HW2: Regression

David Hall
dlwh@cs.berkeley.edu

February 24, 2012

1 Introduction

This report investigates the effect of optimization, objective function, and regularization on a simple regression problem, namely, the Amazon Books dataset described in Blitzer et al. [2007].

2 Model and Optimization

The model we use is a simple regularized regression model using either ℓ_2 or ℓ_1 loss and ℓ_2 or ℓ_1 regularization. That is, we seek to minimize, for $\{p, q\} \in \{1, 2\}$ and tuning parameter λ:

$$\min_{\theta} \sum_i |y^{(i)} - \theta^T x^{(i)}|^p + \lambda ||\theta||_q^q$$

In our situation, each y is the rating of a review (one of $\{1, 2, 4, 5\}$) and x is a bag-of-words of the review. When $p = q = 2$, we recover ridge regression, and when $p = 2$ and $q = 1$ we recover the "lasso," which can obtain sparse solutions. While exact solutions are possible for most of some of these conditions, we instead focus on gradient-based optimizations.

Specifically, we consider two algorithms. First, we consider the widely-used quasi-Newton method LBFGS [Liu et al., 1989], and the OWL-QN variant for ℓ_1 regularization [Andrew and Gao, 2007]. Second, we consider the Adaptive Gradient (AdaGrad) variant of Stochastic Gradient Descent, using forward ℓ_2 and ℓ_1 regularization. [Duchi et al., 2010]. This latter algorithm is much like gradient descent, except that step sizes are determined on a per-component basis, where the step size at time t for component i is defined to be:

$$\alpha_{ti} = \frac{1}{\delta + \sum_{t'=1}^{t} g_{t'i}^2} \tag{1}$$

where g_{ti} is the gradient of component i at time t. Forward regularization complicates the update slightly. We direct the reader to Duchi et al. [2010] for a thorough description, including the actual updates for both ℓ_1 and ℓ_2 regularization.

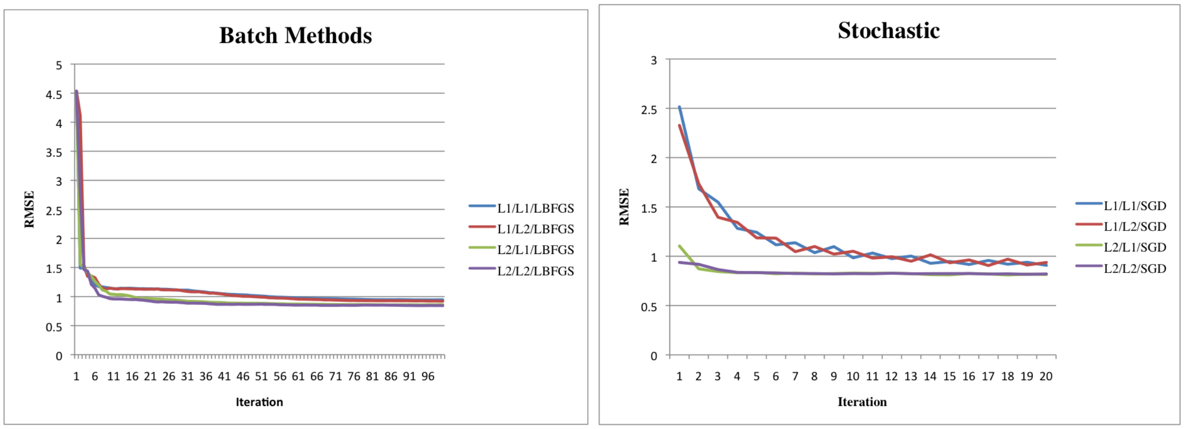

Figure 1: Batch and Stochastic optimization RMSE as a function of iteration

3 Experimental Setup

We use all of the reviews in the Books dataset, using the top 40000 most common words after removing stop words. No attempt was made to remove duplicates, and no additional features were added beyond the raw word counts. We perform a 10-fold cross-validation, as requested.

All implementations of the algorithms are just the ScalaNLP implementations of these algorithms.[1]. The only new code was for reading in the dataset, removing stop words, and the objective function. We parallelized the objective using the Scala-built-in fork/join framework described in Prokopec et al. [2011]. For the AdaGrad updates, we used a batch size of 4096, which was sufficient to saturate two cores of a modern Intel i7 processor. (LBFGS was able to use all four cores.) We use a regularization constant λ of 1E-4. We did not attempt to tune this parameter, except that $\lambda = 1$ clearly underperformed.

For metrics, we focus on Root Mean Square Error (RMSE) as well as classification F1, where the classification task is defined as correctly determining a review as positive (rating of 4 or 5) or negative (rating of 1 or 2).

