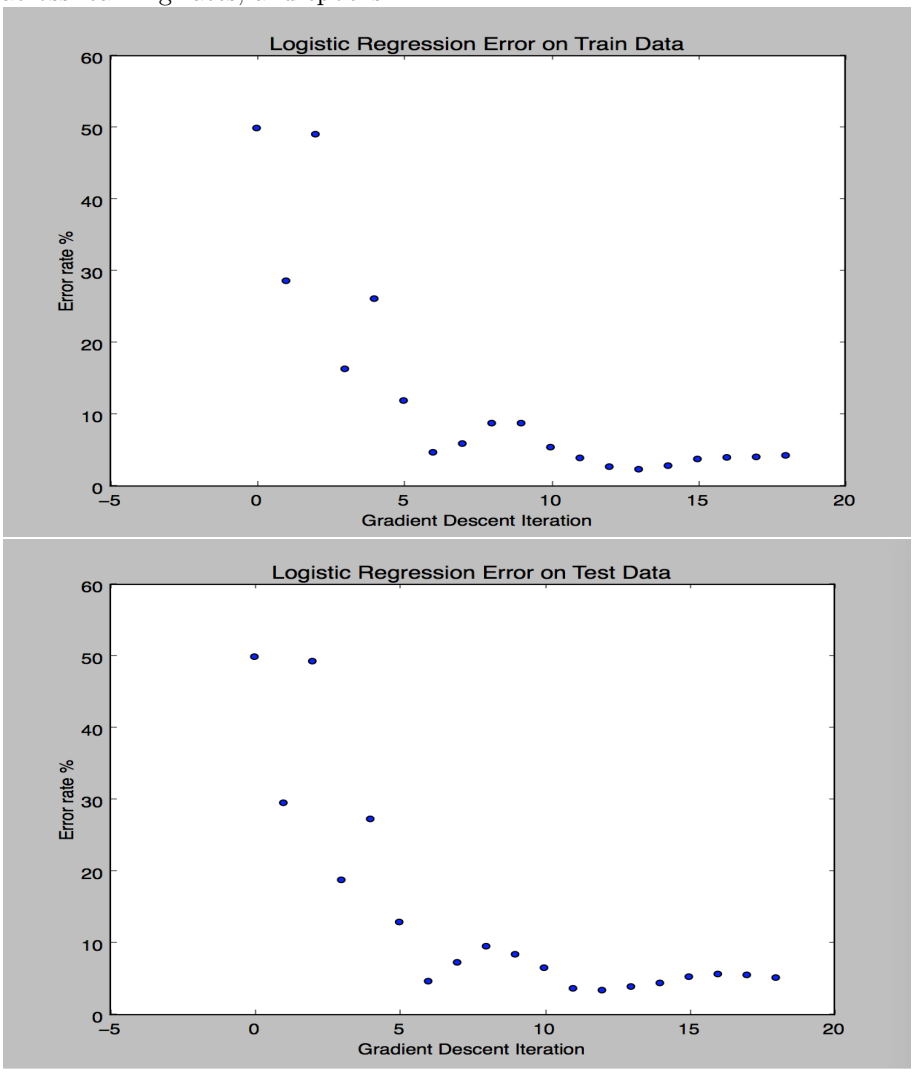
Logistic Regression

Neale Ratzlaff

17 April 2016

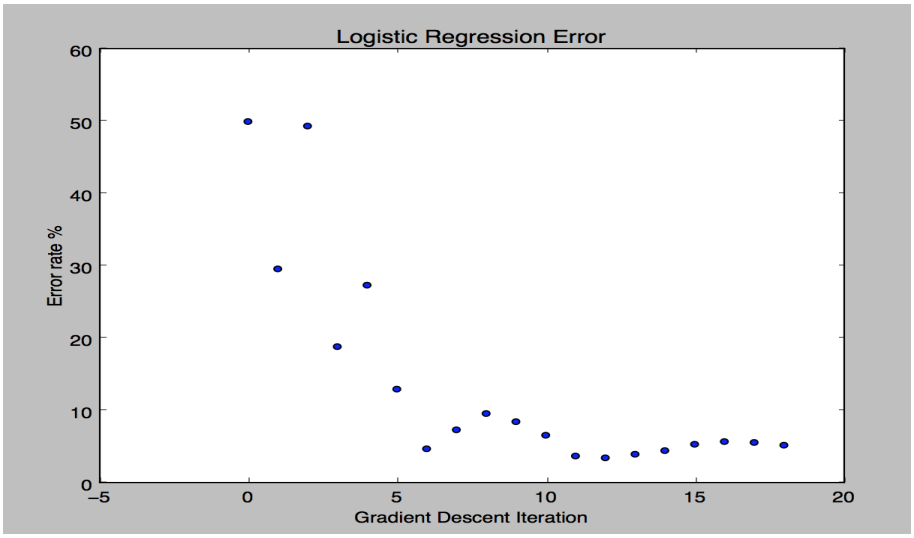
1 Learning Rate

I preformed gradiant descent with learning rates from .000001 to 100000. It quickly became clear that any learning rate above 1.0 would cause an overflow error as the sigmoid function was applied. I attempted to normalize the features between 0 and 1 in order to combat this effect. This allowed me to use learning rates of 100 or less. Over my test runs I noticed a decline in the classifier's ability to converge when using a learning rate greater than or equal to 0.5, with the best results coming from learning rates of 0.01 and 0.005. The improvements were marginal however, epoch 12 would always see a lower error rate than iterations beyond it. The system was tuned using five fold cross validaton across learning rates, and epochs.



