Movie Rating Prediction

Nick Armstrong
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
narmstro@andrew.cmu.edu

Kevin Yoon
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
kmy@cmu.edu

Abstract

The Internet Movie Database (IMDB) is one of the largest online resources for general movie information combined with a forum in which users can rate movies. We investigate the extent to which a movie's average user rating can be predicted after learning the relationship between the rating and a movie's various attributes from a training set. Two methods are evaluated: kernel regression and model trees. Modifications to standard algorithms for training these two regressors lead to better prediction accuracy on this data set.

1 Introduction

The Internet Movie Database (IMDB) is one of the most popular movie websites on the Internet, providing extensive information about movies and allowing users to post comments on movies they have seen and even rate the film[1] on a scale of 0 to 5. Given the voluminous information pertaining to each movie, we conjecture that there exists a relationship between a movie's average user rating and some of its various attributes (which include *Director*, *Actors*, *Box Office Gross*, etc.).

If this assumption holds true, then a rating predictor of significant accuracy can be developed. This rating predictor could be of value to the average movie-goer by predicting a rating for movies not yet rated by other users, even including titles that have not yet come to theatres. Additionally, the film industry would find such a tool invaluable since it inherently provides a "formula for success" in terms of the expected user opinion of a film.

2 Problem Definition

We obtained a single list of approximately 1860 movies[2] and their associated attributes from the IMDB. They are each listed in **Error! Reference source not found.** and can be one of two types: "D" - nominal (discrete with no numerical order) or "N" - numerical (continuous or discrete with equal intervals between adjacent scale values).

We propose to compare the performance of two rating regressors on this data set, each built automatically via a machine learning algorithm. One will use kernel

[1] Rating data obtained from movielens.umn.edu. Other movie data obtained from imdb.com.

[2] Only 1860 titles at movielens.umn.edu were rated.

regression, an instance-based learner better able to approximate continuous output classes than traditional nearest-neighbor approaches. This will require the definition of an appropriate distance function to deal with the mix of attribute types, as well as other weighting parameters to optimize predictor accuracy.

The other regressor will be a linear model tree generated by a modified version of the m5 algorithm [1] [2]. In this modification, it will first split nodes on all nominal attributes based on the P-values of an ANOVA F-test; when all nominal attributes have been exhausted, it will perform multivariate linear regression at the leaves using the numerical attributes.

3 Related Work

3.1 Sequential Forward Selection

Test error can be quite sensitive to the presence of irrelevant attributes; furthermore, many learning algorithms are intractable if run on the full attribute set. Feature selection addresses both of these issues. This task is well-studied [3], so we chose a popular, simple, fast, and effective algorithm: sequential forward selection (SFS). SFS is a greedy heuristic technique that begins with an empty set of features, adds the "best" feature candidate to the selected set according to some "goodness" measure, and repeats this process until all features are chosen or the "best" feature has "goodness" below some threshold. In our case, we used the P-value from an ANOVA F-test on the candidate feature given the previously selected features are in the model [6]. The ANOVA (analysis of variance) F-test is a hypothesis test of H_0: no values of the feature affect the mean response vs. H_a: at least one value of the feature affects the mean response; the response is numerical and the features can be nominal or numerical (in the latter case, the feature is termed a covariate and the test is ANCOVA, analysis of covariance). The associated P-value measures the test's significance, the probability of seeing results at least this unlikely assuming H_0. Hence, the candidate feature with the most significant test was chosen to be added to the selected feature set.

3.2 Model Tree

A model tree consists of a standard decision tree over nominal variables but has linear models at the leaves rather than a discrete class. A common way to build a model tree is with the m5 algorithm [1] [2], which adds nodes based on maximizing the expected error reduction. Post-pruning is done to minimize the expected estimated error, and terms are dropped from the models at the leaves to minimize the product of the expected estimated error and a penalty term based on the number of parameters in the linear model. Our algorithm used the same steps as m5, with the exception that pre-pruning was done rather than post-pruning; the main difference was in our choice of measurement function to choose which nodes to add and which to prune. We chose an ANOVA F-test for this measurement, weighted by number of examples at that node, because it is more sensitive to maintaining degrees of freedom for good variance estimates in subsequent node choices and for regression at the leaves. This benefit is particularly important in our data set, since features such as *Director* tend to have high arity (up to a third the total number of records) and m5 would tend to select these, effectively data-starving the linear models at the leaves.

