Homework-1

Naive Bayes and Logistic Regression for Text Classification

Dataset

Given Dataset Summery

Sl no	Dataset	Train		Test	
		Ham	Spam	Ham	Spam
1	Hw1	340	123	348	130
2	Enron1	319	131	307	149
3	Enron4	133	402	152	391

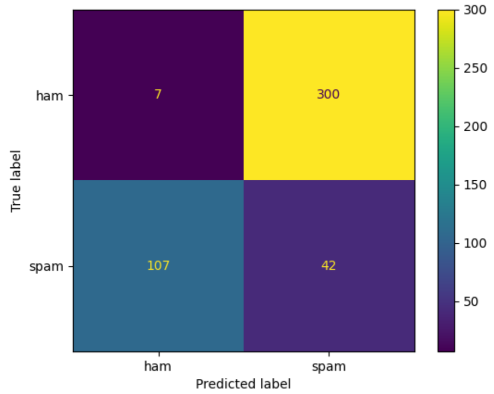
Multinomial Naive Bayes on the Bag of words model

Accuracy of hw1 performed the best between the 3 datasets. Assumption here is that features are conditionally independent.

Sl no	Dataset	Dictionary	Recall	Precision	Accuracy	F1-Score
1	Hw1_test	9133	0.9307692308	0.9097744361	0.9560669456	0.9201520913
2	Enron1_test	8852	0.8456375839	0.9402985075	0.9320175439	0.890459364
3	Enron4_test	16502	0.9462915601	0.9585492228	0.9318600368	0.9523809524

Observations:

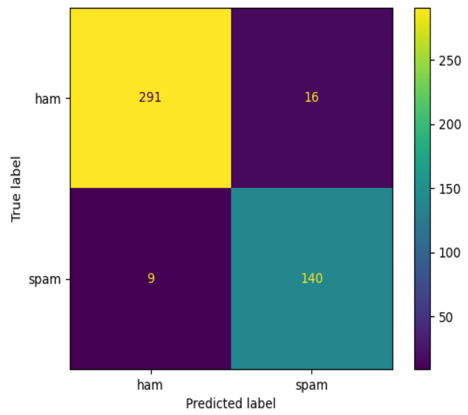
Without considering Laplacian smooth we run into multiple log (0) => -inf cases as the many features have count 0, there are not enough data for this feature to make any kind of prediction. Here we encounter the “Problem of zero probability”, From below result and Metrics we can see the problem taking shape.



Accuracy	0.1074561403
Precision	0.1228070175
Recall	0.2818791946
F1 Score	0.1710794297

NOTE: Above results are obtained by ignoring log (0) probability values (anyway it was -inf)

Considering one Laplacian



From the above results it is clear that Laplacian smoothing helps in eliminating the “Problem of zero probability”.

Accuracy	0.9451754386
Precision	0.8974358974
Recall	0.9395973154
F1 Score	0.9180327869

False positives has reduced more than half but at cost of double the False negatives. But overall F1 score has increased which means FN & FP combined have become better.

Discrete Naive Bayes on the Bernoulli model

Enron4 dataset performed better. Assumption here is that there is only 1 occurrence of a word in one email and also features are conditionally independent.

Sl no	Dataset	Dictionary	Recall	Precision	Accuracy	F1-Score
1	Hw1_test	9133	0.7538461538	0.9607843137	0.9246861925	0.8448275862
2	Enron1_test	8852	0.6711409396	0.9803921569	0.8881578947	0.796812749
3	Enron4_test	16502	0.9923273657	0.9371980676	0.9465930018	0.9639751553

MCAP Logistic Regression algorithm with L2 regularization

Logistic Regression is being run on both Bernoulli model and the Bag of words model. Using Cross validation with train dataset split of 70% as training and 30% as Validation dataset to tune the hyper parameter.

