Alice Wang
Daniel Aranki

Behavioral Data Mining
Homework 2:
Linear regression at scale

Spring 2012

Introduction

We consider the book reviews sub-dataset from the Multi-Domain Sentiment Dataset collected by Dredze et al [1]. This sub-dataset (to which we will refer in this paper simply as "dataset") includes about 1 million book reviews from Amazon.com with their corresponding book ratings on a scale of 1-5 excluding the rating 3 for being neutral.

In this report we describe the implementation considerations, the solution techniques and the results of a linear regression at scale to the dataset described earlier. We also compare these considerations and results among different design options that we considered.

In the first section we describe the basic model that was used. Then we discuss the performance boundaries of running such a large-scale job on a single machine and the implementation details we used in order not to run into such boundaries. At last, we give a summary of performance for our proposed solution.

The model

We apply linear regression on the numerically labeled book reviews. To be more precise, we need to define a features vector for each data point that will be carried into the linear regressions.

For that, we first collect all unigrams in all reviews texts (in the dataset) as our dictionary. Each unigram is then given a distinct token id. Then, for each data point pair of review/rating $(\hat{X}_i, Y_i)$ respectively, we consider the counts of these tokenized unigrams in the specific review as features in our data vector $\hat{X}_i$, normalized by the length of the review. Therefore, the linear regression model can be written as follows:

$$Y_i = \hat{\beta}_0 + \hat{X}_i\hat{\beta} \tag{1.1}$$

Where $\hat{\beta}$ is the linear weights vector and $\hat{\beta}_0$ is a biasing constant. If we define $X_i = \begin{bmatrix} \hat{X}_i & 1 \end{bmatrix}$ and $\beta^T = \begin{bmatrix} \hat{\beta} & \hat{\beta}_0 \end{bmatrix}$. Equation (1.1) can be rewritten as

$$Y_i = X_i\beta \tag{1.2}$$

Given N data points, we can define the matrix $X^T = \left[(X_i)^T \right]_{i=1}^N$ and the ratings (column) vector $Y = [Y_i]_{i=1}^N$. We can rewrite equation (1.2) as

$$Y = X\beta \tag{1.3}$$

In order to solve for β we define an error metric as the residual sum of squares (the L_2 error function) and aim to minimize that error

$$RSS(\beta) = \|Y - X\beta\|_2^2 = (Y - X\beta)^T (Y - X\beta) \tag{1.4}$$

Then by differentiating $RSS(\beta)$ and setting the derivative to zero, the closed form solution for β is (in case the matrix $X^T X$ is invertible)

$$\beta = \left(X^T X\right)^{-1} X^T Y \tag{1.5}$$

Then, given a new unlabeled data point $\tilde{X}$, we can invoke β from equation (1.5) into equation (1.2) to predict the rating of that review.

Performance boundaries

The dictionary of all words that appeared in any review text contains about $1/2$ million unigrams. Therefore, if we use 90% of the data set as training data, the matrix X is (approximately) of size $0.9M \times 0.5M$. This implies that the matrix $\left(X^T X\right)$ is (approximately) of size $0.5M \times 0.5M$ (regardless of the size of the training data). It is also noteworthy to mention that although the matrix X is sparse, the matrix $\left(X^T X\right)$ is not as sparse. Trials to invert this matrix (under sparse representation) failed to converge after 2 days of computation and were aborted.

In the following sub-sections we discuss two different possible solutions to work around this machine performance boundary and we evaluate their performance.

Feature selection & exact solution

Generalizing the observation from the last section, given $\mathbb{F}$ features, the size of the matrix $\left(X^T X\right)$ is $\mathbb{F} \times \mathbb{F}$. Thus, if we limit $\mathbb{F}$ such that inverting the matrix $\left(X^T X\right)$ is tractable, we can compute exact solutions for β using equation (1.5). The technique we used to select the $\mathbb{F}$ features was mutual information (MI).

We calculated 4 MI scores for each feature, one score per class. For each feature, the final MI score was considered to be the maximum score among all 4 scores. This is analogous to picking the features according to their best ability to discriminate between one specific class and the rest of the classes. Table 1 summarizes the verification data ROC area under the curve (AUC) (for rating>3 versus rating<3), root mean square error (RMSE) and the 1% lift scores for different values of $\mathbb{F}$ that were chosen using MI. We had to limit ourselves to these values of $\mathbb{F}$ because higher $\mathbb{F}$ values failed to converge in a 1-day window.

