Oleksii Zadniprianyi, Jordan Gilman

Assignment 1 Written report

Linear regression

1. Learned weight vector:
   [-9.79342380e-02 4.89586765e-02 -2.53928478e-02 3.45087927e+00
    -3.55458931e-01 5.81653272e+00 -3.31447963e-03 -1.02050134e+00
     2.26563208e-01 -1.22458785e-02 -3.88029879e-01 1.70214971e-02
    -4.85012955e-01]

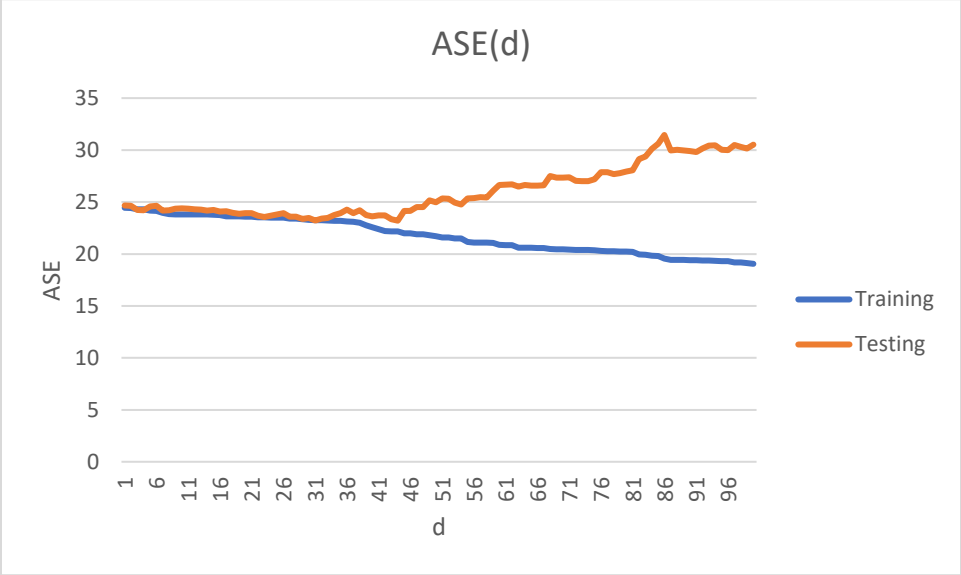
2. Training Results w/ dummy variable:
   SSE: 9561.191289976703 ASE: 22.08127318701317

   Testing Results w/ dummy variable:
   SSE: 1675.2309659473267 ASE: 22.638256296585496

3. Training Results:
   SSE: 10598.057245750708 ASE: 24.475882784643666

   Testing Results:
   SSE: 1797.6256249993 ASE: 24.292238175666217

   Having additional dummy variable results in smaller ASE. Removing dummy variable, which represents shift (b) in our linear function introduces additional error. Therefore ASEs are bigger.

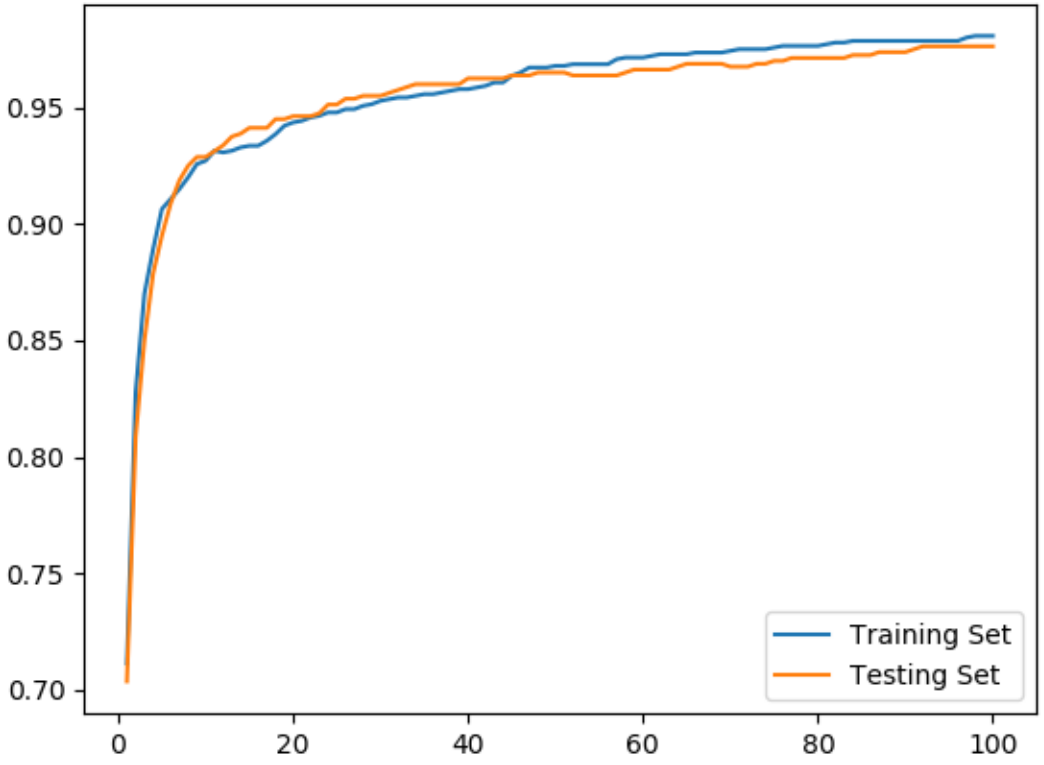
4. 