Hyper parameters which were tuned using cross validation are

1. Learning rate

   All learning rates considered were: [0.1, 0.01, 0.001]

2. Lambda

   All lambda values considered: [0, 0.05, 0.005, 0.001, 0.0005]

3. iterations

   Number of Iteration: [100, 500, 1000]

Total Observations observed for Logistic regression while cross validation:

2 (features) * 3 (learning rate) * 5 (lambda values) * 3 (iterations) = 90

For each parameter 70% of the data was trained and tested with 30% validation split and the result was used and the best accuracy for the validation set was selected for hyper parameter. Out of all the results top 5 were selected and tested against test dataset and the results are shown below for Bernoulli and Bag of Words features.

Bernoulli feature set Observations

Enron1 dataset

Top 5 results were considered, weirdly or not lambda value with 0 performed well in cross validation

CROSS_VALIDATION with 70% train and 30% validation

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	0.8717948718	0.8947368421	0.9328358209	0.8831168831
500	0.1	0.001	0.8461538462	0.8918918919	0.9253731343	0.8684210526
500	0.1	0.0005	0.8461538462	0.8918918919	0.9253731343	0.8684210526

1000	0.1	0.001	0.8461538462	0.8918918919	0.9253731343	0.8684210526
1000	0.1	0	0.8461538462	0.8684210526	0.9179104478	0.8571428571

Final Result

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	0.9261744966	0.9078947368	0.9451754386	0.9169435216
500	0.1	0.002	0.9194630872	0.9072847682	0.9429824561	0.9133333333
500	0.1	0.001	0.9261744966	0.9078947368	0.9451754386	0.9169435216
1000	0.1	0.002	0.9261744966	0.9078947368	0.9451754386	0.9169435216
1000	0.1	0	0.9194630872	0.9072847682	0.9429824561	0.9133333333

Enron4 dataset

Top 5 results were considered, weirdly or not lambda value with 0 performed well in cross validation

CROSS_VALIDATION with 70% train and 30% validation

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	1	0.9448818898	0.9559748428	0.971659919
500	0.1	0.002	1	0.9448818898	0.9559748428	0.971659919
500	0.1	0.001	1	0.9448818898	0.9559748428	0.971659919
1000	0.1	0	1	0.9448818898	0.9559748428	0.971659919
1000	0.1	0.001	1	0.9448818898	0.9559748428	0.971659919

Final Result

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	1	0.9467312349	0.9594843462	0.9726368159
500	0.1	0.001	1	0.9467312349	0.9594843462	0.9726368159
500	0.1	0.0005	1	0.9467312349	0.9594843462	0.9726368159
1000	0.1	0	1	0.9467312349	0.9594843462	0.9726368159
1000	0.1	0.001	1	0.9513381995	0.9631675875	0.9750623441

Hw1 dataset

Top 5 results were considered, weirdly or not lambda value with 0 performed well in cross validation

CROSS_VALIDATION with 70% train and 30% validation

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	0.8333333333	0.8823529412	0.9275362319	0.8571428571
500	0.1	0.001	0.8333333333	0.8823529412	0.9275362319	0.8571428571
500	0.1	0.0005	0.8333333333	0.8823529412	0.9275362319	0.8571428571
1000	0.1	0	0.8333333333	0.8823529412	0.9275362319	0.8571428571
1000	0.1	0.001	0.8333333333	0.8823529412	0.9275362319	0.8571428571

Final Result

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	0.8307692308	0.9152542373	0.9330543933	0.8709677419
500	0.1	0.001	0.8307692308	0.9152542373	0.9330543933	0.8709677419
500	0.1	0.0005	0.8307692308	0.9152542373	0.9330543933	0.8709677419
1000	0.1	0	0.8923076923	0.9133858268	0.9476987448	0.9027237354
1000	0.1	0.001	0.8923076923	0.9133858268	0.9476987448	0.9027237354

Bag of Words feature set Observations

Enron1 dataset

Top 5 results were considered, weirdly or not lambda value with 0 performed well in cross validation

