An Analysis of Linear Regression

Austin Barket

Department of Computer Science
The Pennsylvania State University at Harrisburg
Middletown, PA 17057
amb6470@psu.edu

1 Introduction

I seek to analyze the effect of different parameters on a batch gradient descent implementation of multiple linear regression when using various update rules with and without a uniform regularizer. In Section 2, the reader will find a high level description of the dataset used throughout the experiment along with references to several previous works using this dataset. In Section 3 the performed experiment is described in detail, the results of the experiment are provided in Section 4, followed by a discussion of the results in Section 5.

2 Dataset: Combined Cycle Power Plant (Power Plant)

The Power Plant dataset contains 9568 examples collected over 6 years from a power plant set work at full load. Each instance consists of 4 real valued predictors; namely the Temperature (AT), Ambient Pressure (AP), Relative Humidity (RH) and Exhaust Vacuum (V). The target response variable is the net hourly electrical energy output (EP) of the plant [1]. The Power Plant dataset was subjected to extensive cleaning and preprocessing prior to donation to the UCI repository. This preprocessing procedure filtered out all nonsensical outliers and diminished the noise that resulted from electrical disturbance interfering with the signal [2] [4].

3 Experimental Setup

3.1 Algorithm

First, the generic batch gradient descent algorithm provided in class was implemented for the case of linear regression using the mean squared error function. Psuedocode for the implemented algorithm is shown in Algorithm 1.

1. Initialize the weights vector to contain $(numberOfPredictors + 1)$ random values between 0 and 1
2. While (stopping condition has not been met)
   (a) Calculate gradient of the mean squared error across all training examples with respect to the current weights vector.
   (b) Update the weights using an update rule.
   (c) Calculate and store the new error on the training, validation, and test sets as well as the magnitude of the gradient vector for use in the stopping conditions.

Algorithm 1: Generic Batch Gradient Descent Algorithm

3.2 Update Rules

I tested three versions of the gradient descent update rule. First, the original constant learning rate rule, seen in Figure 1 was tested. Note that the notation μ is used to represent the learning rate and λ for the regularization constant. Of course, setting λ to 0 in these equations is equivalent to using no regularization at all. In addition to constant learning rates, the formulas provided in Figure 2 and 3 were used to adapt the learning rate prior to use in the update rule and the results were compared.

The Descending Learning Rate rule in 2 is designed as a very simple scheme to ensure that as the iteration number increases, the effective learning rate will decrease, hopefully leading to a fast initial descent followed by smoothly approaching a minimum error. The constant of 10^4 in this equation was found through some trial and error and was necessary in order to get decent performance when using the same learning rates as used with the Original update rule, which was desirable for comparison purposes. Without this constant scaling of the learning rates, this adaptation scheme would result in learning rates so small that almost no change would be made to the weights each iteration and thus the algorithm would reach the max number of iterations long before convergence.

A similar rescaling was required in order to use the Gradient Magnitude Scaling update rule provided in class, which is shown in Figure 3. The issue here was that the magnitude of the gradient starts out very large. Without drastic rescaling, the algorithm computes enormous learning rates that lead to immediate divergence and error values shooting off to positive infinity. The constant 10^{-6}

$$\boldsymbol{w} = \boldsymbol{w} \cdot \left(1 - \frac{2}{N} \cdot \mu \cdot \lambda\right) - \left(\mu \cdot \frac{\Delta E_{IN}}{||\Delta E_{IN}||}\right);$$

Fig. 1: Original Update Rule

Call Original Update Rule with...

$$\mu = 10^4 \cdot \frac{\mu}{iterationNum}$$

Fig. 2: Descending Learning Rate Update Rule

Call Original Update Rule with...

$$\mu = 10^{-6} \cdot ||\Delta E_{IN}|| \cdot \mu$$

Fig. 3: Gradient Magnitude Scaling Update Rule

was chosen so that the algorithm would remain stable and the magnitidue of the gradient would in fact continuously descend when using the tested learning rates on this dataset.

Of course if a different dataset were used the error landscape and magnitude of the gradients would differ and these constants would need to be changed. If not for the need of easy comparison between the constant and adapted learning rates, one would simply choose larger and smaller learning rates for use in the Descending Learning Rate and Gradient Magnitude Scaling adaptation schemes respectively and avoid the need for these hard coded constants all together.