   With addition of random features to the data, training ASE decreases. This is due to overfitting of the train data to the random additional features. However, testing ASE increases because we are trying to fit testing data to random noise that is not related to desired outputs. More features don't necessarily lead to better prediction of performance at testing stage. Using random features actually decreases performance. But, using

more features that correlate to the desired outputs would result in an increased performance during testing stage.

Logistic regression with regularization

1.



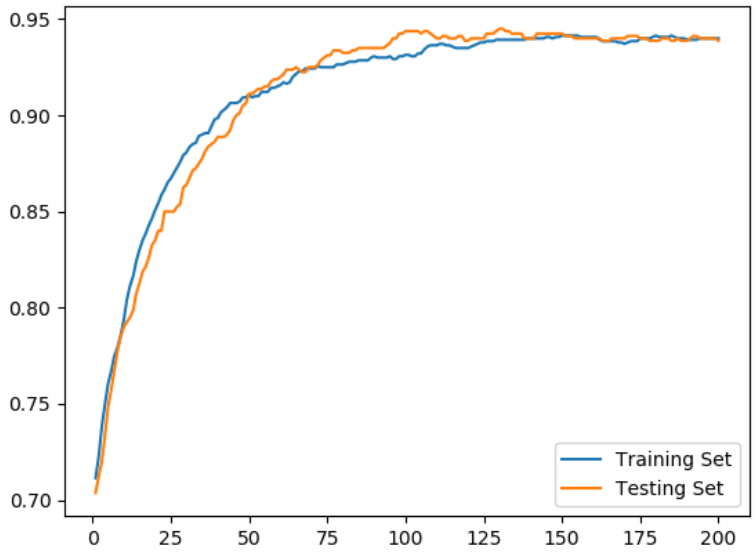
We observe exponential function that quickly decays toward 100% accuracy. In the first 20 batches, algorithm learns to differentiate 4 and 9 with ~95% accuracy, after that each batch offers marginal increase in accuracy.

2.

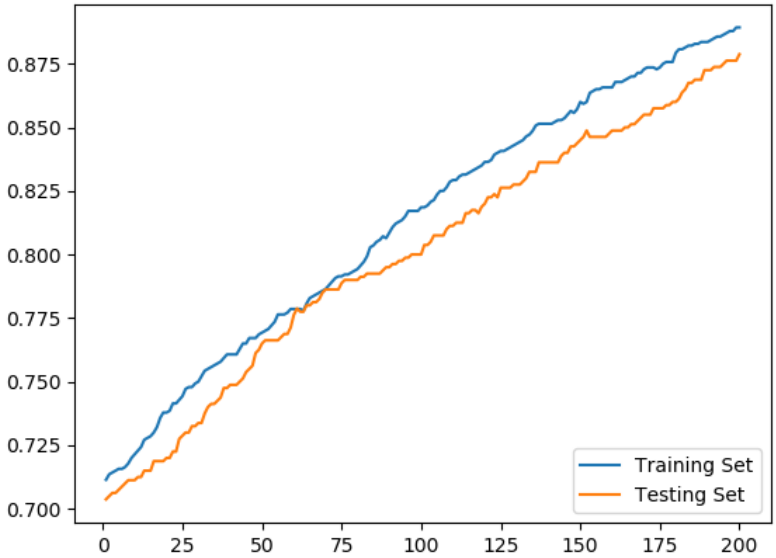
```
while True:
        grad_des = np.zeros(pix_train.shape[1])
        for i in range(pix_train.shape[0]):
                pred_ans = 1.0 / (1.0 + m.exp(-np.dot(np.transpose(w_log), pix_train[i])))
                grad_des += ((pred_ans - ans_train[i]) * pix_train[i]) +
                                (0.5*lambda*|w_log|^2)

        w_log -= (nu*grad_des)
        k += 1
```