CROSS_VALIDATION with 70% train and 30% validation

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	0.8717948718	0.8947368421	0.9328358209	0.8831168831
500	0.1	0.001	0.8461538462	0.8918918919	0.9253731343	0.8684210526
500	0.1	0.0005	0.8461538462	0.8918918919	0.9253731343	0.8684210526
1000	0.1	0	0.8461538462	0.8684210526	0.9179104478	0.8571428571
1000	0.1	0.001	0.8461538462	0.8684210526	0.9179104478	0.8571428571

Final Result

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	0.9261744966	0.9078947368	0.9451754386	0.9169435216
500	0.1	0.001	0.9194630872	0.9072847682	0.9429824561	0.9133333333
500	0.1	0.0005	0.9261744966	0.9078947368	0.9451754386	0.9169435216
1000	0.1	0	0.9261744966	0.9019607843	0.9429824561	0.9139072848
1000	0.1	0.001	0.9261744966	0.9019607843	0.9429824561	0.9139072848

Enron4 dataset

Top 5 results were considered, weirdly or not lambda value with 0 performed well in cross validation

CROSS_VALIDATION with 70% train and 30% validation

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	1	0.9448818898	0.9559748428	0.971659919
500	0.1	0.001	1	0.9448818898	0.9559748428	0.971659919
500	0.1	0.0005	1	0.9448818898	0.9559748428	0.971659919
1000	0.1	0	1	0.9448818898	0.9559748428	0.971659919
1000	0.1	0.001	1	0.9448818898	0.9559748428	0.971659919

Final Result

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	1	0.9467312349	0.9594843462	0.9726368159
500	0.1	0.001	1	0.9467312349	0.9594843462	0.9726368159
500	0.1	0.0005	1	0.9467312349	0.9594843462	0.9726368159

1000	0.1	0	1	0.9513381995	0.9631675875	0.9750623441
1000	0.1	0.001	1	0.9513381995	0.9631675875	0.9750623441

Hw1 dataset

Top 5 results were considered, weirdly or not lambda value with 0 performed well in cross validation

CROSS_VALIDATION with 70% train and 30% validation

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	0.8333333333	0.8823529412	0.9275362319	0.8571428571
500	0.1	0.001	0.8333333333	0.8823529412	0.9275362319	0.8571428571
500	0.1	0.0005	0.8333333333	0.8823529412	0.9275362319	0.8571428571
1000	0.1	0	0.8333333333	0.8823529412	0.9275362319	0.8571428571
1000	0.1	0.001	0.8333333333	0.8823529412	0.9275362319	0.8571428571

Final Result

Iterations	Learning Rate	Lambda	Recall	Precision	Accuracy	F1_score
500	0.1	0	0.8307692308	0.9152542373	0.9330543933	0.8709677419
500	0.1	0.001	0.8307692308	0.9152542373	0.9330543933	0.8709677419
500	0.1	0.0005	0.8307692308	0.9152542373	0.9330543933	0.8709677419
1000	0.1	0	0.9	0.9140625	0.949790795	0.9069767442
1000	0.1	0.001	0.8923076923	0.9133858268	0.9476987448	0.9027237354

SGDClassifier from scikit-learn

SGDClassifier is being run on both Bernoulli features and the Bag of words features.

Hyper parameters which were tuned using cross validation are

1. Learning rate

   All learning rates considered were: ['optimal', 'invscaling', 'adaptive']
2. Alpha

   All lambda values considered: [0, 0.05, 0.005, 0.001, 0.0005]
3. iterations

   Number of Iteration: [100, 500, 1000]

Using GridSearchCV and above parameters constraints hyper parameters was obtained.

