CISC684 2019S
HW2 Group Assignment
Group 3

Accuracy Report of Homework 2 Group Assignment

Note about stop words: In our group assignment, we have implemented the "stop word" mechanism as addressed in 13Bayes paper, which is encouraged by Dr. Tong. The stop_words.txt is provided, which contains some prepositions like "and", "of" and other common English words like "what", "a" that worth being skipped. While we process the documents, the words listed in stop_words.txt are skipped and not being calculated for either probability or weight in all three classifiers, which may affect the accuracy compared to those models that didn't remove the stop words. Please note our accuracies are based on the data sets that have already removed the stop words, for all training, validation and test sets.

1. Naïve Bayes:

dataset 1	348/478 (73%)
dataset 2	307/456 (67%)
dataset 3	391/543 (72%)

2. Logistic Regression:

For LR, we have drawn in a variable called weight_weaken_scaler, whose value can be modified at line 253 in our code. Basically, this variable is used to avoid math.exp(weighted_sum) math range error in python if the calculated weighted_sum is too large, by dividing the weight values in the weight_vector by weight_weaken_scaler after each iteration (on line 194), especially when training iterations is set to be very large. For example, if the updated weight for word "subject" after one iteration is 1600, and if we set weight_weaken_scaler = 1000, then the weight of "subject" becomes 1.6. We can do this because LR uses a sigmoid function in which the weighted_sum are only considered for its degree of sign.

We set the weight_weaken_scaler to 500, learning_rate to 0.1, and training_iterations to 4, and select the best lambda from (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10) by testing on the validation data. We set training_iterations to 4 and weight_weaken_scaler to 500 because during testing, this value pair won't cause the math range error on all 3 data sets and the accuracy we get for each lambda can at least have some difference.

Here are the accuracies on the validation data under each lambda value and the accuracy on the whole test data by learning at the best lambda on the whole training set.

weight_weaken_scaler = 500
learning_rate = 0.1
training_iterations = 4

dataset 1

lamda	0	1	2	3	4	5	6	7	8	9	10
accuracy	104/140 (74%)	104/140 (74%)	105/140 (75%)	105/140 (75%)	105/140 (75%)	105/140 (75%)	105/140 (75%)	106/140 (76%)	106/140 (76%)	106/140 (76%)	107/140 (76%)

From the table above the best choice of lambda is 10

Accuracy of lambda = 10 on the whole training and test set: 348/478 (73%)

dataset 2

lamda	0	1	2	3	4	5	6	7	8	9	10
accuracy	99/136 (73%)	100/136 (74%)	100/136 (74%)	100/136 (74%)	102/136 (75%)	102/136 (75%)	102/136 (75%)	102/136 (75%)	104/136 (76%)	105/136 (77%)	106/136 (78%)

From the table above the best choice of lambda is also 10

Accuracy of lambda = 10 on the whole training and test set: 307/456 (67%)

dataset 3

lamda	0	1	2	3	4	5	6	7	8	9	10
accuracy	140/161 (87%)	140/161 (87%)	140/161 (87%)	140/161 (87%)	140/161 (87%)	140/161 (87%)	140/161 (87%)	140/161 (87%)	140/161 (87%)	140/161 (87%)	140/161 (87%)

From the table above the best choice of lambda can be any, because it seems the algorithm converges under these 3 values for any lambda.

We choose lambda = 0 to test on the whole test set, accuracy: 472/543 (87%)

Plus:

Here are some abandoned accuracy data for dataset1 and dataset2 when we set

weight_weaken_scaler = 100
learning_rate = 0.1
training_iterations = 4

We abandoned these data because setting these 3 values will cause math range error on dataset3, but we think it's worth mentioning because the accuracy data we get for each lambda can be very different under these values and then there was a clear choice of the best lambda for testing the accuracy on the whole test set.

dataset 1

lamda	0	1	2	3	4	5	6	7	8	9	10
accuracy	116/140 (83%)	119/140 (85%)	121/140 (86%)	119/140 (85%)	120/140 (86%)	134/140 (96%)	118/140 (84%)	78/140 (56%)	45/140 (32%)	39/140 (28%)	37/140 (26%)