3.3 Parameters

The learning rates and lambda values tested are provided in Table 1, along with the max number of iterations parameter. For each triplet of μ, λ, and update rule, the gradient descent algorithm was run until either the max number of iterations was reached or all of three tested stopping conditions discussed in subsection 3.4 have evaluated to true. This leads to 6 $\mu's$ $*$ 6 $\lambda's$ $*$ 3 update rules $=$ 108 total tests in each run.

μ	0.0001, 0.001, 0.01, 0.1, .5, 1
λ	0.0, 0.05, 0.1, 0.5, 1, 5
Max Iterations	500000

Table 1: All Tested Parameters

3.4 Stopping Conditions

During each test using a given set of parameters, three individual stopping points are tracked, and the algorithm breaks early only if all three conditions have been met. This will allow us to plot all three stopping points on each of our error curves for a convenient visual representation of the effectiveness of each condition and how they relate to one another.

The first, and most conservative stopping condition is triggered when the algorithm has gone 20000 iterations without improving upon the minimum training error seen so far. Of course stopping the algorithm based on the training set error is never a good idea in practice since the learned weights are heavily biased towards predicting the training data and thus one cannot obtain a reliable estimate of the generalization error based on the training data alone. For our purposes this stopping condition serves as a conservative way to prevent the algorithm from running all the way to the max number of iterations if it does not need to. If the algorithm has converged to minimum error on the training data (or in odd circumstances it begins to diverge on the training data), this should be a pretty safe indication that further training is pointless, and should result in error curves that simultaneously display the point at which a better stopping condition would have stopped and what would have happened if it did not stop.

The second, more typical stopping condition is triggered when the algorithm has gone 5000 iterations without improving upon the minimum validation error seen so far. As we discussed extensively in class the best way to get an unbiased estimate of E_{out} is to use some form of validation set, which is used during training only for the purposes of estimating the out of sample error and stopping the algorithm when this validation error stops decreasing or begins to increase due to overfitting of the training data. For the purposes of our results and discussion, the validation stopping iteration, training error, validation error, and test error refer to the iteration at which the minimum validation error was actually acheived, not the iteration at which the stopping condition was triggered (which will be 5000 iterations past the actual minimum).

One additional stopping condition was implemented that is specialized for the gradient descent algorithm. Specifically, the training procedure tracks the point at which the magnitude of the gradient vector does not decrease for more than 5000 iterations. Similar to the validation based stopping condition, when we refer to the gradient stopping iteration, training error, validation error, and test error, we actually mean the iteration prior to that stopping point that corresponds to the minimum validation error, not the actual iteration when the condition was triggered or even the point where the minimum gradient magnitude was observed. This distinction is important. Even though we are stopping based on the gradient magnitude, this value is not suitable for use as a generalization error estimate since, similar to the training error, the gradient magnitude is heavily biased to the examples in training sample. Thus the validation error is still used to determine the actual weights (and thus the error values) that would

be returned if the algorithm were truly stopped using this gradient magnitude based condition and was meant for real world use.

3.5 Run Procedure

Each run begins with a uniformly at random partitioning of the dataset. 60% goes to the training set, 20% is used for validation, and the last 20% is held as a test set to get an unbiased estimate of E_{out}. All 108 sets of parameters are evaluated using this same dataset split to ensure comparability of their results.

Ten independent runs were performed and all run data was averaged. The results found in 4 are derived from these averages. Note I had planned on performing twenty or thirty runs rather than five to improve statistical significance. Unfortunately, I ran out of time due to last minute adjustments to the experiment. Although not ideal, ten runs should still be sufficient to get a good idea of how the hypothesis space of linear regressors responds to the various parameters tested.

4 Results

Following the collection and averaging of all the run data, the parameters were sorted by the minimum validation error achieved when using both the validation error and gradient magnitude stopping conditions.

Table 2 summarizes the best parameters found for each of the update rules. The top subtable contains the best parameters found when the validation error stopping condition was used; and the bottom subtable contains the best parameters found using the gradient magnitude stopping condition. Note the bold entries in these tables represent the minimum value in their respective rows.

Figures 4 and 5 contain the error curves, gradient magnitude curves, and learning curves for the parameters shown in Table 2. The gradient magnitude curve simply graphs the actual magnitude of the gradient vector at each iteration shown in the corresponding error curve. This is interesting to see since we know for gradient descent works best when we have a smooth continuous decrease in the magnitude of the gradient. Note the learning curves only show up to using 200 examples, rather than all 9568 examples in the dataset. This is because, as the curves clearly show, the linear regressors converge to having nearly equivalent training, validation, and test errors extremely quickly. Early testing revealed that, when graphed all the way to 9568 examples, the training error (red) continues its trend of decreasing variance and converges to be approximately identical to both the validation and test error. This interesting phenomena, as well as the trends seen in the error and gradient magnitude curves, will be discussed in more detail in Section 5.