Below 2 section describes the results obtained from the above conditions

Bernoulli feature set Observations

Enron1 dataset

Running GridSearchCV gave the following hyper parameters

Max_iter	Learning Rate	Alpha	tol	loss	penalty
100	optimal	0.005	1e-3	Log_loss	L2

Final Result

Max_iter	Learning Rate	Alpha	Recall	Precision	Accuracy	F1_score
100	optimal	0.005	0.9395973154	0.9150326797	0.951754386	0.9271523179

Enron4 dataset

Running GridSearchCV gave the following hyper parameters

Max_iter	Learning Rate	Alpha	tol	loss	penalty
500	optimal	0.005	1e-3	Log_loss	L2

Final Result

Max_iter	Learning Rate	Alpha	Recall	Precision	Accuracy	F1_score
500	optimal	0.005	0.9948849105	0.9628712871	0.9686924494	0.9786163522

Hw1 dataset

Running GridSearchCV gave the following hyper parameters

Max_iter	Learning Rate	Alpha	tol	loss	penalty
100	optimal	0.05	1e-3	Hinge	L2

Final Result

Max_iter	Learning Rate	Alpha	Recall	Precision	Accuracy	F1_score
100	optimal	0.05	0.9230769231	0.8955223881	0.949790795	0.9090909091

Bag of Words feature set Observations

Enron1 dataset

Running GridSearchCV gave the following hyper parameters

Max_iter	Learning Rate	Alpha	tol	loss	penalty
1000	Optimal	0.005	1e-3	Log_loss	L2

Final Result

Max_iter	Learning Rate	Alpha	Recall	Precision	Accuracy	F1_score
1000	Optimal	0.005	0.9530201342	0.9161290323	0.9561403509	0.9342105263

Enron4 dataset

Running GridSearchCV gave the following hyper parameters

Max_iter	Learning Rate	Alpha	tol	loss	penalty
100	Optimal	0.005	1e-3	Log_loss	L2

Final Result

Max_iter	Learning Rate	Alpha	Recall	Precision	Accuracy	F1_score
100	Optimal	0.005	0.9948849105	0.9604938272	0.9668508287	0.9773869347

Hw1 dataset

Running GridSearchCV gave the following hyper parameters

Max_iter	Learning Rate	Alpha	tol	loss	penalty

500	Optimal	0.005	1e-3	Hinge	L2

Final Result

Max_iter	**Learning Rate**	**Alpha**	**Recall**	**Precision**	**Accuracy**	**F1_score**
500	Optimal	0.005	0.9153846154	0.937007874	0.960251046	0.9260700389

Questions

1. Which data representation and algorithm combination yields the best performance (measured in terms of the accuracy, precision, recall and F1 score) and why?

Model Type	**Dataset**	**Recall**	**Precision**	**Accuracy**	**F1_score**
Bag of Words	Enron1	0.8456375839	0.9402985075	0.9320175439	0.890459364
Bernoulli	Enron1	0.6711409396	0.9803921569	0.8881578947	0.796812749
LR (Bernoulli)	Enron1	0.9261744966	0.9078947368	0.9451754386	0.9169435216
LR (Bag of Words)	Enron1	0.9261744966	0.9078947368	0.9451754386	0.9169435216
SGD (Bernoulli)	Enron1	0.9395973154	0.9150326797	0.951754386	0.9271523179
SGD (Bag of Words)	Enron1	0.9530201342	0.9161290323	0.9561403509	0.9342105263

Model Type	**Dataset**	**Recall**	**Precision**	**Accuracy**	**F1_score**
Bag of Words	Hw1	0.9307692308	0.9097744361	0.9560669456	0.9201520913
Bernoulli	Hw1	0.7538461538	0.9607843137	0.9246861925	0.8448275862
LR (Bernoulli)	Hw1	0.8923076923	0.9133858268	0.9476987448	0.9027237354
LR (Bag of Words)	Hw1	0.9	0.9140625	0.949790795	0.9069767442
SGD (Bernoulli)	Hw1	0.9230769231	0.8955223881	0.949790795	0.9090909091
SGD (Bag of Words)	Hw1	0.9153846154	0.937007874	0.960251046	0.9260700389

