CSCI 181 / E-181 Spring 2014 Practical 2

Kaggle Team "No Comment"

David Wihl
davidwihl@gmail.com

Zack Hendlin
zgh@mit.edu

February 27, 2014

Warm-Up

Baseline

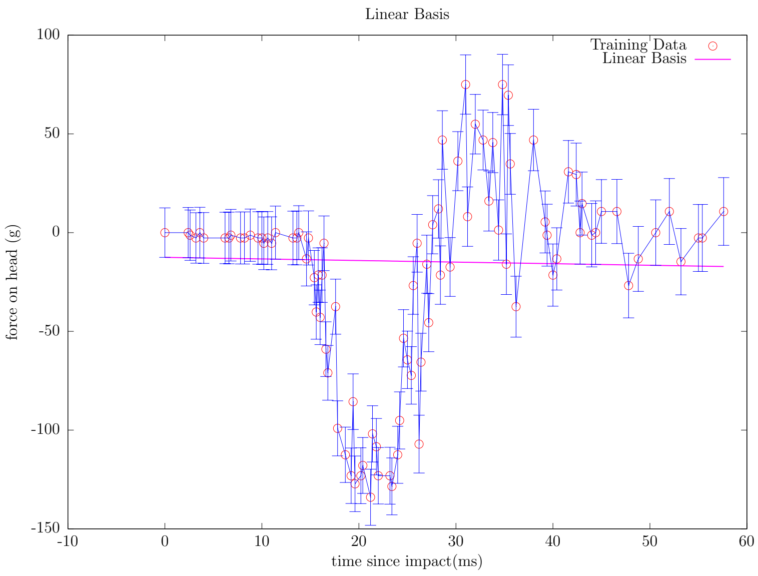

Figure 1: Warmup: Linear Basis

As a baseline, we first created a simple linear gradient descent with a flat slope and intercept.

We also used a polynomial basis, iterating with polynomials from n^2 up to n^{12} and selecting the lowest error. Unsurprisingly, n^{12} had the lowest error rate, but is likely highly overfit.

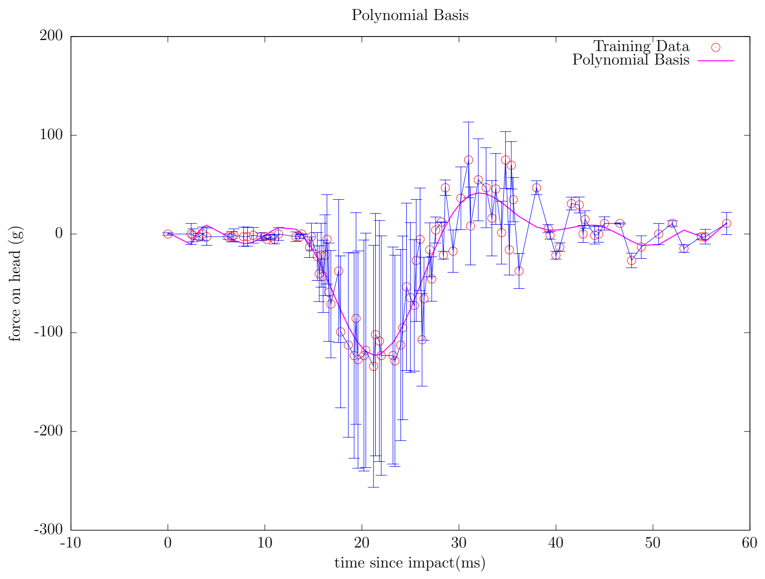

Figure 2: Warmup: Polynomial Basis n^{12}

Bayesian Linear Regression

Using Gaussian Likelihood and Prior, we solved for W using Moore Penrose.

$$W_{ML} = (\Phi^T \Phi)^{-1} \Phi^T t \tag{1}$$

This was simple to implement, especially in Octave/Matlab. However, without normalization the error rate was close to the baseline linear basis and significantly worse than the polynomial.