Update Rule	Original	Descending Gradient	Magnitude
Learning Rate	0.0001	0.01	0.5
Lambda	1	1	0.1
Validation Stop Iteration	**499940**	499989	499996
Validation Stop Training Error	5.040914	**5.028825**	5.047131
Validation Stop Validation Error	**5.018132**	5.042819	5.020697
Validation Stop Test Error	5.097072	5.07811	**5.077022**
Gradient Stop Iteration	**499940**	499989	499996
Gradient Stop Training Error	5.040914	**5.028825**	5.047131
Gradient Stop Validation Error	**5.018132**	5.042819	5.020697
Gradient Stop Test Error	5.097072	5.07811	**5.077022**

Update Rule	Original	Descending Gradient	Magnitude
Learning Rate	0.0001	0.01	0.5
Lambda	5	1	0.1
Validation Stop Iteration	**195112**	499989	499996
Validation Stop Training Error	5.041867	**5.028825**	5.047131
Validation Stop Validation Error	**5.018964**	5.042819	5.020697
Validation Stop Test Error	5.097071	5.07811	**5.077022**
Gradient Stop Iteration	499998	**499989**	499996
Gradient Stop Training Error	5.0401	**5.028825**	5.047131
Gradient Stop Validation Error	**5.017267**	5.042819	5.020697
Gradient Stop Test Error	5.095771	5.07811	**5.077022**

Table 2: Parameters with Lowest Validation Stop Validation Error (Top) and Lowest Gradient Stop Validation Error (Bottom)

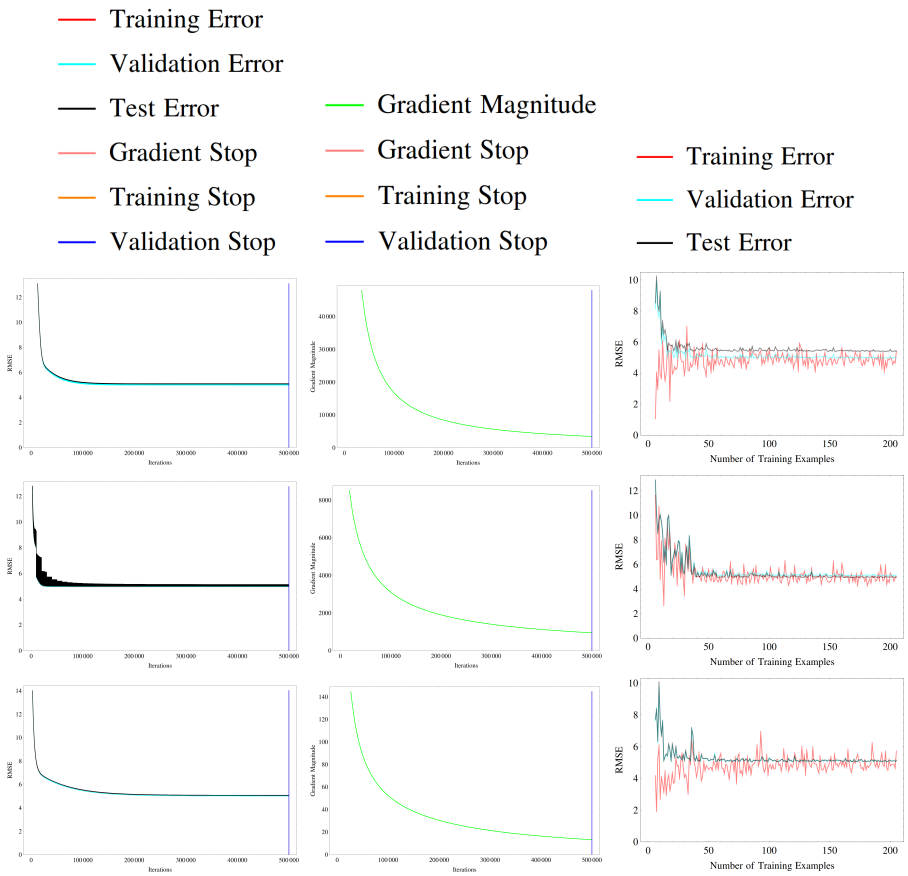

Fig. 4: Error (Left) , Gradient Magnitude (Middle), and Learning (Right) Curves for Best Original (Top), Descending Learning Rate (Middle), and Gradient Magnitude Scaling (Bottom) Parameters with Lowest Validation Stop Validation Error