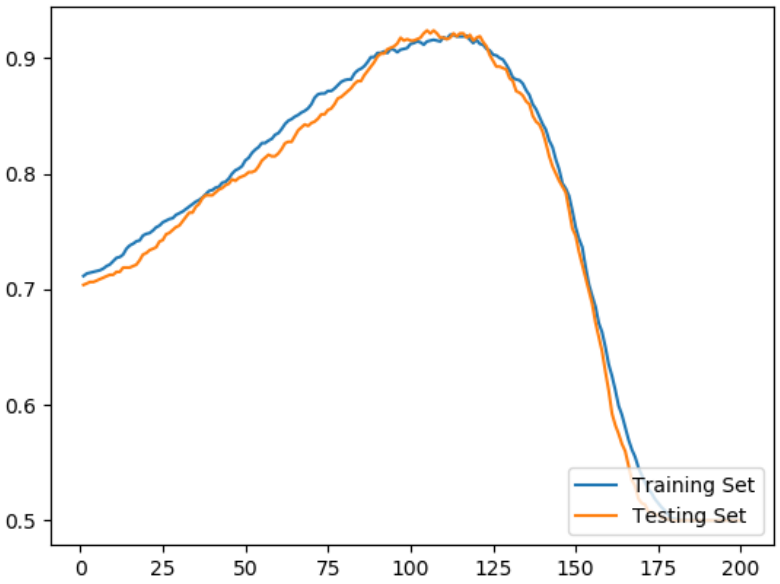
3.



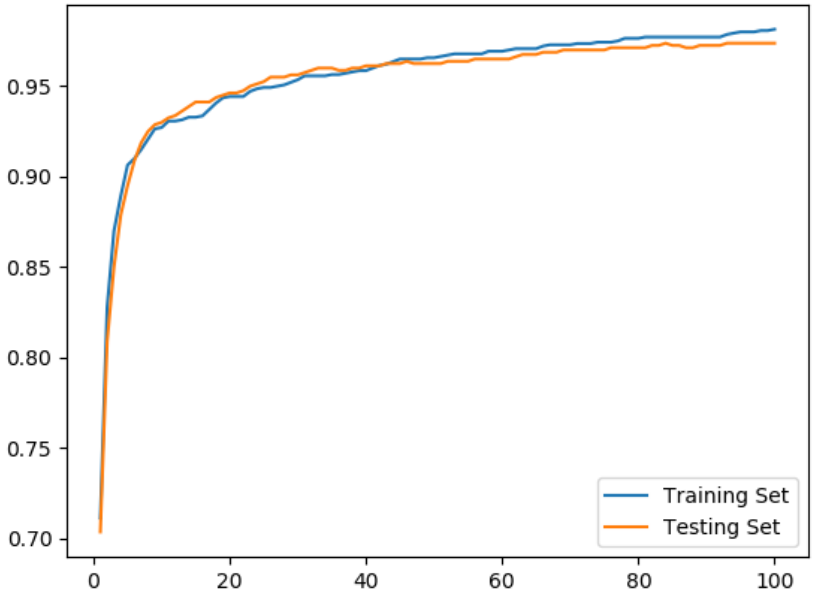
Graph 1: lambda = 0.05



Graph 2: lambda = 1



Graph 3: lambda = 10



Graph 4: lambda = 0.001

Adding L2 regularization to our batch learning algorithm slightly improved our accuracy on both test and training sets, provided lambda was very small. Increasing values of lambda caused an “overcorrection” effect that began to lower accuracy after so many batches. Furthermore, high values of lambda gave overflow errors in the programming process that required either more batches, a smaller learning rate, or both. The larger values oversimplified our model and resulted in poor performance.