From the table above the best choice of lambda is 5

Accuracy of lambda = 5 on the whole training and test set: 348/478 (73%)

<u>dataset 2</u>

lamda	0	1	2	3	4	5	6	7	8	9	10
accuracy	111/136 (82%)	120/136 (88%)	124/136 (91%)	120/136 (88%)	114/136 (84%)	123/136 (90%)	124/136 (91%)	70/136 (51%)	44/136 (32%)	41/136 (30%)	40/136 (29%)

From the table above, the best choice of lambda is either 2 or 6

We choose lambda = 2, accuracy on the whole training and test set: 307/456 (67%)

<u>dataset 3</u>

Math range error…

It may also be worth mentioning that when we set these values to

weight_weaken_scaler = 10000
learning_rate = 0.003
training_iterations = 100

The accuracy we get for each lambda for each data set is the same, which seemed to us that the algorithm converges under those extreme values. (We grabbed those results from running our code at Google Cloud VM.)

The accuracy we get for dataset 1 under each and every lambda is 103/140 (74%), and final accuracy at the whole test set is 348/478 (73%). For dataset2 it is 97/136 (71%) on validation set under each lambda and for the whole test set it is 308/456 (68%). For dataset3 it is 136/161 (84%) on validation set under each lambda and for the whole test set it is 475/543 (87%).

Our code has all these processes automated to support this accuracy report. Please go to line 251~254 to set the training_iterations, learning_rate, weight_weaken_scaler and regularization_lambda_values variables to the values we set above to replicate our results or set to other values to test our program for grading.

3. Perceptron

For perceptron, besides testing for the best choice of training iterations, we also tested the hyperparameter learning rate.

Here are the accuracy data with the chosen training_iterations and learning_rate on the validation data:

<u>dataset 1</u>

Training Iterations	Learning rate	Accuracy on validation set
10	0.003	136/140 (97%)
	0.01	137/140 (98%)
	0.03	137/140 (98%)

Training Iterations	Learning rate	Accuracy on validation set
20	0.003	132/140 (94%)
	0.01	136/140 (97%)
	0.03	136/140 (97%)

Training Iterations	Learning rate	Accuracy on validation set
50	0.003	136/140 (97%)
	0.01	138/140 (99%)
	0.03	137/140 (98%)

From the table above, the best choice of iterations is 50 and learning rate is 0.01.

With the above chosen value pair, the accuracy on the whole test set is 450/478 (94%).

dataset 2

Training Iterations	Learning rate	Accuracy on validation set
10	0.003	135/136 (99%)
	0.01	135/136 (99%)
	0.03	136/136 (100%)

Training Iterations	Learning rate	Accuracy on validation set
20	0.003	134/136 (99%)
	0.01	135/136 (99%)
	0.03	136/136 (100%)

Training Iterations	Learning rate	Accuracy on validation set
50	0.003	131/136 (96%)
	0.01	135/136 (99%)
	0.03	136/136 (100%)

From the table above, the best choice of iterations is any, but learning rate under all iterations is 0.03.

With the choice of learning iterations 10 and learning rate 0.03, the accuracy on the whole test set is 421/456 (92%).

dataset 3

Training Iterations	Learning rate	Accuracy on validation set
10	0.003	160/161 (99%)
	0.01	160/161 (99%)
	0.03	160/161 (99%)

Training Iterations	Learning rate	Accuracy on validation set

20	0.003	161/161 (100%)
	0.01	159/161 (99%)
	0.03	159/161 (99%)

Training Iterations	Learning rate	Accuracy on validation set
50	0.003	161/161 (100%)
	0.01	160/160 (99%)
	0.03	160/160 (99%)

From the table above, the best choice of iterations can be either 20 or 50 with the learning rate 0.003.

With the choice of learning iterations 20 and learning rate 0.003, the accuracy on the whole test set is 500/543 (92%).

Again, please go to the line 272 and 273 in our code to alternative the values of training_iterations_values and learning_rate_values to replicate the results we have above for perceptron or set to other values to test our program for grading.

Thank you.
Group3
April 4, 2019