3.3 Kernel Regression

Regression techniques have been greatly researched over the years. Less work has been done in the area of heterogeneous distance functions (distances between points defined by both nominal and numerical attributes) applied to classifiers for discrete outputs. Finally, even less work seems to have been done on heterogeneous distance functions for real-valued outputs. In [5], a clever distance function called the Value Distance Metric (VDM) for nominal attributes is introduced, but it is applicable

only for discrete classes. To our knowledge, few people have developed methods for our specific task involving data consisting of both nominal and numerical attributes and continuous outputs. Those who have done similar work have been unable to achieve classification accuracy matching simpler instance-based learners [1].

Of course, a strategy like VDM could be applied to our problem if we chose to discretize the output class (e.g. create rating categories: 0.0-0.25, 0.25-0.50, etc.) but this approach loses information about the inherent ordering of the output. There is also the dilemma that fewer classes would lead to higher classification accuracy of the discrete classes, but the true accuracy would be hard-limited to the resolution of the discretization. We therefore decided to use regular kernel regression in combination with a normalized heterogeneous distance function with emphasis on parameter selection for optimal performance.

4 Proposed Methods

4.1 Data Set Preparation

We first parsed the MovieLens data set for ratings of all users on movies (using a Perl script), then took the mean over all users for a given movie to become our prediction variable Y. We then parsed the predictive feature set from the output of the IMDB query tools using another Perl script. We chose 32 attributes with the following characteristics: any textual data that was not nominal was excluded; only U.S. dollars were taken from monetary data; some list attributes were excluded (such as countries in which scenes were shot), while for others we only took the first entry (such as *Languages*); and for actors we chose the first 5 positions in the credits list (as attributes *Actor1*, *Actor2*, etc.), under the assumption that position in this list implies degree of stardom in the movie. We were careful to include even those attributes we thought would not be important, so as not to introduce our own biases, while excluding only those that would be difficult to process meaningfully.

The processed data was divided into a training set (95.6%) and a test set (5.4%).

4.2 Missing Data Values

Some of the movie records contained empty fields under the attributes *Budget*, *Box Office Gross*, and *Rentals Revenue*. When a film's budget information, for example, is unavailable, the field contains *NaN*. In order to deal with these records, the training set was preprocessed and all instances of *NaN* were replaced with the mean value of the attribute over the set. Test/query data points were also "repaired" with mean values from the training set.

4.3 Feature Selection with SFS and ANOVA

We used SFS to remove superfluous features that would otherwise cause overfitting and intractable computation. The selected features (those not removed) were then fed to the two regressor methods so that they could be compared on an equal playing field. Note that since this task itself is computationally expensive, we limited the feature set to only use the 32 singleton features (both nominal and numerical). Had we included just pairwise interactions, we would have generated an additional 496 features each with much larger arity; SFS required about 2 hours to complete with only 32 singleton features, so adding the interactions was essentially infeasible.

Our criterion for "best" candidate feature was the P-value of an ANOVA F-test of the candidate given that the selected features were in the model (lower P-values are better); the response of the model is mean user-rating. Note that when the model includes a numerical attribute (a "covariate"), the test is an ANCOVA. We continued SFS until the P-value of the best candidate feature was above .05.

4.4 Kernel Regression

Instance-based learners (IBLs) are practical learning approaches to deduce the behavior of complex target functions when the data is multivariate and numerous. Rather than estimating a global function, IBLs can approximate functions locally near a query point. Disadvantages of IBLs include a back-heavy computational cost since all calculations are performed at query time. There is, however, no loss of data since all training examples are considered for each query. To mitigate this computational load, we hybridize our regression algorithm with a **k** nearest neighbors approach so that local averages are not calculated over all data points.

