pre-written packages to compute optimal parameters for the L1- and L2-regularized logistic regression problem. For L2 regularization, we used the LR-TRIRLS package [3]; for L1 regularization, we used a Matlab implementation of the IRLS-LARS method [4].

Although both L1 and L2 regularization serve to prevent overfitting, they result in vastly different feature weights. L2 regularization imposes a Gaussian prior on feature weights; as mentioned in the Introduction, the resulting weights are essentially all non-zero. L1 regularization, on the other hand, results in non-zero weight for less than 1% of the features. Our hope, therefore, is that L1-regularized logistic regression will outperform L2 on at least some subset of the complete data; the two can then be combined (i.e. using ENSEMBL) to create a universally better classifier. We initially hoped to also use L1 weights as a basis for constructing a Markov model that encodes relationships between movies, but the task of training L1-regularized LR on the Netflix dataset proved quite challenging in its own right. See "Future Work" for more information on Markov model construction.

5 Feature Design

As explained in section 3, in order to predict what a user *u* will rate some target movie *m*, we build a training set consisting of vectors of movie ratings for all users that have rated movie *m*. Each training example's feature vector consists of pairs of features <viewed, rating> for a user's rating on each movie: "viewed" is either 0 or 1, depending on whether the user rated the movie or not, and "rating" is a normalized, centered user rating (or 0 if the movie was not rated by this user).

To center the ratings, we tried several approaches:

1. User average: centers the rating around the user's average rating. The advantage to this approach is that it scales for user bias; for example, some users will tend to rate all movies relatively high, while others will rate on a more conservative scale. The

disadvantage of user-average centering is that some movies are “better” than others and thus get higher overall ratings.

2. Movie average: centers the rating around each movie’s average rating. This method takes care of movie bias, but does not account for user bias.

3. Weighted combination: (0.3 * MovieAvg + 0.7 * UserAvg).
   Centers the rating around a linear combination of both sources of bias; weights chosen based on empirical performance.

4. Adjusted movie average: (MovieAvg * UserAvg / AVG(UserAvg))
   Centers the rating around a movie average that is scaled by a user’s relative average rating. For example, if the user usually rates movies lower than other users, this approach adjusts the user’s rating for a given movie to a level that is proportionally below that movie’s average rating.

The rating for the target movie r_m is then centered around the same baseline and translated into a binary {1, -1} label, where label = 1 if $r_m > 0$, label = -1 otherwise. The feature vectors for all users who rated the target movie *m* along with their corresponding labels are then passed as input to L1-regularized logistic regression.

6 Rating Prediction

The output of the trained logistic regression classifier represents the probability that user *u* would “label” movie *m* “1” (where label $\in$ {1,-1}). In our case, label = 1 corresponds to user *u* rating movie *m* above the baseline rating (where the baseline is defined as described above). Using this probability, we can now predict the actual rating for this user-movie pair.

To compute the predicted rating, we consider the full distribution of movie ratings given by user *u* and compute an expected value for two sets of ratings: those above the baseline and those below it. More specifically, let e_1 = E[ratings below baseline] and e_2 = E[ratings above baseline]. We compute the predicted rating as:

$$\text{Predicted\_rating} = h(x)*e_1 + (1-h(x))*e_2$$

7 Testing Methodology

To compare L1- and L2-regularization for logistic regression, we generate L1 and L2 predictions for sets of movies with approximately 100, 500, 1000, 2000, and 5000 ratings (where each bin’s movie set contains ~4 movies). We also compute simple baseline predictions (e.g., predict a user’s rating to be average rating) for another basis of comparison.

For each movie, we hold out 10% of the data for testing and train a classifier on the remaining 90% of users who rated that movie. We then judge the accuracy of the ratings by computing the root mean squared error (RMSE) on the test set. Due to time and memory constraints, we were unable to compute RMSE for the full Netflix probe set (it takes approximately 3 hours to generate input features and train a classifier for a single movie with 2000 ratings). The RMSE values shown below are therefore estimates of the true test-set RMSE.