Number of Features	AUC	RMSE	1% lift score
20,000	0.50151	0.50426	0.8316
40,000	0.50286	0.65586	0.8598
60,000	0.49672	0.83246	0.7892
80,000	0.49674	0.86746	0.7575

Table 1: Performance in the case of exact solution for selected sizes of features

The results in Table 1 are not satisfying. We suspect the reason to be that we did not select enough features. The reason for this belief is that by looking at the feature vectors, we can see that since we "pruned" most of the features, some reviews are left with zeros in all features (since all its words are not in the selected features set). Therefore, a need to increase $\mathbb{F}$ arises but under the boundaries of single-machine performance. From the limitations mentioned above, for higher values of $\mathbb{F}$, we will not be able to calculate exact solutions and will have to solve for β differently.

Feature selection & (stochastic) gradient descent

It is easy to verify that $RSS(\beta)$ as defined in (1.4) is a convex function. Therefore (stochastic) gradient descent is one possible approach to solve the optimization problem $\beta^* = \min_\beta RSS(\beta)$. Table 2 summarizes the performance of the linear regression, in terms of ROC AUC, RMSE and 1% lift scores for different features sizes. In this approach too we still use mutual information as our feature selection technique.

Number of Features	AUC	RMSE	1% lift score
100,000	0.66351	0.00366	2.3994
120,000	0.72322	0.00382	0.7892
130,000	0.75574	0.00367	1.0880
140,000	0.75189	0.00375	6.2315
150,000	0.82743	0.00346	2.1313
160,000	0.89812	0.00300	10.7139
170,000	0.95982	0.00272	52.8770
250,000	0.94645	0.00289	46.3832

Table 2: Performance in the case of gradient descent solution for selected sizes of features

Training the regression for 250,000 features took significantly more time than with 170,000. We believe that with 250,000 features we could get to the same performance of 170,000 features, if not better. Although, because of the time and resources constraints, we decided to not optimize for 250,000 features more than that and move to try other techniques that will be described in later chapters in this report.

Figure 1 shows the training RMSE vs iteration during the gradient descent runs and the different ROCs for different sizes of features

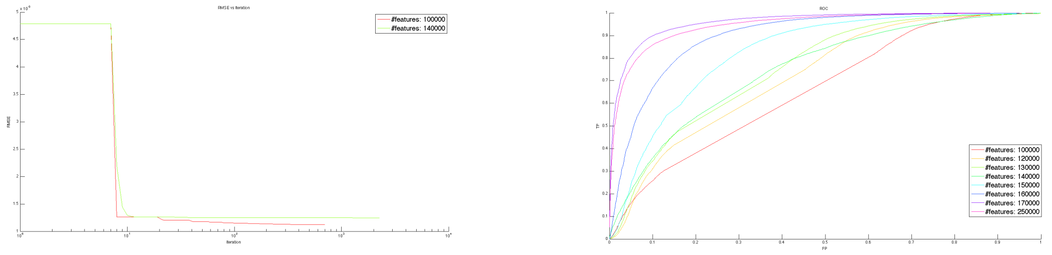

Figure 1: Training RMSE vs Iteration and ROC plots for selected feature sizes.

Lasso Regularization

It is of our interest to have a "constrained" weight vector β. Extraordinary large weights for certain terms may cause the following two effects *(i)* Two reviews that differ in one high-weight term may have different ratings although they are essentially the same and *(ii)* high weights may drive predicted ratings in a long range (relative to 1-5) and therefore RMSE will take a hit. Thus, we chose to penalize for the norm of β by adding a cost for the L_2 norm of β. This approach is known as the Lasso Regularization and yields the following error function

$$Error_{Lasso}(\beta) = \frac{1}{N}\|Y - X\beta\|_2^2 + \lambda\left\|\hat{\beta}\right\|_2 \tag{3.1}$$

Where $\hat{\beta}$ is taken from (1.1) (i.e. without the $\hat{\beta}_0$). The reason we don't penalize for large $\hat{\beta}_0$ is that the term $\hat{\beta}_0$ is a biasing factor that is equally applied to all data-points and is not affected by any terms in the review.

A question, though, about choosing "good" values of λ arises. Intuitively, λ incorporates the relative penalty weight of having a "large" β to the penalty of squared error in the training data. Since we have previous results of training data RMSE and norms of β (which is very close to the norm of $\hat{\beta}$), we can apply the following heuristic of picking λ (using the previous solutions from feature selection & gradient descent)

$$\lambda \approx \frac{\frac{1}{N}\|Y - X\beta\|_2^2}{\|\beta\|_2} = \frac{\frac{1}{N}RSS(\beta)}{\|\beta\|_2} \tag{3.2}$$

This value will "equally" penalize for "big" β's as for root mean squared errors. Half of this value will penalize root mean squared errors twice as much as it penalizes "big" β's.

