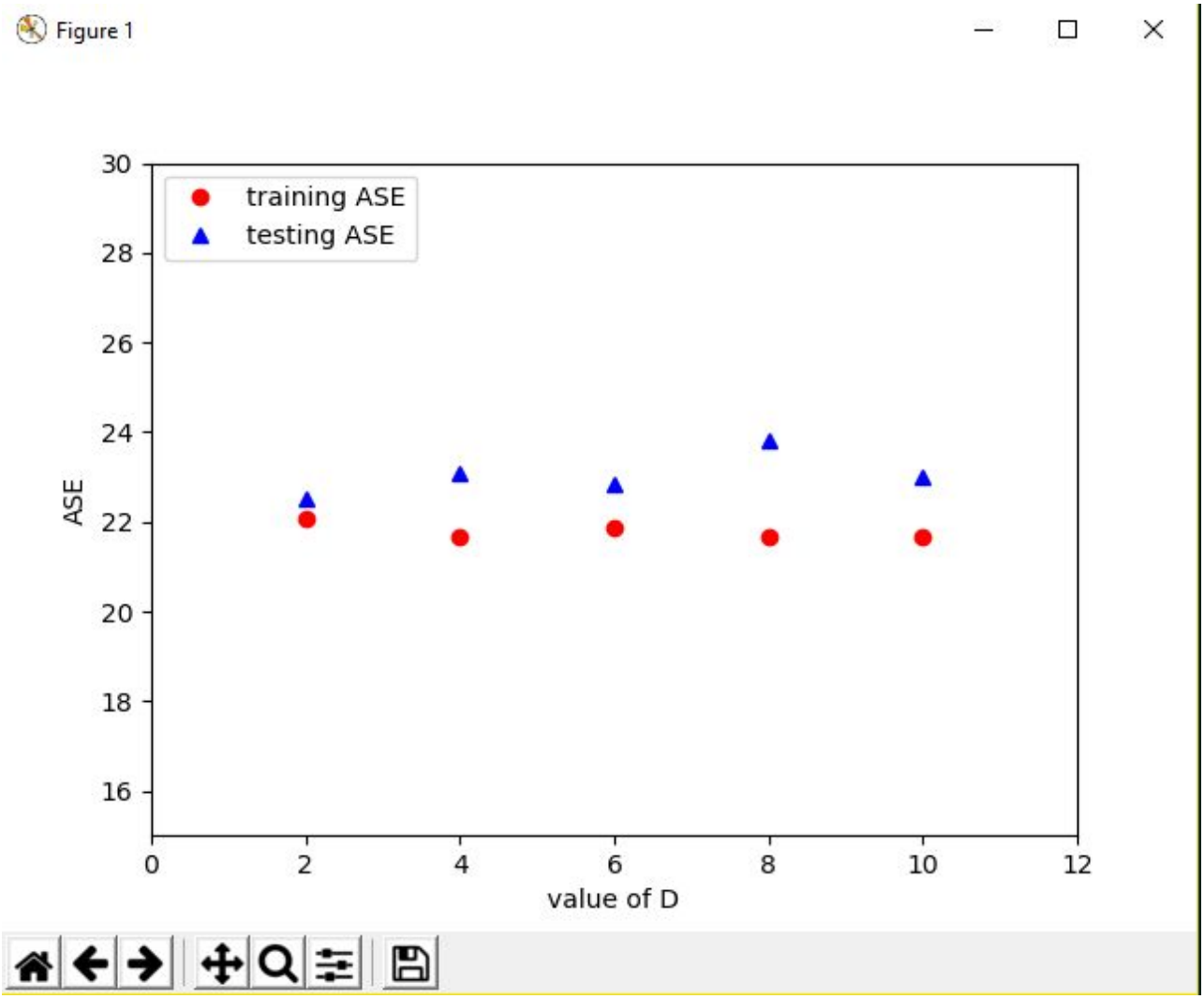
Danh Nguyen
Coulby Nguyen
Mingyu Zhang

# Implementation Assignment 1 WriteUp

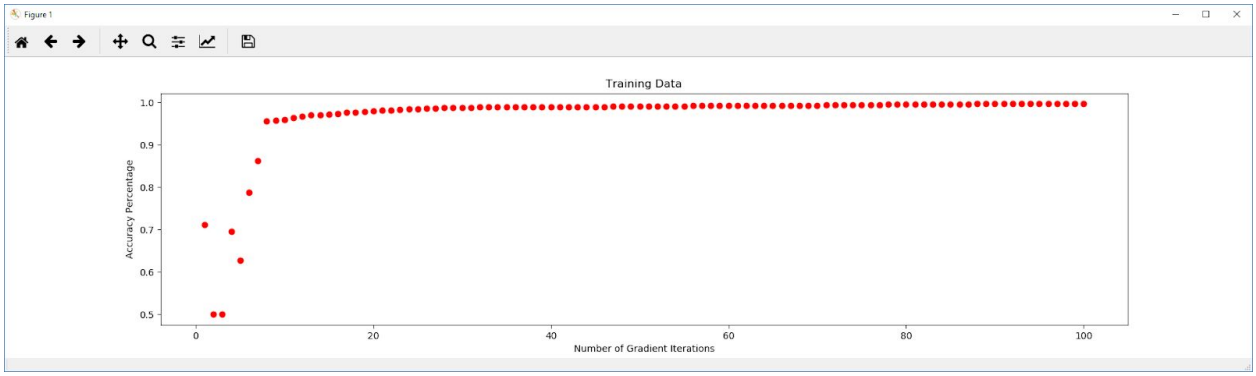
**Part 1**

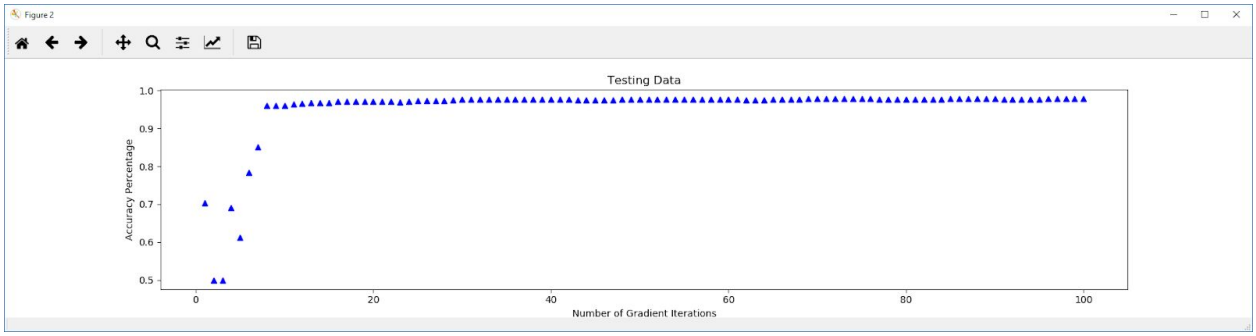
1) The Learned vector which we got from the data was
   [39.584, -0.011, 0.046, -0.002, 3.072, -17.225, 3.711, 0.007, -1.599, 37.362, -0.016, -1.024, 0.009, -0.586]
2) The ASE for the training data is 22.081
   The ASE for the testing data is 22.638
   The ASE for the testing data is higher because the weight vector which we calculated was created as a model from the training data; therefore, the average square error for the training data should be less since the model generalize the training data more. This outcome was consistent with our expectation because we knew how the model was created.
3) The learned vector which we got from this unbiased data was
   [-0.097, 0.049, -0.025, 3.450, -0.355, 5.817, -0.003, -1.021, 0.227, -0.012, -0.388, 0.017, -0.485]
   The ASE for the training data is 24.476
   The ASE for the testing data is 24.292
   Without the dummy variable, the ASE for both data were a bit higher. The reason the ASE are higher because we don't have y-intercept variable within our model. The y-intercept would have shift the prediction of the data upward or downward toward the area where most of the results occur.
4) The below figure shows the average square error for training and testing data as the number of random features increases. The figure suggest that as the number of features increases, the difference in average square error seems to also increase. Which is close to what we predicted since we know that the average square error for testing data would always be more than training data; therefore, if we created a model on the training data and use it for the testing data, then we should see the difference in ASE of both data increases as we increase the number of features.