Locally Weighted Linear Regression

Locally Weighted Linear Regression (LWLR)[1] provided the lowest cost overall and a smooth fit to the data without overfitting given the profile of this dataset. A variety of K values were attempted. 0.001 never converged. Values from 0.5, 1.0, 5.0 and 10.0 did converge with 1.0 seemingly providing the best balance between fit and smoothness.

LWLR is an expensive operation. Since the dataset here was small and did not match a typical straight line or polynomial pattern, it was appropriate to attempt LWLR.

[1] *Machine Learning in Action* by Peter Harrington. © 2012 ISBN 978-1617290183

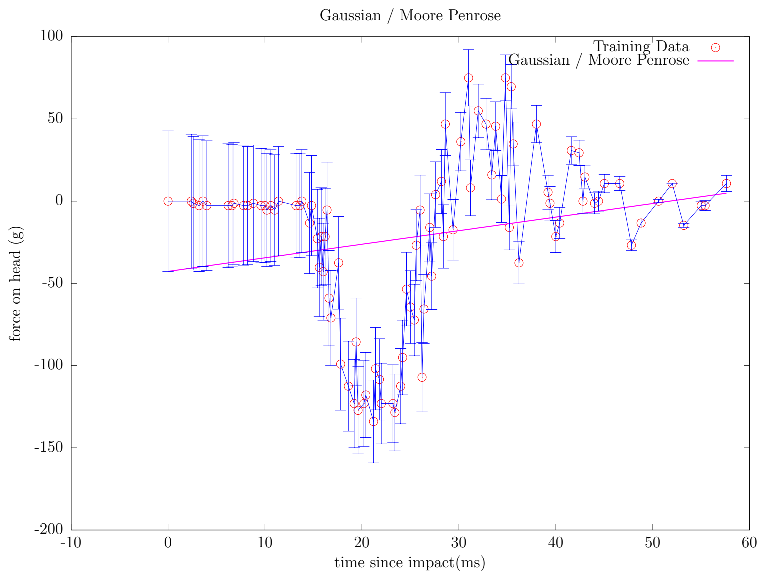

Figure 3: Warmup: Gaussian

Warmup Summary

Across all basis functions, overall error rate was calculated by sum-of-squares:

$$J = \frac{1}{2N}\sum_{i=1}^{N}(y_i - t_i)^2 \tag{2}$$

The following table summarizes our results. LWLR was reasonably simple to implement and provided the lowest cost. For this particular data set, it would be our basis function of choice.

Basis	Lowest Error
Linear Basis	1293.0
Gaussian Basis	1187.7
Polynomial Basis	211.9
LWLR Basis	185.6

Predicting Movie Opening Weekend Revenues

Preliminary Data Analysis

The training set consists of movie metadata and textual movie reviews. From the sample code provided in the problem, the initial set of features has a classic $P > N$ problem of too