Kernel regression is one of the most versatile types of IBLs, yet it remains an intuitively straightforward algorithm. It does, however, require the definition of a meaningful distance function that quantifies the degree of separation between two data points composed of both nominal and numerical attributes. We employ a heterogeneous distance function that considers the two separately. Given a query point $\mathbf{x_q}$ with *n* attributes, its distance from $\mathbf{x_i}$ in the training set **x** is given by

$$D(x_q, x_i) = \sqrt{\sum_{a=1}^{n} d_a^2(x_q, x_i)} \tag{1}$$

where the attribute distance function is defined as:

$$d_a^2(x_q, x_i) = \begin{cases} \frac{x_q(a) - x_i(a)}{\max_a(x) - \min_a(x)} & \text{if attribute a is linear} \\ Bool(x_q(a) == x_i(a)) & \text{if attribute a is nominal} \end{cases} \tag{2}$$

Hence, if the attribute is linear, the distance between $\mathbf{x_q}$ and $\mathbf{x_i}$ along the a^{th} dimension is the normalized difference of their a^{th} attributes. Normalization prevents overweighting of one attribute over another. If the attribute is nominal, we use the Hamming distance, which defines the distance between two points along the a^{th} dimension as 1 if their a^{th} attributes are different and 0 if they are the same.

The predicted rating is a weighted average of the ratings of its **k** nearest neighbors:

$$\text{Prediction} = \frac{\sum_{i=kneighbors} w_i y_i}{\sum_{i=kneighbors} w_i}, \text{ where } w_i = e^{-D^2(x_q, x_i) / K_w^2} \tag{3}$$

is the Gaussian weighting factor and $\mathbf{K_w}$ is the kernel width.

4.5 Model tree

As previously stated, a model tree consists of a standard decision tree over nominal variables but has linear models at the leaves rather than a discrete class. Model trees can handle mixed nominal and numerical features and predict a continuous output.

The basic idea to learn the tree is to first build a decision tree from the nominal features, then to perform linear regression at the leaves with the numerical features. To choose the "best" node to add to the decision tree, we use the node with the most significant (smallest) weighted sum of P-values from performing an ANOVA F-test on each of its branches. If the best node fails a Chi-squared test or has too few examples to perform a linear regression for the majority of its nodes (2 examples), then we don't add it and stop adding nodes. This pre-pruning reduces overfitting and ensures useful models at the majority of the leaves.

After the decision tree is built, we perform a linear regression at each of the leaves of all numerical features on mean user rating. To further combat overfitting, we then use an ANOVA F-test to do backwards elimination of numerical features from these linear models. This may remove none, some, or all of the variables (leaving only the intercept term, the mean of the examples).

In the end, each leaf comprises a linear model (which may use different sets of numerical feature from the other leaves); to predict the rating for a new example, we descend the tree according to that example's nominal features and then apply the linear model at the leaf to its numerical features.

5 Experiments

With the following experiments we hope to answer these questions:
1) Which features have the greatest effect on user rating?
2) Can we achieve with our regressors higher prediction accuracy than a control predictor (predicting the mean rating of the entire training set for every query)?
3) How does kernel regression compare in prediction accuracy to the model tree?
4) Do $\mathbf{K}_\mathbf{w}$ and $\mathbf{k}$ have any bearing on prediction accuracy?
5) Which features most affect user rating?

To answer these questions, we divided the data into a training set (94.6%, or 1760 records) and a test set (5.4% or 100 records). We performed feature selection and trained the regressors using the training set then tested the regressors on the test set using the error metric mean relative absolute difference:

$$\text{Test Error} = \frac{1}{\|\{test\}\|} \sum_{x \varepsilon \{test\}} \frac{|predicted\_rating(x) - rating(x)|}{rating(x)} \qquad (4)$$

5.1 SFS

Nine features were selected by SFS (5 nominal and 4 numerical) and are shown below in Table . The order here is indicative of the feature's significance.

