CS-589

Konstantinos Pantazis **Homework 2** March 22, 2019

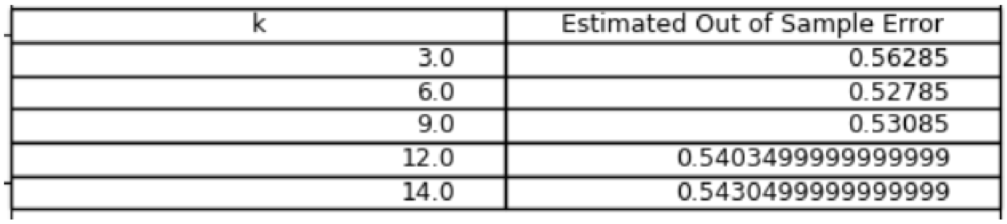
Problem 1

(a) The information gain is closely related with the entropy, more specifically,in order to decide which feature to choose to be in the root node, first compute the total entropy $E(Y) = -\sum_i p_i \log(p_i)$, i.e the entropy of the target Y, and next, compute the corresponding entropy for each feature $E(f_i, Y)$. Finally, the information gain for each feature f_i is given by :

$$Gain(f_i) = E(Y) - E(f_i, Y).$$

After finding the gain for each feature, you choose the one that has the largest information gain value. Note that this procedure continues recursively on each subset D_0, D_1 by subtracting the entropy of each feature ,conditional upon the given subset, from the entropy of the current node, i.e root node.

(b) As the maximum depth gets bigger the training error gets smaller, so the accuracy goes higher. For the out of sample error, if maximum depth is too big we will overfit the training data and the validation error will increase, while, if the maximum depth is too low we will introduce bias and we will underfit the training data. The model with the



k	Estimated Out of Sample Error
3.0	0.56285
6.0	0.52785
9.0	0.53085
12.0	0.5403499999999999
14.0	0.5430499999999999

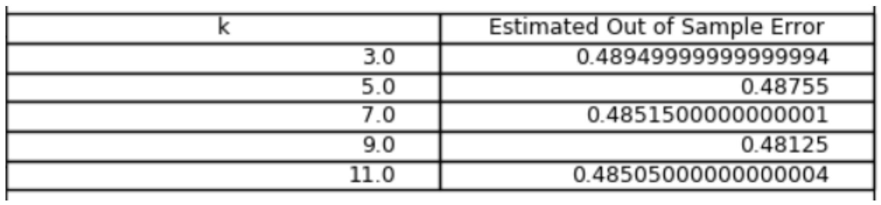
Figure 1: Table corresponding to the maximum depth k and the out of sample error

lowest estimated out of sample error is for $k = 6$ with predicted error 0.52785. The real error(model with k=6 trained on the full set of data) is 0.518(since the accuracy reported on kaggle is 0.482.

Problem 2

(a) First, we calculate the distance of X with each one of the training datum and then we keep the k closest to X data(nearest neighbors). So, for the first step, the complexity is $O(NF)$ since the distance is the sum of square differences between feature values. For the next step, the complexity is $O(kN)$. Thus, the final complexity of the procedure described above is $O(NF + kN) \sim O(NF)$, since most often the number of features is large!

(b) The model with $k = 9$ has the smallest error,so it has the biggest accuracy 0.51875 which is the predicted accuracy. The real accuracy is 0.51149 very close to the predicted one.



k	Estimated Out of Sample Error
3.0	0.48949999999999994
5.0	0.48755
7.0	0.4851500000000001
9.0	0.48125
11.0	0.48505000000000004

Figure 2: Tablet corresponding to number of neighbors k and the out of sample error