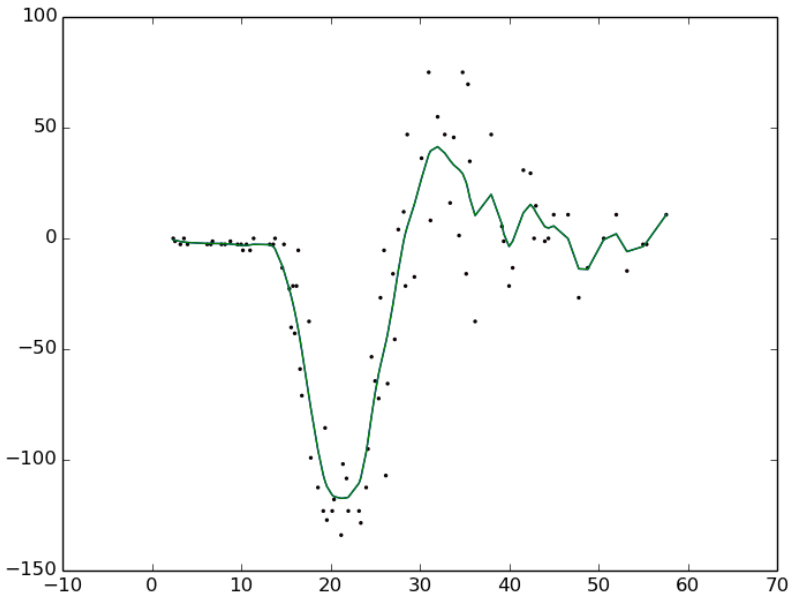

Figure 4: Warmup: Locally Weighted Linear Regression $K = 1$

many dimensions (105403) for too little data (1147). This is mostly due to the unigrams converting each word of each movie review and description into a different dimension.

Eliminating the unigrams as a first step significantly improved classification results and reduced dimensionality to only(!) 11276 dimensions.

Using Cross-validation

To quickly evaluate the regression algorithms, we build a simple cross validation set, using 10% of the data and ten folds. This enabled us to track J_{cv} vs J_{train}. By measuring the learning curve,[2] we could see our algorithms' progression.

Using multiple test runs, examining error rates and looking at the resulting weighting values, we were able to which dimensions had value.

Approaches considered

We considered two approaches:

[2] *The Elements of Statistical Learning* by Hastie, et al. © 2009 ISBN 978-0-387848587

Feature Engineering

We tested a number of approaches to both add features, and to remove features which could lead to overfitting

(1) Exclusion of unigrams. Highly dimensional unigrams hurt model performance.

(2) Further reduction in dimensionality by examination of generated weights. For example, in the original sample code, every date (including year) was a new feature. We transformed this into month of the year, but there was no change in weighting or error rates. So this feature was eliminated.

(3) Testing of positive and negative wording in reviews. We recognized that while all words in a review may cause overfitting and result in statistically insignificant weight for an individual word (due to the small N in the data), some words in reviews could contain particularly high information content. As such we tested two features:

a 'good' list : ["good", "great", "excellent", "seeing", "superb", "must-see", "fantastic", "credible", "best"]

a 'bad' list : ["bad", "slow", "boring", "horrible", "waste", "poor", "lazy","worst"]

Unfortunately, that didn't improve our model's performance.

(4) Creation of squared values for all quantitative variables. This did not help model performance but did lead us understand that "number of screens" a movie shown on was quite predictive. We then tuned the parameter for the exponent of "number of screens" (from 2.0 to 5.0) and found 4.3 to maximize our performance on two holdout samples – this helped our performance gains in the model significantly.

(5) Creation of logs for all quantitative variables on the basis that large numbers (e.g. budgets) may have non-linear impacts on the revenues of a movie. This did not help the model's performance.

(6) Threshold values as binary 1/0 values for if a film's budget was above a particular value, and if the number of theaters a movie shown in was above a particular value. We ran sensitivities to various values on the logic that there can be 'step functions' insofar as a movie with a very low budget may not be sufficiently high quality, but anything about X may be sufficient). This did not help the model's performance.

By examining weighting values, error values including outliers, cross-validation rates, we concluded that only two dimensions had any significant value: *number of screens* and to a lessor extent *production budget.*

Selection of fitting technique

We evaluated (1) ordinary least squares linear regression, (2) lasso regression (which penalizes the absolute value of regression coefficients), and (3) ridge regression (which penalizes the size of the regression coefficients). Lasso and ridge were considered to help with variable

