Report

Arvind Krishna Parthasarathy

Axp133230 HW3

Perceptron classifier:

This is a linear classifier which converges to an optimal model based on the number of iterations and learning rate.

Here I ran 20 different pairs of these parameters and these are the accuracies obtained on the three datasets.

True and false determine if the stop words were removed from the dataset.

As we can see, for extremely slow learning rate and low number of iterations, the classifier does not converge. When I used 0.4 as my learning rate and 120 as my number of iterations, I was able to get 96% on the third dataset.

Naïve Bayes accuracy – without removal of stop words – 95.188%

			DataSet 1		Enron1		Enron3		Logistic Regression
S No.	Learning Rate	Num Iterations	True	False	True	False	True	False	Accuracy
1	0.1	80	92.26	91.63	91.09	91	94.84	93.222	82.008
2	0.2	50	93.30	93.09	91.22	92.33	94.84	93.922	79.707
3	0.3	100	93.30	92.67	92.32	92.105	93.922	94.84	79.479
4	0.5	90	93.09	92.88	92.32	92.54	95.55	95.03	78.242
5	0.4	120	92.05	93.72	92.15	92.76	95.211	95.58	78.870
6	0.8	180	93.30	94.35	92.15	91.22	95.39	95.4	77.615
7	0.55	100	92.67	94.33	91.22	91.88	94.47	95.02	78.242
8	0.1	160	94.14	92.88	91.66	91.66	94.84	95.39	82.675
9	0.1	20	94.14	93.30	89.52	91.88	93.18	94.65	81.171
10	0.1	3	66.94	65.65	57.7	69.29	85.18	84.39	72.594
11	0.2	3	69.03	69.03	68.42	68.85	83.9	83.3	72.803
12	0.1	300	93.30	93.72	91.00	92.54	94.29	94.28	82.635
13	0.2	150	93.30	93.09	91.88	91.88	95.02	94.29	80.962
14	0.1	1500	93.09	92.82	92.105	92.32	95.39	94.58	83.682
15	0.05	1000	90.16	90.16	91.44	89.69	93.73	93.53	85.355
16	0.5	60	93.30	93.31	89.91	91.66	95.01	95.211	78.242
17	0.9	90	93.72	93.93	91.44	91.66	95.94	95.39	77.405
18	0.6	45	93.51	93.9	92.10	91.88	95.76	94.65	77.405
19	0.4	60	93.50	93.51	92.76	91.22	95.02	94.10	78.451
20	0.2	80	91.84	92.25	90.78	91.88	95.84	95.58	79.916

Collaborative Filtering:

This is a memory based algorithm that predicts a user's rating based on his average rating and based on the weights assigned to each user depending on the similarity in the ratings provided.

Formula used:

$$w(a,i) = \frac{\sum_j (v_{a,j} - \overline{v}_a)(v_{i,j} - \overline{v}_i)}{\sqrt{\sum_j (v_{a,j} - \overline{v}_a)^2 \sum_j (v_{i,j} - \overline{v}_i)^2}}$$

where the summations over j are over the items for which both users a and i have recorded votes.

Given this huge dataset, it is natural that the calculation of weights for each user will take some time as there are 27000+ users on the record.

It performed pretty well on the Netflix dataset with the following error rates on these metrics.

Results:

Average Mean absolute error: 0.7902268915281676

Average Root mean square error: 0.9886979385869478

PTO

Neural Networks

No	Hidden Layers	Hidden Units (Separated by Commas)	Accuracy (hw2 dataset)	Accuracy (enron1 dataset)	Accuracy (enron4 dataset)
1	1	1	93.9	92.9	96.3
2	1	2	95.4	62.5	72.00
3	1	3	94.6	92.8	96.0
4	1	4	93.5	93.6	72.0
5	1	5	92.7	92.10	68.0
6	2	1,1	80.1	67.3	72.0
7	2	1,2	72.20	67.30	71.9
8	2	2,2	72.00	78.0	79.13
9	3	4,2,3	71.2	66.4	73.00
10	3	2,1,2	72.8	68.42	30.38
11	3	4,4,3	72.8	69.51	72.00
12	3	4,2,3	81.79	67.32	72.00

As we can see, as the number of hidden layers increase, the classifier overfits. This results in decrease in accuracy.

So, we have to keep the complexity of the structure to as small as possible. If we can represent a function using just one layer of hidden nodes, then we should always choose that over other complex models, even though the training accuracy increases as we increase the number of hidden layers.

I have added the commands to convert the files to arff format to the zip file for this project. Please execute them one by one.