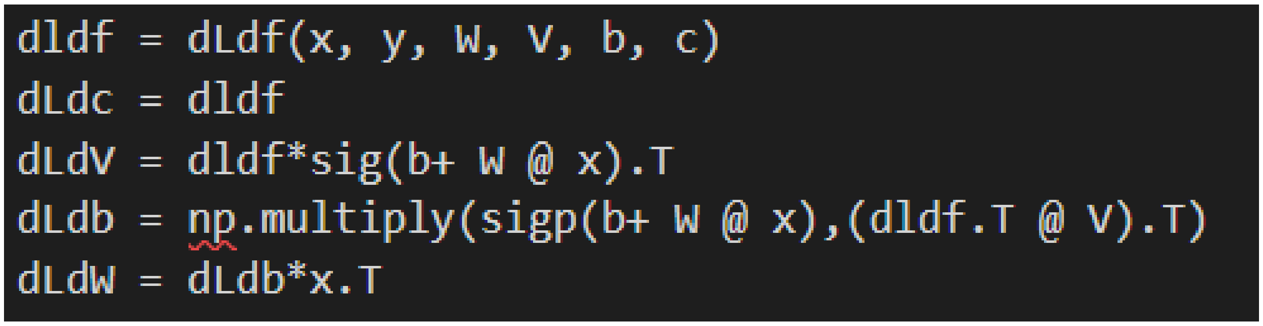
Problem 3

We observe that in general the models with the Hinge loss perform better. We choose the best model to be $alpha = 10$ with predicted accuracy for the Hinge loss equal to 0.4757 and real accuracy 0.5010 which is somewhat not close to the predicted one. Similarly, the best model using Logistic Loss is $alpha = 10$ and the predicted accuracy is 0.4725 and the real accuracy is 0.4900.

alpha	Estimated out of sample error(Hinge)
10^{-6}	0.5432
10^{-4}	0.5409
10^{-2}	0.55955
1	0.57814
10	0.5243
alpha	Estimated out of sample error(Logistic Loss)
10^{-6}	0.60079
10^{-4}	0.57745
10^{-2}	0.56080
1	0.5451
10	0.527500

1 Problem 4(a):

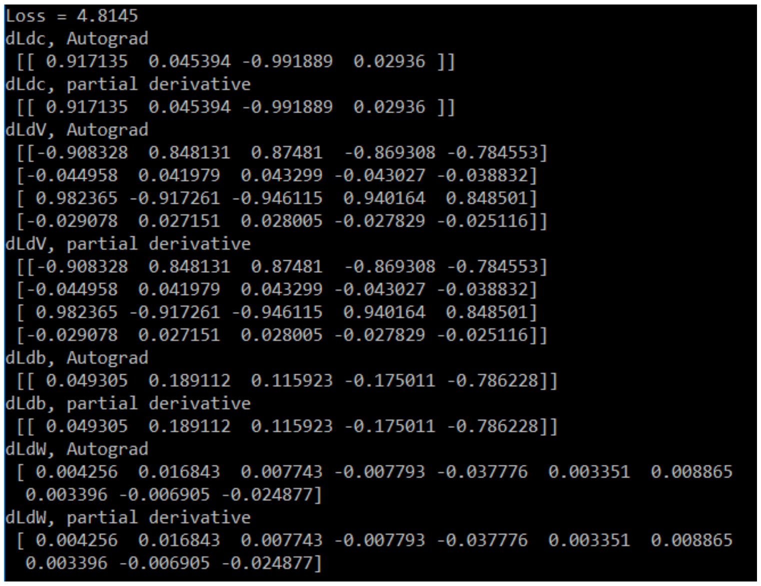
1. -



```
dldf = dLdf(x, y, W, V, b, c)
dLdc = dldf
dLdV = dldf*sig(b+ W @ x).T
dLdb = np.multiply(sigp(b+ W @ x),(dldf.T @ V).T)
dLdW = dLdb*x.T
```

Figure 3: The seeking derivatives $dLdc, dLdV, dLdb, dLdW$.