Model Type	**Dataset**	**Recall**	**Precision**	**Accuracy**	**F1_score**
Bag of Words	Enron4	0.9462915601	0.9585492228	0.9318600368	0.9523809524
Bernoulli	Enron4	0.9923273657	0.9371980676	0.9465930018	0.9639751553
LR (Bernoulli)	Enron4	1	0.9513381995	0.9631675875	0.9750623441
LR (Bag of Words)	Enron4	1	0.9513381995	0.9631675875	0.9750623441
SGD (Bernoulli)	Enron4	0.9948849105	0.9628712871	0.9686924494	0.9786163522
SGD (Bag of Words)	Enron4	0.9948849105	0.9604938272	0.9668508287	0.9773869347

From the Above analysis, Logistic Regression and Bag of Words is able to generalise the data better. Cause might be that occurrence of certain words matters in terms of spam and ham detection, ex: commonly spam messages are mostly likely repeat certain words to divert attention such as “Offer!!! Offer !!!” etc, where as in ham, might just say “we offer these services to our clients”. Logistic regression is able to classify better due to the fact that features might not always be conditionally independent which is the assumption for Naive Based Classifiers and also not always linearly separable.

2. Does Multinomial Naive Bayes perform better (again performance is measured in terms of the accuracy, precision, recall and F1 score) than LR and SGDClassifier on the Bag of words representation? Explain your yes/no answer.

   No, based on the data provided able. It does give good accuracy but since both LR and SGDClassifier use the same features to fit the model without the conditional independence assumption.

3. Does Discrete Naive Bayes perform better (again performance is measured in terms of the accuracy, precision, recall and F1 score) than LR and SGDClassifier on the Bernoulli representation? Explain your yes/no answer.

   No. Since, Discreate Naïve Bayes suffers from Assumptions such as only 1 time consideration of the word in the email and also the conditional independence might be the reason behind low performance.

4. Does your LR implementation outperform the SGDClassifier (again performance is mea sured in terms of the accuracy, precision, recall and F1 score) or is the difference in performance minor? Explain your yes/no answer.

Model Type	Dataset	Recall	Precision	Accuracy	F1_score
LR (Bernoulli)	Hw1	0.8923076923	0.9133858268	0.9476987448	0.9027237354
LR (Bag of Words)	Hw1	0.9	0.9140625	0.949790795	0.9069767442
SGD (Bernoulli)	Hw1	0.9230769231	0.8955223881	0.949790795	0.9090909091
SGD (Bag of Words)	Hw1	0.9153846154	0.937007874	0.960251046	0.9260700389

Model Type	Dataset	Recall	Precision	Accuracy	F1_score
LR (Bernoulli)	Enron1	0.9261744966	0.9078947368	0.9451754386	0.9169435216
LR (Bag of Words)	Enron1	0.9261744966	0.9078947368	0.9451754386	0.9169435216
SGD (Bernoulli)	Enron1	0.9395973154	0.9150326797	0.951754386	0.9271523179
SGD (Bag of Words)	Enron1	0.9530201342	0.9161290323	0.9561403509	0.9342105263

Model Type	Dataset	Recall	Precision	Accuracy	F1_score
LR (Bernoulli)	Enron4	1	0.9513381995	0.9631675875	0.9750623441
LR (Bag of Words)	Enron4	1	0.9513381995	0.9631675875	0.9750623441
SGD (Bernoulli)	Enron4	0.9948849105	0.9628712871	0.9686924494	0.9786163522
SGD (Bag of Words)	Enron4	0.9948849105	0.9604938272	0.9668508287	0.9773869347

No, My LR implementation does not outperform SGDClassifier. For dataset Enron4, the results might be minor but for other dataset, it is not able to generalise as good as SGDClassifier.

My Opinion: SGDClassifier massively outperform my LR implementation, as I believe after certain accuracy, it becomes harder and harder to generalize, every % improvement after certain point needs lot of effort and knowledge.

Extra Credit

The paper does not open at all