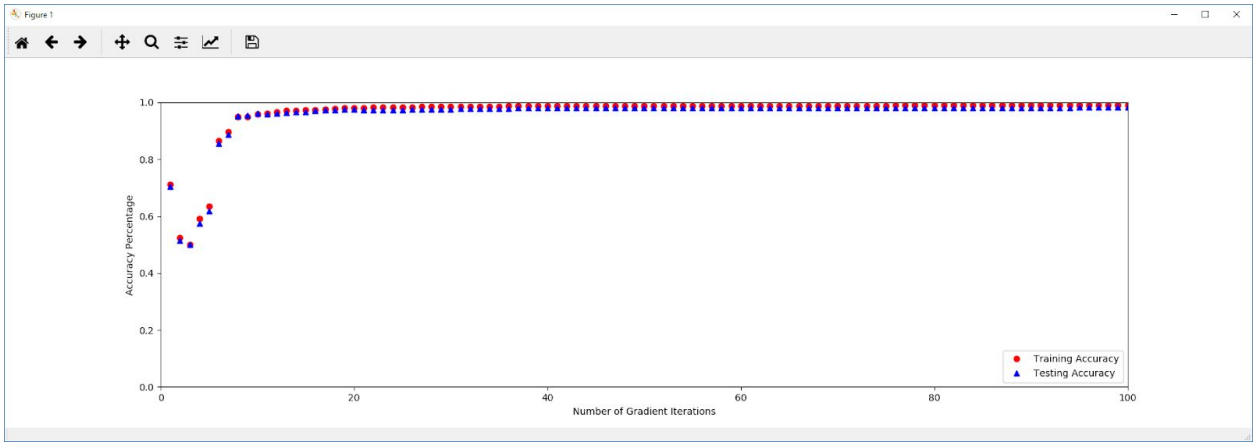



**Part 2**

1) The figures below shows the batch gradient logistic algorithms accuracy in regards to predicting the outcome based on the training and testing data sets. In the figure it can be easily noted that the predictions on the training data yielded higher results than the testing data, and this is due to the fact that the algorithm was classified on the training set therefore it should have a higher accuracy. Using a learning rate of .00000001 and 100 iterations, there becomes an observable trends, that with greater numbers of iterations through the batch gradient logistic algorithm, the predictions become more and more accurate. One thing to note in output of this problem, was that after the initial iteration the accuracy dropped and then converged to near 98% prediction.




2) Below is the gradient of the objective function.

$$L(w) = -\sum_{i=1}^{n} l(g(w^T x^i), y^i) + \frac{1}{2}\lambda|w|^2$$

$$l_i = -y_i \log \sigma(w^T x_i) - (1-y_i)\log(1-\sigma(w^T x_i)) + \frac{1}{2}\lambda|w|^2$$

$$\nabla l_i = \frac{-y_i}{\sigma(w^T x_i)} \nabla\sigma(w^T x_i) + \frac{1-y_i}{1-\sigma(w^T x_i)} \nabla\sigma(w^T x_i) + \lambda|w|$$

$$\nabla l_i = (\sigma(w^T x_i) - y_i + \lambda|w|)x_i$$

Below is the pseudo code for the new gradient function.

Given: training examples($\mathbf{x}_i$, $y_i$), i = i, …, n
Let w <= (0, …, 0) // initialization
Repeat
$\nabla$ = (0, …, 0)
For i = 1,...,n

predictedY <= $\frac{1}{1+e^{-w^T x_i}}$

$\nabla$ <= $\nabla$ + (predictedY - expectedY)$x_i$

w <= w -λ ( $\nabla$ + |w|)
Until | $\nabla$ | $\leq$ ϵ

3)In regards to performance, when using a large learning rate, lambda, the data could not predict higher than a 50% chance. This was true for any lambdas such that $\lambda \geq 0.1$ . However when lambdas were smaller sizes they accuracy shot up greatly to to over 85% at just $\lambda = 0.01$ . This could be the result of overstepping the peaks, and not being able to fine tune to get the weight vectors needed to accurately predict the outcome.