4 Experiments

We conduct three experiments. First, we plot test RMSE as a function of the number of passes through the data. Second, we consider classification performance for both ℓ_1 and ℓ_2 losses and regularization. Finally, we are interested the sparsity associated with the different executions.

4.1 Convergence rates

In Figure 1 we plotted the average test-set RMSE across validation runs, as a function of pass through the data for the batch and stochastic methods, respectively. Unsurprisingly, the ℓ_2 methods are better at minimizing RMSE, since they are optimizing the actual evaluation criterion.

As is generally observed in the literature, SGD is much faster than LBFGS, achieving optimal perfor-

[1] `http://scalanlp.org/`. Hey, I wrote it!

Obj.	Reg.	Micro	Macro
ℓ_1	ℓ_1	0.917	0.742
ℓ_1	ℓ_2	0.912	0.752
ℓ_2	ℓ_1	0.942	0.841
ℓ_2	ℓ_2	0.937	0.822

Table 1: Micro- and Macro-averaged F1 scores for different combinations of regularization and objective.

mance in just 5 iterations for the ℓ_2 losses. ℓ_1 loss takes a little longer, which is not surprising since it is not smooth. Indeed, the stochastic ℓ_1 methods clearly show a significant amount of "bouncing around," while the ℓ_2 methods converge much more quickly.

It is actually surprising to see ℓ_1 loss behave as well as it does with LBFGS and OWL-QN, which is typically quite sensitive to discontinuities and more generally errors in the gradient in our prior experience. We did observe that LBFGS had to choose smaller step sizes more frequently for the ℓ_1 loss updates.

In terms of final performance, ℓ_2/ℓ_1 performed the best with an RMSE of 0.815, while ℓ_2/ℓ_2 was not far behind at 0.821. The ℓ_1 objectives were around 0.91-0.92, which again is unsurprising since they were not optimizing that metric.

4.2 Classification Performance

We then considered the classification performance for each of the objectives and regularization settings. We report micro- and macro-averaged F1, to account for the fact that most reviews are positive. All scores are averaged over the 10 runs. Table 1 contains the results.

Interestingly, the ℓ_2 objective also outperforms the ℓ_1 on classification accuracy, as well as on RMSE. Perhaps this could be fixed by improving the regularization constant, but we do not pursue that here.

4.3 Sparsity

ℓ_1 regularization is generally known for producing sparse solutions. We investigated the claim empirically by comparing the number of zero entries in the weight vector for both ℓ_2 and ℓ_1 regularization using SGD. ℓ_2, as predicted, did not produce a sparse solution, with more than 99% of the parameters having a non-zero weight. However, using ℓ_1 regularization, only 46% of the parameters have non-zero weight.

4.4 High Weight Terms

Just as an interesting visualization exercise, we print the top twenty terms with the highest absolute weight from the ℓ_2/ℓ_1 configuration, in Table 2. Many of the words are obviously correlated with high or low reviews. ('money' is fairly negative, which is pretty funny, really!) The xml tags are basically bias features: they appear in every document, and the dataset is fairly biased to positive reviews.

Term	Weight
boring	-0.3583
waste	-0.3565
excellent	0.321
disappointed	-0.269
money	-0.263
bad	-0.261
disappointing	-0.2597
wonderful	0.2413
best	0.2362
worst	-0.2285
</title>	0.225
<date>	0.225
</date>	0.225
<title>	0.225
<reviewer>	0.225
</reviewer>	0.225
<review>	0.225
</reviewer_location>	0.225
<review_text>	0.225
</review_text>	0.225

Table 2: Terms with the highest weight in terms of absolute value from our model.

5 Conclusion

We examined the effect of various modifications to optimization, regularization, and objective on a simple regression task. We demonstrated known previously known results: that stochastic methods are typically faster than batch methods, and that ℓ_1 regularization can lead to sparsity. Interestingly, we also found that ℓ_1 loss is not as effective for the classification loss as ℓ_2.

References

Galen Andrew and Jianfeng Gao. Scalable training of l1-regularized log-linear models. In *In ICML 07*, 2007.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Association for Computational Linguistics*, Prague, Czech Republic, 2007.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *COLT 2010: Proceedings of the 23st annual workshop on Computational learning theory*, 2010. URL `http://www.colt2010.org/papers/023Duchi.pdf`.

Dong C. Liu, Jorge Nocedal, Dong C. Liu, and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

Aleksandar Prokopec, Phil Bagwell, Tiark Rompf, and Martin Odersky. A generic parallel collection framework. In *Proceedings of the 17th international conference on Parallel processing - Volume Part II*, Euro-Par'11, pages 136–147, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-23396-8. URL `http://dl.acm.org/citation.cfm?id=2033408.2033425`.