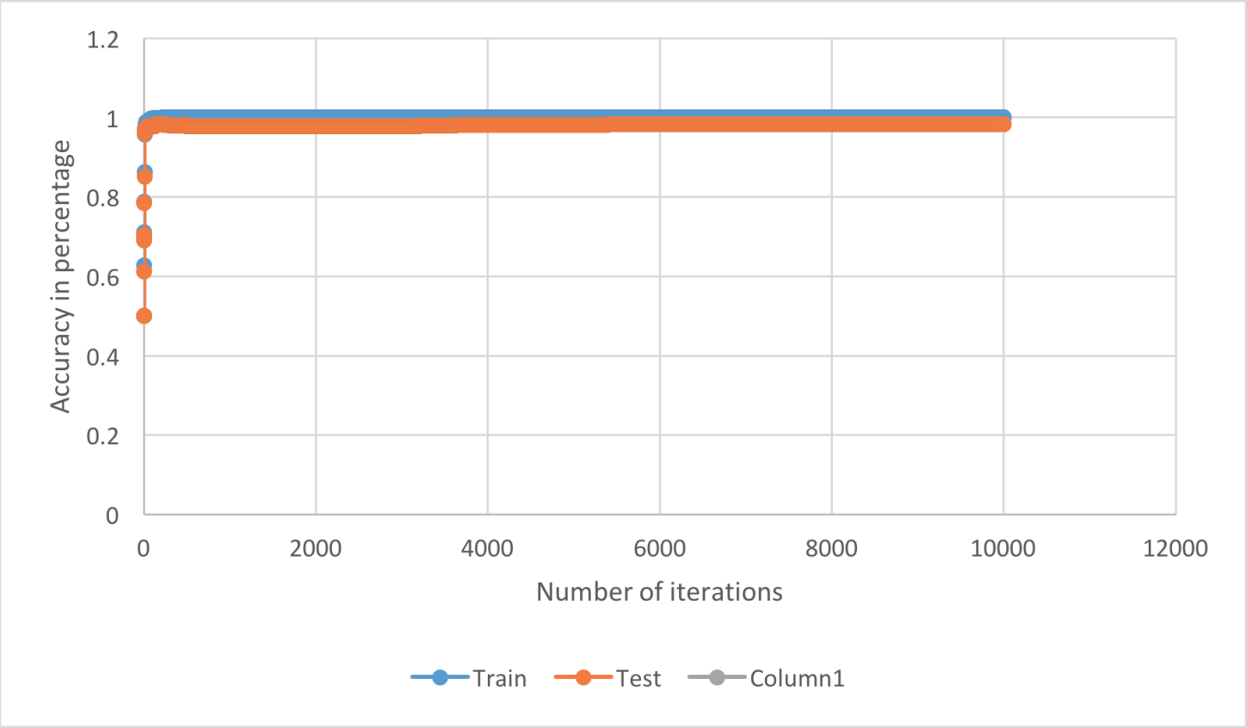
CS 434 Write up (HW #2)

SR Kanna, Ramcharan Sudarsanam, Preston Wipf

1. *(30 pts) Implement the batch gradient descent algorithm to train a binary logistic regression classifier. The behavior of Gradient descent can be strongly in influenced by the learning rate. Experiment with different learning rates, report your observation on the convergence behavior of the gradient descent algorithm. For your implementation, you will need to decide a stopping condition. You might use a fixed number of iterations, the change of the objective value (when it ceases to be significant) or the norm of the gradient (when it is smaller than a small threshold). Note, if you observe an overflow, then your learning rate is too big, so you need to try smaller learning rates.*

   We used a learning rate of approximately .0000001. We choose a fixed stopping condition of 10,000 iterations.

2. *(15 pts) Once you identify a suitable learning rate, rerun the training of the model from the beginning. For each gradient descent iteration, plot the training accuracy and the testing accuracy of your model as a function of the number of gradient descent iterations. What trend do you observe?*



For every gradient descent, we updated w. We used this w on the training and testing set to predict the classification using $p(y=1|x)/p(y=0|x) > 1$. This does not account for the corner case of $p(y=1|x)= 0.5$, but due to it's rarity, this method is still accurate. We used this to predict the accuracy by determining the number of training/testing examples correctly classified. The graph above shows the training and testing accuracy by using a learning rate of .0000001 across the 10,000 iterations.

The training data, which updates w, has 100% accuracy by the 200th iteration. The first ten to fifteen predictions are below eighty percent, but once w becomes updated with the training data's delta, the accuracy is extremely precise. The testing data reaches reaches 98% accuracy by the 15th or so iteration. (To put this in perspective, 98% in a set of 800 is less than ten incorrect predictions.) The accuracy is as low as 50% because w is neither fit well for the training or testing data. Eventually as w gets updated, the testing accuracy reaches the high 90's, but it never reaches 100% accuracy because w is updated using training data. Even with learning rates other than 0.0000001, there were only minute improvements.