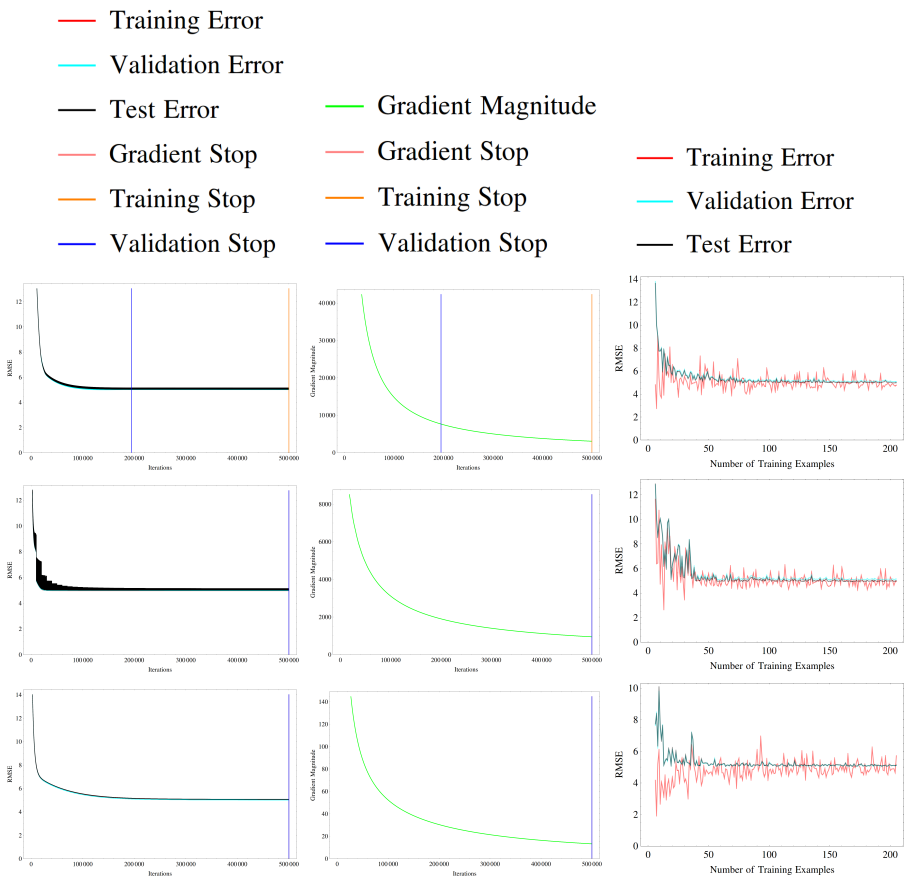

Fig. 5: Error (Left) , Gradient Magnitude (Middle), and Learning (Right) Curves for Best Original (Top), Descending Learning Rate (Middle), and Gradient Magnitude Scaling (Bottom) Parameters with Lowest Gradient Stop Validation Error

5 Discussion

Starting with the information about the best parameters found in Table 2, the first thing to notice is that regardless of which stopping condition and update rule is used, the minimum validation root mean squared error achieved is essentially the same, differing by only a few hundredths between the different update rules, thus which stopping condition and update rule combination actually achieved the minimum is not very significant since with more run data its very possible that these averages will shift slightly in favor of a different update rule stopping condition pair.

The other factor that jumps out from these tables is that the training, validation, and test errors are all essentially the same across the board. By looking at the Vapnik-Chervonenkis Generalization Bound we can get an intuition behind why this is the case. This equation tells us that as the number of samples N becomes large, the quality of generalization improves. More specifically, the probability of E_{IN} and E_{OUT} being very close to each other increases. In addition linear regression has a very small vc dimension, in particular it should be approximately the same as the number of weights the algorithm has available to define the resulting line, which in this case is only 5. The VC Generalization Bound also tell us that by using a simple hypothesis space such as linear regressors, the generalization penalty will be small and E_{IN} and E_{OUT} will be close to each other. Thus the results are in line with what one would expect when applying a simple hypothesis space to a relatively large dataset.