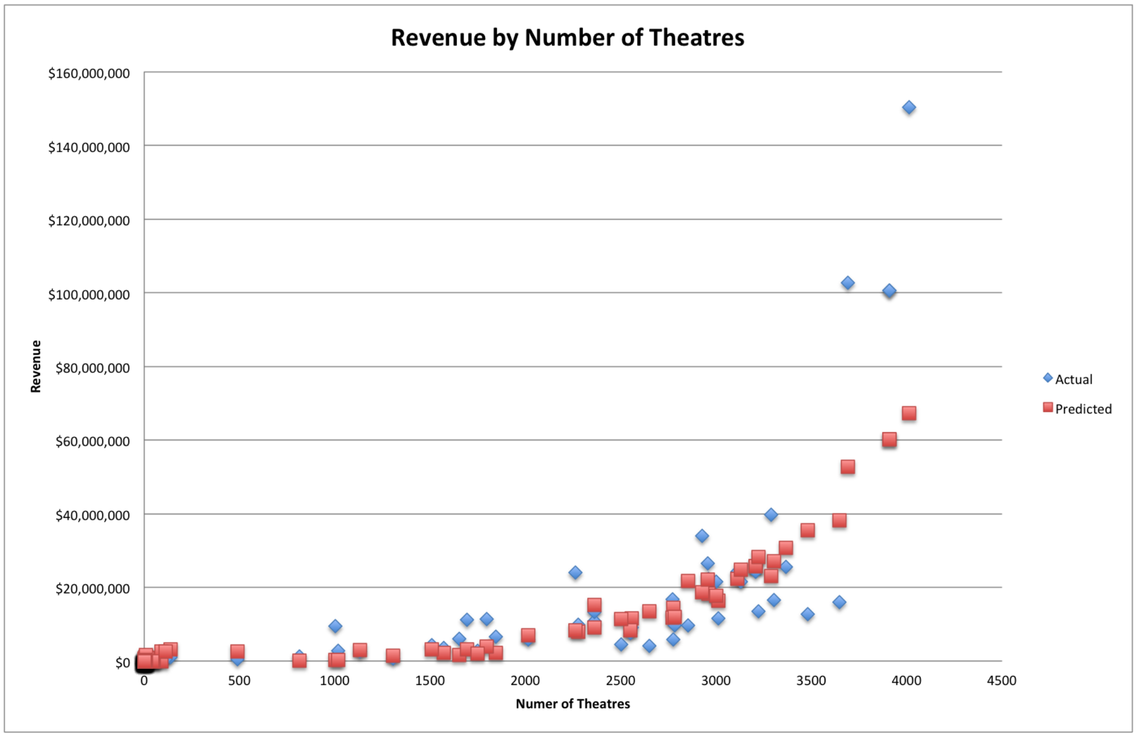

Figure 5: Actual vs Predicted by Number of Screens

selection issues which occur with a extremely large set of potential variables for inclusion, as was the case with unigrams.

Exploratory Data Analysis

By Feature Engineering, we were able to pare the dataset down to a very manageable 1146×2, well within the bounds of a simple tool like Excel.

The largest errors were on the most successful movies. Production budgets $> \$50,000,000$ were deemed a "blockbuster" type. Upon detailed analysis of the training data, we needed to apply a correction of 0.85 to this different category of movie. Given the large production budget, a studio can produce only a few blockbusters per year so by definition they are outliers compared to the majority of the movies. While our correction for large budget films decreased the overall error rate somewhat, it did not have as dramatic as effect as we'd hoped. Given most of the big budget films had lower openings relative to smaller budget films, we assume the studios were generally also disappointed.

Conclusion

The key learning from this practical is that even relatively simple techniques (such as linear regression) can be extremely powerful if appropriate features are selected.

Feature engineering ended up being key to getting reasonable results from the linear regression. For instance, introducing polynomials of numeric information was important to capturing trends in the data. We also had to grapple with how to avoid overfitting (alas, unigrams on such a small training dataset let to significant over-fitting and had poor-out-of-sample performance), and how to represent text in a meaningful way quantitatively (something which , despite trying NLTK and 'good' / 'bad' lists for the text of reviews, we never quite got much signal from).

We found that what we excluded was just as important as the features we included. Poor features radically overfit and gave poor out-of-sample performance. Poor features also lacked a reasonable interpretation (e.g. what does the count of the non stopword unigrams in a movie review really mean?). Another key learning was that more sophisticated algorithms like ridge and lasso are not necessarily better as they involve tweaking normalizing values.

Interestingly, feature engineering consequently required us to be more thoughtful about the problem as we sought other ways to get signal from our data.