8 Results and Discussion

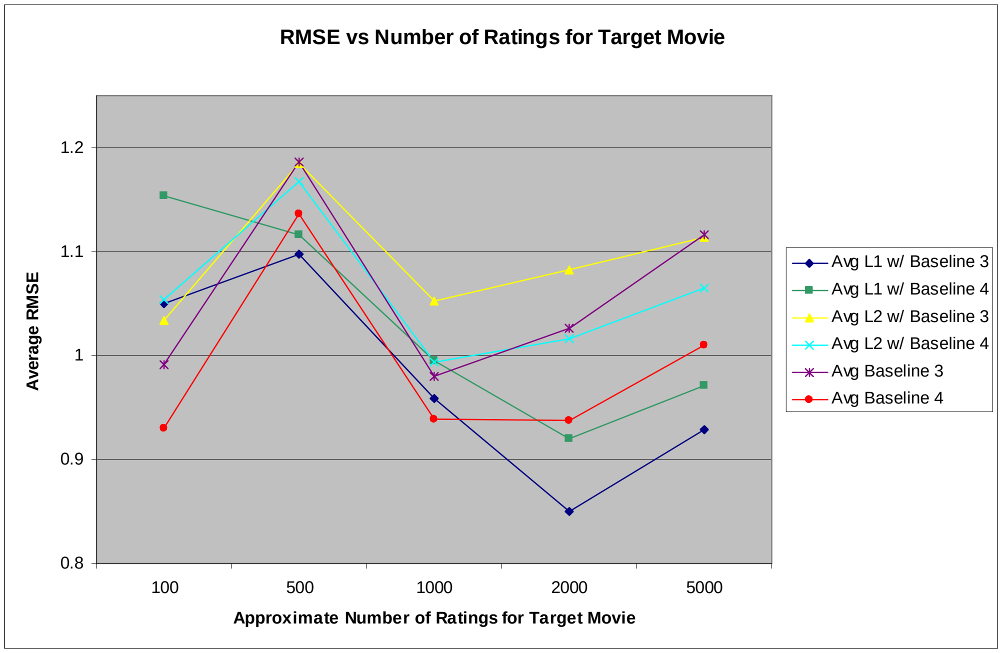

Figure 1: Average RMSE for movie prediction as the number of ratings increases

Figure 1 highlights several important trends. Note first that, as expected, a larger training set generally results in better performance. For example, the average RMSE for all classifiers for movies with $\geq$ 1000 ratings is 0.096 lower than average RMSE for movies with < 1000 ratings.

More importantly, however, L1 logistic regression starts to significantly outperform other classifiers when the size of the training set increases. On movies with 1000 ratings or less, L1 averages RMSE=1.035 (baseline 3), which is worse than the RMSE=1.002 achieved by the baseline-4 predictor. However, when the size of the training set is greater than 1000, L1 regression (baseline 3) averages RMSE=0.885, while the baseline-4 predictor yields RMSE=0.97. Thus, as we might expect, L1 performance increases dramatically as the training set size increases. Surprisingly, though, the L2-regularized logistic regression classifier does not exhibit similar behavior (Figure 2).

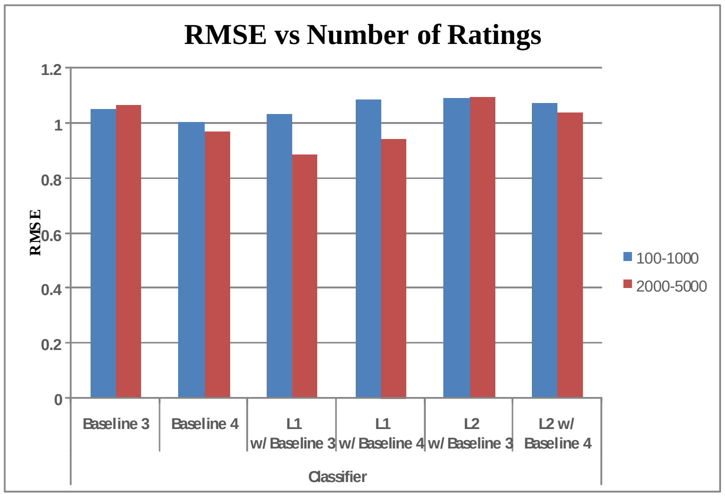

Figure 2: Average RMSE across classifiers for large and small training sets

Figure 1 also shows that L1- and L2-regularized logistic regression classifiers offer little or no benefit over simple baseline prediction methods for movies with very few ratings. Consequently, in a full-blown Netflix prediction system, we can avoid training LR classifiers for these sparsely-rated movies and instead rely on simple baseline classifiers. However, for movies with more ratings, L1-regularized logistic regression clearly offers significantly improved prediction accuracy. Furthermore, supposing L1-regularized logistic regression performs acceptably well on movies with 2000-5000 ratings, we can make the prediction task more tractable for movies with many (e.g. >10000) ratings by training on only a subset of the users who rated them.