3. *(15 pts) As discussed in class, Logistic regression is typically used with regularization. We will explore L2 regularization for this question. In particular, we will the following objective with an additional regularization term that is equal to the squared Euclidean norm of the weight vector. where the loss function l is the same as introduced in class (slide 7 of the logistic regression notes). Find the gradient for this objective function and modify the batch gradient descent algorithm with this new gradient. Provide the pseudo code for your modified algorithm.*

Matrix cookbook:

$$\frac{\partial ||\mathbf{x}||_2^2}{\partial \mathbf{x}} = \frac{\partial ||\mathbf{x}^T\mathbf{x}||_2}{\partial \mathbf{x}} = 2\mathbf{x}$$

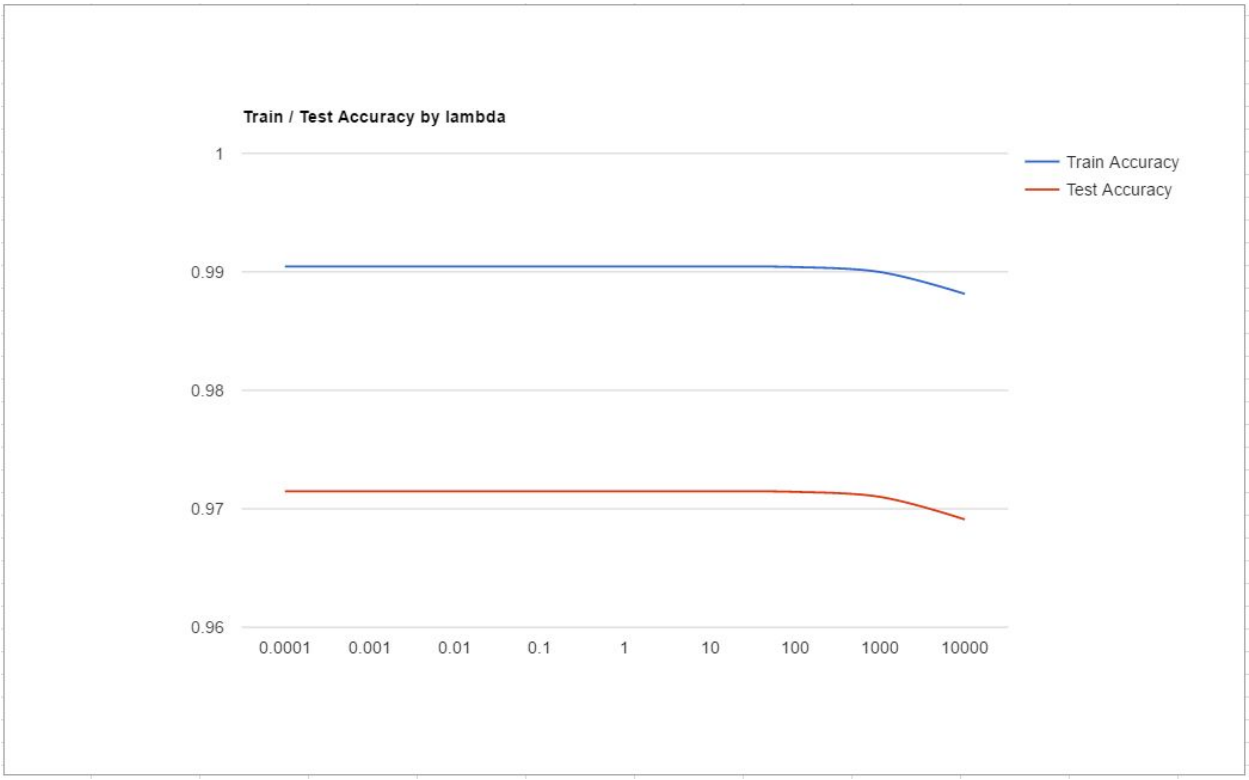
$= 1/2\ \lambda||w||^2$

$= 1/2 * \lambda * 2w$

$= \lambda w$

Given: training examples (x^i,y^i), i = 1...N
Let w <- (0,0,0...0)
Repeat until convergence
    D <- (0,0,0..0)
    For i = 1 to N do
        Yhat = 1/1+e(-w*xi)
        Error = yi - yhat
        Delta = delta + error*x^i
    $w \leftarrow w + n(d + \lambda w)$

4. *(30 pts) Implement the algorithm in [3], and experiment with different Lambda values (e.g., 10e−3 ; 10e−2 ; . . . ; 10e3). Report the training and testing accuracies achieved by the weight vectors learned with different values. Discuss your results in terms of the relationship between training/testing performance and the values.*



The updated algorithm take the sum of the loss function and the regularization to compute w. The larger lambda gets, the more weight it holds the equation we are trying to minimize. The complexity of our model is penalized by the largeness of the lambda. So as we scale up lambda, the regularization term makes up a larger and larger proportion, forcing the loss function to be smaller and smaller.

There is minimal change as the as the lambda values increase. As lambda increases, the accuracy decreases by less than a percent. A greater decrease in accuracy can be observed if value of lambda is substantially increased. This can be explained because with a large lambda, the the loss function becomes less important, which produces less accurate results. The training and testing data differs by approximately .02 accuracy. This is because w is fit based on the training data.