2. -



```
Loss = 4.8145
dLdc, Autograd
 [[ 0.917135  0.045394 -0.991889  0.02936 ]]
dLdc, partial derivative
 [[ 0.917135  0.045394 -0.991889  0.02936 ]]
dLdV, Autograd
 [[-0.908328  0.848131  0.87481  -0.869308 -0.784553]
 [-0.044958  0.041979  0.043299 -0.043027 -0.038832]
 [ 0.982365 -0.917261 -0.946115  0.940164  0.848501]
 [-0.029078  0.027151  0.028005 -0.027829 -0.025116]]
dLdV, partial derivative
 [[-0.908328  0.848131  0.87481  -0.869308 -0.784553]
 [-0.044958  0.041979  0.043299 -0.043027 -0.038832]
 [ 0.982365 -0.917261 -0.946115  0.940164  0.848501]
 [-0.029078  0.027151  0.028005 -0.027829 -0.025116]]
dLdb, Autograd
 [[ 0.049305  0.189112  0.115923 -0.175011 -0.786228]]
dLdb, partial derivative
 [[ 0.049305  0.189112  0.115923 -0.175011 -0.786228]]
dLdW, Autograd
 [ 0.004256  0.016843  0.007743 -0.007793 -0.037776  0.003351  0.008865
  0.003396 -0.006905 -0.024877]
dLdW, partial derivative
 [ 0.004256  0.016843  0.007743 -0.007793 -0.037776  0.003351  0.008865
  0.003396 -0.006905 -0.024877]
```

Figure 4: The loss and the derivatives computed using autograd and by hand.

1.1 Problem 4(b):

1. -

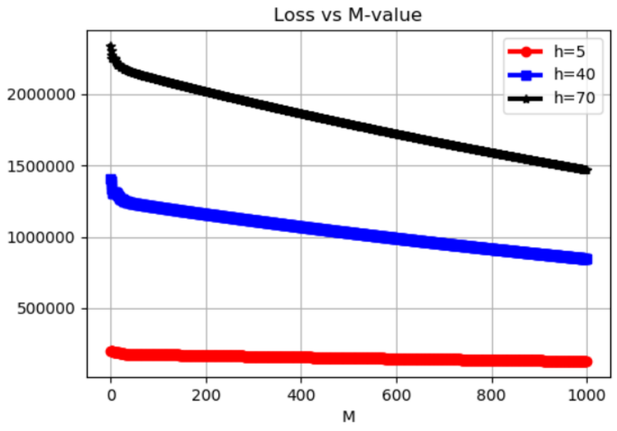


Figure 5: a

2. -

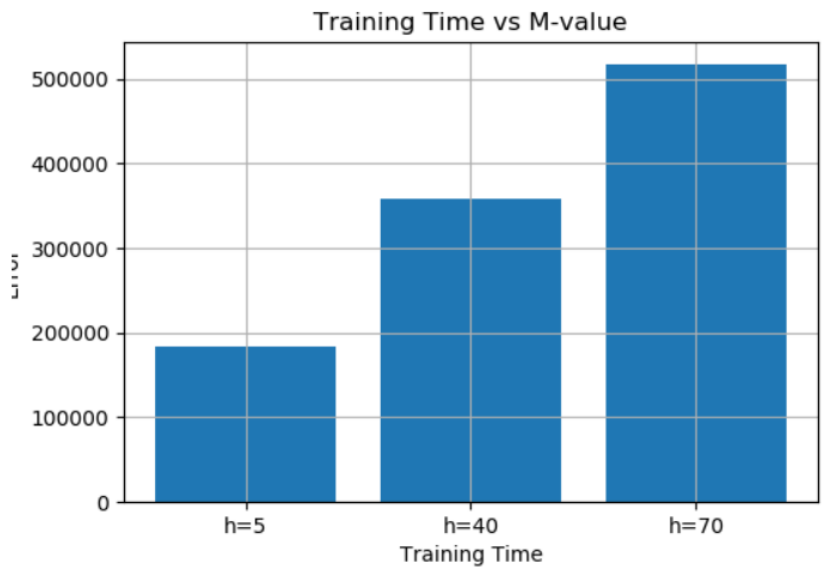


Figure 6: a

3. I did train-validation stratified split but I didn't report it using a table. As the number of hidden units grow the estimated accuracy increases since the weights are more and they are being trained over the epochs. The NN with the lowest estimated validation error is $M = 70$.