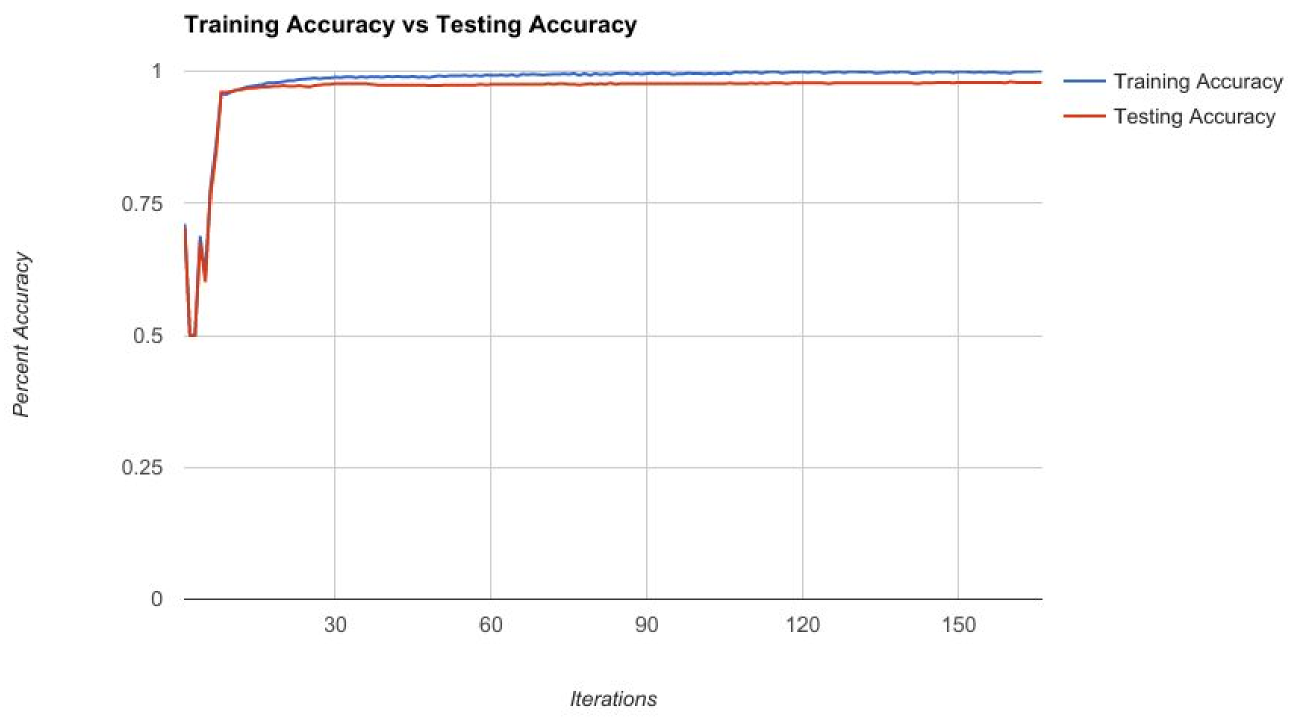
Nathan Brahmstadt
Jordan Crane
CS 434
Assignment 2

1. Our observation with experimenting with learning rates shows a few things. Generally, decreasing the learning rate meant that the gradient descent took longer to converge. With a smaller learning rate though, we were able to obtain a higher accuracy if we were willing to wait for the program to execute.

Our decided stopping condition was a combination of two things. First, finding the norm of the gradient, and checking to see if it was less than an arbitrarily small number (epsilon). Second, check to see if a max number of iterations has run (to prevent it from taking too long).

2. Graphing accuracy against number of iterations for a learning rate of 0.001 we get the following:



As you can see from the data obtained by plotting the accuracy over iterations, the testing accuracy closely follows training accuracy. We can observe slight overfitting from this data, as training accuracy improves but testing accuracy does not.

3. Just as we derived the gradient from the loss function, we can find the derivative of the regularization term, with respect to w, and subtract it from our gradient. For that we get simply get λ*w. So we get the following in pseudocode:

```
Given : training examples ($\mathbf{x}^i$, $y^i$), i = 1,...,N
Let $\mathbf{w}$ = (0,0,0, ...,0)
Repeat until convergenc e
    d = (0,0,0, ...,0)
    For i = 1 to N do
        $y\_hat^i$ = sigmoid($-\mathbf{w}\cdot\mathbf{x}^i$) + $\lambda * \mathbf{w}^i$
        error = $y^i$ - $y\_hat^i$
        d = d + error$\cdot\mathbf{x}^i$
    $\mathbf{w} = \mathbf{w} + \eta$d
```

4. The behavior we observed for various levels of regularization were somewhat odd. As expected, the regularization term had a slowing effect on the change of the norm, so it took more iterations to converge. However, the regularization term appeared to have no effect on accuracy until it reached the order of 10 or 100.. The effect of lambda was also heavily dependent on learning rate, which makes intuitive sense: if the learning rate and lambda are orders of magnitude different, the larger one will quickly overpower the smaller one. Since we used a learning rate of 0.01 for our end results, it makes intuitive sense that lambda would start having an effect when it did, as it had reached several orders of magnitude greater than the learning rate. For a learning rate of 0.01, our tests suggest that a lambda somewhere in the range of 1-10 would be ideal.