Order of Addition								
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
isDrama	Year	Director	Box Office Gross	Budget	isHorror	Costume Designer	Writer	Actor

Table 1: Training attributes

We also ran SFS with a different selection criterion – maximizing the expected reduction in MSE. This criterion initially selected a similar feature set, with *isDrama* still being first, but since this criterion does not take into account already-selected features, it soon began to choose different features from our ANOVA criterion. In general, it tended to select lower arity features first than did ANOVA.

5.2 Control Predictor

Test errors for both regressors are compared against the control predictor, which simply predicts the mean user rating over all movies in the training set for any new query. Looking at the distribution of ratings in the training set, we can see essentially a normal distribution but with light tails (Figure 1). Indeed, the control predictor achieves very good accuracy, with only 16.07% error.

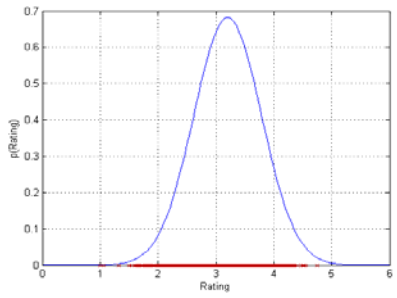

Figure 1: Distribution of ratings in training data

5.3 Kernel Regression

Parameters **k** and $\mathbf{K_w}$ were selected manually at first, then automatically via 10-fold cross-validation (see Figure 2) on the training set using the error metric (4).

Parameters chosen manually: $\mathbf{K_w} = 2$, $\mathbf{k} = 17$; Test error: 15.10%

Parameters chosen with CV: $\mathbf{K_w} = 1$, $\mathbf{k} = 11$; Test error: 14.11%

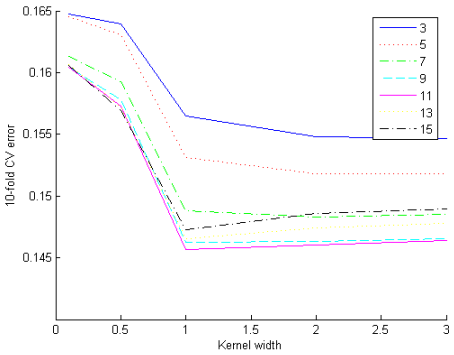

Figure 2: 10-fold cross-validation error for various numbers of nearest neighbors k

The kernel regressor achieves almost 2% less error than the control predictor; and as can be plainly seen in Figure 2, optimal selection of **k** and $\mathbf{K_w}$ have critical bearing on this performance.

In an effort to further increase the prediction accuracy, we also considered an attribute weighting scheme. We can modify (1) to get the new distance function:

$$D(x_q, x_i) = \sqrt{\frac{\sum_{a=1}^{n} w_a^2 d_a^2(x_q, x_i)}{\sum_{a=1}^{n} w_a^2}} \qquad (5)$$

where w_a is a weight associated with the attribute a. Before, an equal weighting scheme was used where $w_a = 1$ for all attributes.) Initially, weights were selected to be equal to the associated normalized P-values as were determined during feature selection, since P-values are indicative of the effect of the attribute on the rating. An array of linear weights, proportional to the order in which the attribute was added to the used feature sequence was also used (i.e., the first attribute added during feature selection, *isDrama,* had the highest weight and the last attribute, *Actor1,* had the lowest).

Unfortunately, both attribute weighting approaches resulted in no improvement in accuracy, although the resulting predictor still outperformed the control predictor. The lack of improvement could possibly be attributed to non-optimal feature selection. A more complex feature selection algorithm might be able to extract pair-wise significances of attributes that could lead to more accurate predictions.