The error function in (3.1) is a sum of two convex functions and thus is a convex function itself. We used (stochastic) gradient descent to minimize this error function for various features sizes and λ values. The results are summarized in terms of ROC AUC, RMSE, 1% lift scores and $\|\beta\|_2$ values before and after regularization in Table 3.

Table 3 shows that for "low" values of λ, compared to the suggested value in (3.2), the solution with the Lasso regularization approximately coincides with the solution without penalizing for the norm of β. On the other hand, for "large" values of λ, compared to the suggested value in (3.2), the solution tends to converge to β's that are very small (and in extreme cases, $\beta = 0$ because the penalty for any deviation from that point is way larger than the decrease in the root mean squared error). In the case of $N = 170{,}000$ and $\lambda = 5 \cdot 10^{-7}$ we see pretty good results with relatively small norm of β.

Number of features	λ	AUC	RMSE	1% lift	$\|\|\beta\|\|_2$	
					w/ Lasso	w/o Lasso
100,000	8e-8	0.66407	0.00370	1.7761	336	562
130,000	3e-7	0.75942	0.00345	6.7431	251	802
	6.3e-8	0.76108	0.00355	1.8443	591	
160,000	4e-7	0.87402	0.00317	16.1927	251	1401
170,000	1e-7	0.95842	0.00273	51.8631	648	669
	5e-7	0.93094	0.00302	40.0835	199	
	1.36e-6	0.87033	0.00333	27.1348	69	

Table 3: Performance in the case of gradient descent solution with Lasso regularization for selected sizes of features and values of lambda

Figure 2 plots the Training Error vs iteration and ROCs for the states reported in Table 3

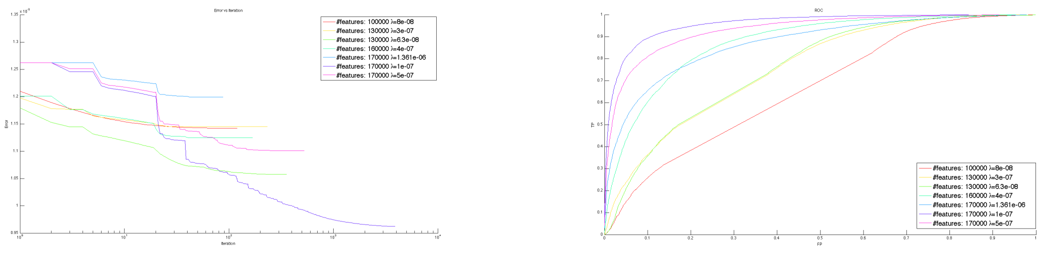

Figure 2: Training Error vs Iteration and ROC plots for selected feature sizes and values of lambda.

Strongest terms

We'll be considering the solution with 170,000 features. Table 4 lists the strongest 30 positives and Table 5 lists the strongest 30 negatives (according to their weights).

reviewer	unlike	masterpiece	disappoint	invaluable	hooked
explains	hilarious	refreshing	viorst	gem	amazed
superb	nonetheless	reviewers	winner	fantastic	rocks
highest	complaints	outstanding	magnificent	terrific	finest
riveting	funniest	chilling	thrilled	delivers	copies

Table 4: Strongest 30 positives

disappointing	disappointment	poorly	worst	misleading	useless
ridiculous	fails	waste	unfortunately	claims	lacks
unless	awful	tedious	supposed	unreadable	drivel
terrible	disappointed	pointless	superficial	trash	hoping
outdated	pathetic	garbage	worse	worthless	stupid

Table 5: Strongest 30 negatives

Summary

We described the design of a linear regression machine to be applied on book reviews in order to predict review ratings from the dataset collected by Dredze et al [1]. After optimizing for number of features sizes we got the results of ROC AUC 0.96, RMSE of 0.0027 on the verification data and 1% lift score of 52.8 with 170,000 features. After applying Lasso regularization to "limit" the magnitude of β. We got the results of ROC AUC 0.93, RMSE of 0.003 on the verification data and 1% lift score of 40 with 170,000 features and $\lambda = 5 \cdot 10^{-7}$. Furthermore, the magnitude of β dropped from 669 without regularization to 199 with regularization.

References

[1] J. Blitzer, M. Dredze, and F. Pereira, "Multi-Domain Sentiment Dataset (version 2.0)." [Online]. Available: http://www.cs.jhu.edu/~mdredze/datasets/sentiment/.