9 Future Work

L1-regularized logistic regression worked surprisingly well for the Netflix movie rating prediction task. Unfortunately, our method of extracting features and training the L1 classifier are essentially infeasible for the full Netflix dataset. However, the feature extraction task can certainly be optimized (we currently use a Perl script to parse the raw data file), and the L1 training code (currently implemented in Matlab) can probably be ported to C++, which would result in orders-of-magnitude speed increase. Once we are able to train L1-regularized LR classifiers for all of the movies in the Netflix dataset, we can use ENSEMBL methods to combine the L1 classifier with other prediction methods (including the L2 classifiers). Furthermore, the L1 classifier itself can probably be tweaked further via changes to the normalization and probability-to-rating transformation steps.

Another direction we wish to explore, as mentioned in the “Regularized Logistic Regression Models” section, is the use of L1-trained feature weights as a basis for a Markov model. Consider a trained L1 classifier for some movie m; movies whose features were assigned non-zero weights can be viewed as “highly correlated” with m. We can thus use these feature weights to construct a simple Markov network, where nodes represent movies and each movie is connected to all other movies it is “related” to (note that, by adjusting the weight of the L1 prior, we can force a smaller number of non-zero weights if necessary). Edge affinities can also be computed as functions of the learned L1 weights. We can then run inference on this Markov network to compute the likelihood of each possible rating for a target movie, given ratings for a subset of the other movies in the network.

10 References

[1] http://ai.stanford.edu/~ang/papers/icml04-l1l2.pdf
[2] http://www.stanford.edu/class/cs229/proj/RaksheKumar-L1RegularizedLogisticRegression.pdf
[3] http://komarix.org/ac/lr/#LR-TRIRLS
[4] http://www.stanford.edu/~hllee/aaai06_L1logreg.pdf

11 Appendix

Table 1: Classifier RMSEs

Num Ratings	ID	Baseline3	Baseline4	L1 w/ Baseline 3	L1 w/Baseline 4	L2 w/ Baseline 3	L2 w/ Baseline 4
	2179	0.9179	0.8431	1.1254	1.1796	0.8359	0.853
	3344	0.6923	0.6626	0.8305	1.1276	0.7516	0.7682
	1677	1.0565	1.0332	1.147	1.197	1.226	1.2698
100	1957	1.3	1.183	1.098	1.11	1.3198	1.3258
	3616	1.271	1.225	1.2	1.2	1.2216	1.2229
	3971	1.217	1.175	1.133	1.128	1.1927	1.1688
	6036	1.116	1.049	1.069	1.144	1.0926	1.1181
500	6465	1.143	1.098	0.989	0.995	1.2318	1.1598
	3010	0.791	0.733	0.613	0.737	0.9202	0.8194
	7362	1.303	1.274	1.189	1.17	1.3378	1.305
	3943	0.9328	0.8713	1.1066	1.1659	1.0282	0.9686
1000	7243	0.892	0.877	0.925	0.907	0.9254	0.884
	14990	1.167	1.141	0.8322	0.9065	1.2552	1.25
	1260	0.925	0.877	0.92898	0.93415	1.0818	0.9456
	1293	1.291	1.012	0.8517	1.0206	1.1931	1.0792
	8793	0.834	0.825	0.8055	0.8394	0.9067	0.9033
2000	9070	0.916	0.835	0.8325	0.8967	0.9769	0.9019
	12297	1.258	1.047	0.9054	0.9858	1.1245	1.1268
	13477	1.03	1.006	0.99547	0.9662	1.1083	1.0462
	16794	1.207	1.025	0.9851	1.0765	1.1445	1.0809
5000	2981	0.969	0.96	0.8306	0.8579	1.0798	1.007
Avgs		**1.058547619**	**0.9882**	**0.971092857**	**1.025945238**	**1.093066667**	**1.057347619**

Table 2: Average RMSE for number of ratings

Num Ratings	L1 w/ Baseline 3	L1 w/ Baseline 4	L2 w/ Baseline 3	L2 w/ Baseline 4	Baseline 3	Baseline 4
100	1.05	1.15355	1.033	1.153	0.992	0.930
500	1.098	1.11675	1.185	1.116	1.186	1.137
1000	0.958	0.994975	1.053	0.995	0.979	0.939
2000	0.850	0.91947	1.083	0.919	1.027	0.938
5000	0.929	0.9716	1.114	0.972	1.116	1.009

Table 3: Average RMSE

Number of Ratings	Baseline 3	Baseline 4	L1 w/Baseline 3	L1 w/Baseline 4	L2 w/Baseline 3	L2 w/ Baseline 4
100-1000	1.0527	1.002	1.0355	1.088425	1.0903	1.07195
2000-5000	1.066	0.969	0.8853	0.942638889	1.096755556	1.037877778