Another possible reason attribute weighting did not work is that the approaches we attempted simply did not reflect the true weights of the attributes. Therefore, a brute force search over 9 discretized weights was performed – with the optimal **k** and $\mathbf{K_w}$ parameters found previously – by minimizing the test set error (simultaneously performing cross-validation would have taken an unreasonable amount of computing time on the machines we had available). Each of the 9 weights could have a value of 0.33, 0.66, or 0.99 (again, these weight values were limited by computational costs). The minimum test set error achieved was 13.88% with a weight sequence that had $w_{director} = 0.99$ and $w_a = 0.33$ for all other attributes. A small improvement in the test set error is achieved with these new weights. Better performance might possibly be achieved if weights are determined simultaneously with the **k** and $\mathbf{K_w}$ parameters.

5.4 Model tree

The training set produced the tree below (with trimmed linear models, using hallucinated means at each leaf). The crossed-out numerical attributes were trimmed from the full linear models.

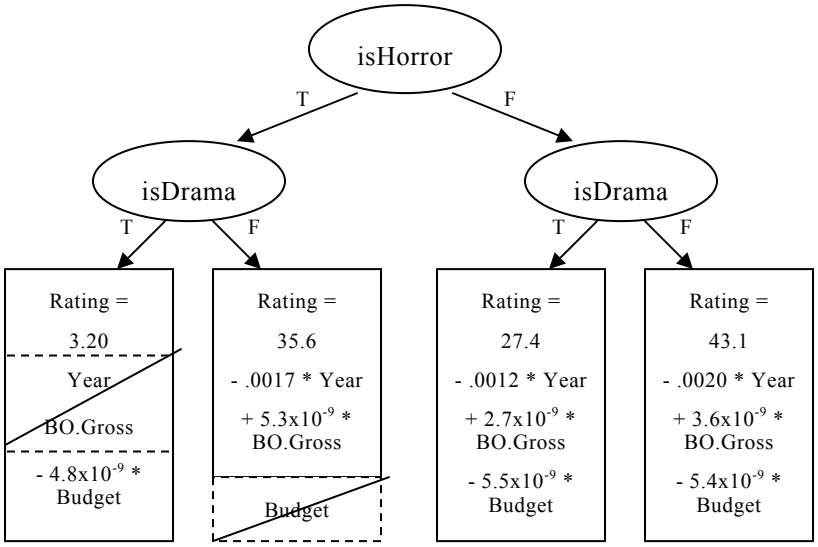

Figure 3: Model tree

Test errors for different model tree strategies are shown below in Table . Small improvements were expected by more intelligently hallucinating missing data. The basic technique was to use the mean of the entire training set; using just the mean of the remaining training data at the leaves, both during training and testing, was also attempted, but did not result in any significantly improved accuracy (probably these techniques just increased overfitting of the training set).

	Full Linear Models	Trimmed Linear Models
Hallucinated over all training data	14.50%	14.40%
Hallucinated at each training leaf	14.61%	14.71%
Hallucinated at each leaf in testing	14.58%	14.65%

Table 2: Test error for various model tree strategies

6 Conclusions

We have shown that it is possible to obtain approximations of the underlying relationship that describes mean user rating as a function of a movie's attributes. Using kernel regression we were able to achieve prediction accuracy with 14.11% deviation from the true rating, outperforming the control predictor accuracy which of 16.07% error. The model tree exhibited test error of 14.40%, which is slightly worse (though not significantly) than the kernel regression, but still significantly better than the control predictor. Significantly, here, means statistically different according to a two-tailed t-test with $\alpha = .05$. Of our two regressors, the best relative reduction in error (from the control predictor) was achieved by kernel regression, with a relative reduction of 12.19%.

Given the nature of the data we are trying to learn, it is not surprising that a more significant increase in prediction accuracy (over predicting the training mean) was not achieved. Many of the reasons people like or dislike a movie depend upon factors that are cannot be captured (or are subjective), such as how intriguing a

plotline is; or are difficult to encode as a feature, such as the plot description. As another example, the physical attractiveness of the starring actor(s) can play a huge role in the success of a film. Moreover, the entertainment industry is constantly changing with new people and new technologies, as well as changing qualities that make a film successful. One thing many successful films have in common is that they are usually quite unique – and not necessarily only in terms of storyline – thereby making it difficult to observe any pattern in successful films.