2 Gradient Descent

To examine the trend of the error as the gradient descent update occurs, I chose the learning rate of 0.01 as it had the lowest validation error rate. I ran batch gradient descent 20 times and examined the error using matplotlib. The plot of error percentage against gradient descent iteration is shown below. Running it further became computationally prohibitive as running times approached one hour, and the model showed no signs of converging further beyond a 2 to 5 percent error.
The data did very well on the testing data, acheiving lows of 3 percent error. The error for the training data is very similar in error rate. In all cases, there seems to be a minima during epoch 12 that is reached, the subsequently left as the error starts to climb again, leveling off at around 5 percent.

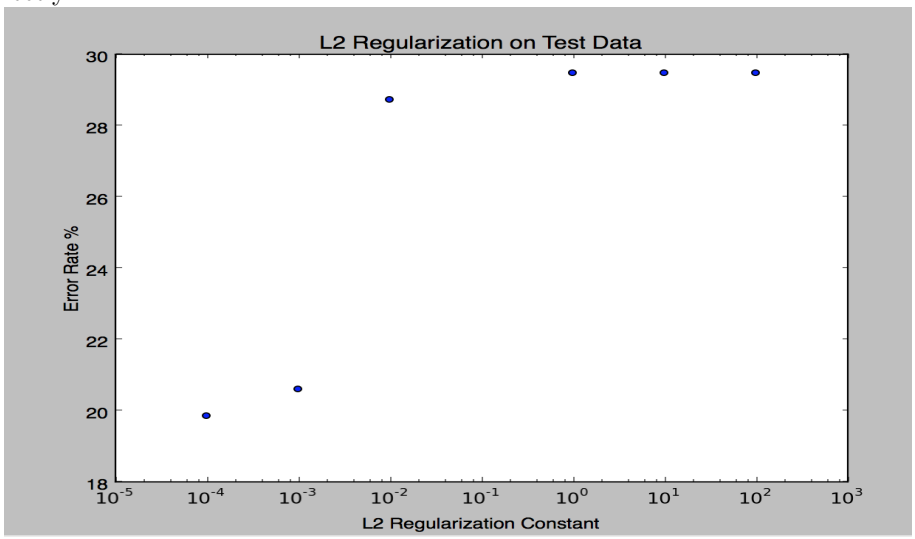


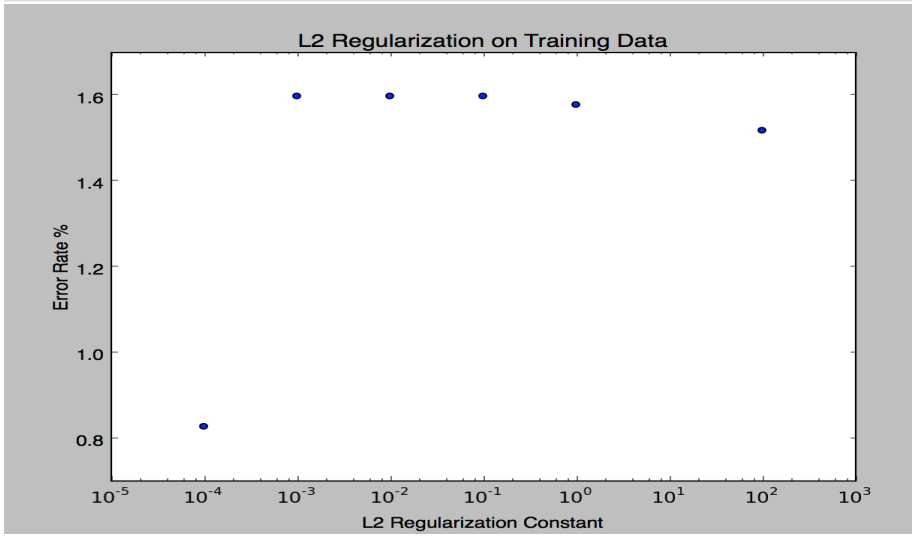
3 Regularization Pseudocode

- gradient descent using L2 regularization
- for all xi in X, sum:
- calculate the sigmoid activation function for each xi
- subtract value from the given label to get error
- add regularization term: constant * square of the norm
- Update parameters with new delta * learning rate

4 Regularization testing

L2 regularization was added to the objective function as a way to penalize the parameters. As seen from the graphs below there was not a lot of movement in terms of the error rate with variation in lambda. What regularization did clearly do, is make the classifier less accurate on the training data. There are some possible explanations for this. It is possible that the training and testing data lacks varience and is very prone to overfitting. The data is so similar that overfitting the training data results in a high degree of accuracy on the test data. Optimizing with the regularization term then forces the model to avoid large weights and therefore be able to generalize better, and become simaltanously less accurate on the given test data. It is also possible the algorithm was not executed correctly.





Theoretically it can be said that as lambda grows, the weights are allowed to grow larger in proportion, this is reflected in the analysis below. Here the parameters were averaged into one value and compared as lambda is allowed to grow to 100.