Also note that in all cases, the best parameters contain a non-zero regularization constant λ. This was somewhat unexpected since linear regression is such a simplistic algorithm, it would seem that regularization would only imped the ability of the algorithm to make the best fit it possibly can on a dataset that is not perfectly linear. However, from the linear regressor's point of view, the non-linear relationships in the dataset would appear as deterministic noise, and thus its possible that the regularization allows the model to focus more on the predictors that are linear correlated with the response and less so on those that are not which would explain the reduced overall error that we are seeing.

Finally, in Table 2 we see the unexpected result that the best parameters are the same for both stopping conditions for all cases except the Original update rule. In this case, the learning rates are the same, only lambda differs, and the observed error values only differ in the thousandths place. Moving to the error, gradient magnitude, and learning curves of Figures 4 and 5, its apparent that the reason the stopping condition does not seem to matter among the best parameters, is that the conditions did not have a change to be met before reaching the maximum number of iterations. We can also see that in all of these cases, there would be very little value to continuing with the learning as the error curves have essentially flatlined long before that max number of iterations was reached. The fact that the stopping conditions were not met indicates that the validation error was technically still decreasing all the way to the end but by visual analysis these decreases were too small to be of any practical value. The logical remedy for this would be to add a small threshold on the amount the validation error must de-

crease by in order to continue training. The gradient magnitude based stopping condition could also be remedied by adding a decrease threshold. Specifically, the threshold should be applied to the slope of the gradient magnitude curve and would ideally detect the point where the curve is at the "vertex" of the slightly parabolic looking trend it follows. That is the point where the iterations begin increasing "much faster" than the magnitude is decreasing. The difficulty would be finding exactly where this "much faster" point is since it would likely vary drastically for different datasets, response variables, and especially error functions.

The evidence of the VC Generalization Bound is quite apparent in the error curves where we see that there is never a significant, noticeable difference between the training, validation, and test error values. Since the number of training examples far exceeds the vc dimension of linear regressors, all the examples found in the validation and test sets are highly unlikely to contain any relationships that are not already in the training set and that the linear regressor is actually capable of modeling using its extremely limited number of parameters. Thus regardless of where the learned line actually lies in the space of the dataset, its performance will be almost identical across all of the dataset partitions. In bias-variance-noise terms the same argument can be made by noting that simple models have very low variance, so all the possible models will have very similar performance. The low variance, high bias tradeoff is quite apparent in the learning curves of figures 4 and 5. Here we see that past 50 or so training examples, the learning capacity of the linear regressor has already been saturated and all the error values have converged to a value around five which is approximately the same minimum error we were able to achieve using 60% of the dataset for training. This makes sense from a bias-variance-noise point of view because this apparent invisible threshold corresponds to the bias and the noise that is inherent to the chosen hypothesis space and this dataset. No matter what parameters or training data we throw at the algorithm, it can never achieve better than the bias plus the noise.

6 Conclusion

I implemented batch gradient descent implementation of multiple linear regression and applied it to a relatively large publicly available dataset from the UCI Machine Learning Repository. The algorithm was tested using several variations of the gradient descent update rule with various learning rates and regularization constants.

The results show that the tested update rules, stopping conditions, learning rates, and lambda values do not result in significant differences in the resulting models; the conclusion being that linear regressors are too simple of a hypothesis space for this dataset and under fitting is likely. Due to the small vc dimension, low variance of linear regressors, and the large size of the dataset, the VC Generalization Bound and Bias-Variance-Noise analysis accurately predict the observed results, which show almost no difference in the errors on the training,

validation, and test sets for the best parameters. In addition, the learning curves show us that using 50 or greater training examples results in achievement of approximately the same minimum error values. Thus it seems that this minimum can be attributed to the bias and noise of linear regressors on this dataset and its unlikely that any parameter values could lead to a linear regression model significantly better than the minimums observed in this experiment. In the real world, if better performance than this was required, linear regressors would need to be thrown out in favor of a more complex algorithm with a larger VC dimension and lower bias.

References

1. "Combined Cycle Power Plant Data Set", [Online]. Available: http://archive.ics.uci.edu/ml/datasets/Combined+Cycle+Power+Plant
2. P. Tfekci, "Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods", *International Journal of Electrical Power & Energy Systems*, pp. 126-140, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.ijepes.2014.02.027
3. H. Kaya, P. Tfekci, "Local and Global Learning Methods for Predicting Power of a Combined Gas & Steam Turbine", *Proceedings of the International Conference on Emerging Trends in Computer and Electronics Engineering ICETCEE 2012*, pp. 13-18, 2012. [Online].
4. S. Chang, "COMP 520 Class Notes".