Additionally, people who rate movies not only base their judgments on completely different factors, making the entire process highly stochastic; but also, those who take the time to rate a film are likely to have a strong opinion of it one way or the other, leaving the average viewer under-represented in a response-biased dataset. Other biases likely exist as well; e.g., although older movies tend to be rated higher, perhaps it is because the only old movies that were chosen and rated were the "classics," and no "flops" from 1959 were chosen to be viewed by modern audiences. It is possible that no higher accuracy can be achieved due to these biases and user caprice.

Feature selection, though, yielded valuable insight about which features of a movie are important to its mean user rating. It appears that whether the film is a drama or not is the most significant factor; if it is, the rating tends to be higher. Next most important is the year of release, with older movies typically receiving higher ratings. Other relative importances can be read directly from the SFS output. Notably, only the first-listed actor in the credits was selected, and this feature is the very last chosen by feature selection (thus the least important of those chosen). Intuitively, we know that the actors in a movie are very important to its rating; perhaps a better strategy exists that ignores the order of the actors in the credits but still keeps the feature set tractable.

Future work in this vein should include better encoding the movie's features (perhaps even parsing semantic textual data such as plot description, but more importantly encoding a non-ordered list of actors); directly comparing the model tree training algorithm against m5; more methodically choosing weights on the attributes for kernel regression; and using a more optimal feature selection algorithm (using a better heuristic and examining feature interactions). Perhaps user ratings could be taken from IMDB itself rather than from MovieLens, which could give many more training examples. To combat computational costs and eliminate associated simplifications/shortcuts, the algorithms should be parallelized and run on a cluster or supercomputer.

References

[1] Quinlan, J.R. (1992) Learning with Continuous Classes, *Proceedings AI'92 (Adams & Sterling, Eds)*, pp. 343-348. Singapore: World Scientific.

[2] Quinlan, J.R. (1993) Combining Instance-Based and Model-Based Learning, *Proceedings ML'93 (Utgoff, Ed)*, San Mateo, CA: Morgan Kaufmann.

[3] Dash, M. & H. Liu (1997) Feature Selection for Classification, *Intelligent Data Analysis*, pp. 131-156. Singapore: Elsevier Science.

[4] Wang. Y. & I. Witten (1997) Inducing Model Trees for Continuous Classes, *Department of Computer Science, University of Waikato*. New Zealand.

[5] Wilson, D.R. & Martinez, T.R. (2000) An Integrated Instance-based Learning Algorithm. *Computational Intelligence,* vol. 16, number 1, pp. 1-28. Cambridge, MA: Blackwell Publishers.

[6] Rutherford, A. (2001) Introducing ANOVA and ANCOVA: a GLM Approach. Thousand Oaks, CA: SAGE Publications Ltd.

Appendix A: Movie Attributes

	Attribute	Type	Description	Arity
1	Title	D	-	1860
2	Year	N	Year film was released	
3	Director	D	-	989
4	Writer	D	-	1431
5	Editor	D	-	730
6	Cinematographer	D	-	620
7	Production Designer	D	-	663
8	Costume Designer	D	-	661
9	Country	D	Country of production	26
10	Production Company	D	-	800
11	Color	D	-	2
12	Sound	D	-	55
13	Running Time	N (min)	-	-
14	Language	D	-	11
15-20	Actor1 - 5	D	Principal cast, ordered as in credits	927-1594
21	isAdventure	D (T or F)	Genre	2
22	isComedy	D	Genre	2
23	isDrama	D	Genre	2
24	isFamily	D	Genre	2
25	isFantasy	D	Genre	2
26	isHorror	D	Genre	2
27	isRomance	D	Genre	2
28	isSciFi	D	Genre	2
29	isThriller	D	Genre	2
30	Budget	N ($USD)	-	-
31	Box Office Gross	N ($USD)	-	-
32	Rental Revenue	N ($USD)	-	-
	Average User Rating	N	Range: 0.00 - 5.00	-