Jashan Shah

CS 6375.003

Dr. Vibhav Gogate

9/20/2022

Homework 1

Machine Learning

Naive Bayes and Logistic Regression for Text Classification

I did Text Classification using Naive Bayes and Logistic Regression by following steps as mentioned below:

1. Convert the text data into a matrix of features × examples (namely our canonical data representation), using the following approaches. – Bag of Words model: Recall that we use a vocabulary having w words—the set of all unique words in the training set—and represent each email using a vector of word frequencies (the number of times each word in the vocabulary appears in the email). – Bernoulli model: As before we use a vocabulary having w words and represent each email (training example) using a 0/1 vector of length w where 0 indicates that the word does not appear in the email and 1 indicates that the word appears in the email.
2. Implement the multinomial Naive Bayes algorithm for text classification described here: http://nlp.stanford.edu/IR-book/pdf/13bayes.pdf (see Figure 13.2 in the document). Note that the algorithm uses add-one laplace smoothing. Make sure that you do all the calculations in log-scale to avoid underflow. Use your algorithm to learn from the training set and report accuracy on the test set. Important: Use the datasets generated using the Bag of words model and not the Bernoulli model for this part.
3. Implement the discrete Naive Bayes algorithm we discussed in class. To prevent zeros, use add-one laplace smoothing. Make sure that you do all the calculations in log-scale to avoid underflow. Use your algorithm to learn from the training set and report accuracy on the test set. Important: Use the datasets generated using the Bernoulli model and not the Bag of words model for this part.
4. Implement the MCAP Logistic Regression algorithm with L2 regularization that we discussed in class (see Mitchell’s new book chapter). Try different values of λ. Divide the given training set into two sets using a 70/30 split (namely the first split has 70% of the examples and the second split has the remaining 30%). Learn parameters using the 70% split, treat the 30% data as validation data and use it to select a value for λ. Then, use the chosen value of λ to learn the parameters using the full training set and report accuracy on the test set. Use gradient ascent for learning the weights (you have to set the learning rate appropriately. Otherwise, your algorithm may diverge or take a long time to converge). Do not run gradient ascent until convergence; you should put a suitable hard limit on the number of iterations. Important: Use the datasets generated using both the Bernoulli model and the Bag of words model for this part.
5. Run the SGDClassifier from scikit-learn on the datasets. Tune the parameters (e.g., loss function, penalty, etc.) of the SGDClassifier using GridSearchCV in scikit-learn. Compare the results you obtain for SGDClassifier with your implementation of Logistic Regression. Important: Use the datasets generated using both the Bernoulli model and the Bag of words model for this part.

Note: Some files were not encoded in UTF - 8.

Naive Bayes - Bag of words and Bernoulli representation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Datasets | Representation & Algorithm | Accuracy | Precision | Recall | F1 score |
| Hw1 | Multinomial Naive Bayes & Bag of words | 0.9414 | 0.9282 | 0.9908 | 0.9585 |
| Enron1 | Multinomial Naive Bayes & Bag of words | 0.9408 | 0.9251 | 0.9861 | 0.9546 |
| Enron4 | Multinomial Naive Bayes & Bag of words | 0.6980 | 0.9934 | 0.4810 | 0.6480 |
| Hw1 | Discrete Naïve Bayes & Bernoulli | 0.9017 | 0.8707 | 0.9934 | 0.9280 |
| Enron1 | Discrete Naive Bayes & Bernoulli | 0.9364 | 0.9055 | 0.9999 | 0.9504 |
| Enron4 | Discrete Naive Bayes & Bernoulli | 0.9613 | 0.9408 | 0.9225 | 0.9316 |

Logistic Regression - Bag of words and Bernoulli representation

# of iterations = 1000, lambda = 0.5 for Hw1, 0.9 for others, learning rate = 0.01

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Datasets | Representation & Algorithm | Accuracy | Precision | Recall | F1 score |
| Hw1 | Logistic Regression & Bag of words | 0.9289 | 0.9167 | 0.9846 | 0.9494 |
| Enron1 | Logistic Regression & Bag of words | 0.9122 | 0.9055 | 0.9619 | 0.9328 |
| Enron4 | Logistic Regression & Bag of words | 0.9650 | 0.9407 | 0.9346 | 0.9377 |
| Hw1 | Logistic Regression & Bernoulli | 0.9477 | 0.9569 | 0.9708 | 0.9638 |
| Enron1 | Logistic Regression & Bernoulli | 0.9539 | 0.9543 | 0.9767 | 0.9554 |
| Enron4 | Logistic Regression & Bernoulli | 0.9539 | 0.8355 | 0.9999 | 0.9104 |

SGD Classifier - Bag of words and Bernoulli representation

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Datasets | Representation & Algorithm | Accuracy | Precision | Recall | F1 score |
| Hw1 | SGD Classifier & Bag of words | 0.8891 | 0.8563 | 0.9979 | 0.9183 |
| Enron1 | SGD Classifier & Bag of words | 0.8903 | 0.9316 | 0.9079 | 0.9196 |
| Enron4 | SGD Classifier & Bag of words | 0.9503 | 0.8618 | 0.9562 | 0.9068 |
| Hw1 | SGD Classifier & Bernoulli | 0.9393 | 0.9367 | 0.9789 | 0.9574 |
| Enron1 | SGD Classifier & Bernoulli | 0.9254 | 0.8990 | 0.9892 | 0.9419 |
| Enron4 | SGD Classifier & Bernoulli | 0.9686 | 0.8947 | 0.9927 | 0.9411 |

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

SGD Classifier & Bernoulli seems to perform better than the other options when measured in terms of the accuracy, precision, recall, and F1 score. Logistic Regression & Bernoulli may be better depending on the dataset that is provided and used. Overall, it depends on the datasets on which one will perform better. For example, SGD Classifier & Bernoulli has a greater accuracy than Logistic Regression & Bernoulli for the Enron1 dataset, however for enron4 it is the opposite.

1. 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.

In Multinomial Naïve Bayes, the features are never removed however in SGD Classifier and Logistic Regression, weights could become 0. When this happens, the feature will be removed. This can really affect the F1 score as seen from the results. Multinomial Naïve Bayes outperforms the others sometimes due to it not removing features. The performance is also really dependent on the dataset.

1. 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, logistic regression and SGD Classifier seem to perform better than Discrete Naïve Bayes algorithm measured in terms of accuracy, precision, recall, and F1 score. Discrete Naïve Bayes considers non-occurrences while logistic regression and SGD Classifier do not. The average values of logistic regression and SGD Classifier are greater than Discrete Naïve Bayes.

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

No, LR might outperform SGD Classifier due to good convergence, but they are both close in score. The difference between them in terms of performance is not too much meaning the difference in performance is minor.

PS C:\MLHW1A> python bow\_NB.py ./hw1

TP : 323

TN : 127

FP : 25

FN : 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Scores for Multinomial Naive Bayes-Bag Of Words

Accuracy : 94.14225941422593

Precision : 92.81609195402298

Recall : 99.079754601227

F1 : 95.84569732937686

PS C:\MLHW1A> python bow\_NB.py ./enron1

TP : 284

TN : 145

FP : 23

FN : 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Scores for Multinomial Naive Bayes-Bag Of Words

Accuracy : 94.07894736842105

Precision : 92.50814332247556

Recall : 98.61111111111111

F1 : 95.46218487394957

PS C:\MLHW1A> python bow\_NB.py ./enron4

TP : 151

TN : 228

FP : 1

FN : 163

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Scores for Multinomial Naive Bayes-Bag Of Words

Accuracy : 69.79742173112339

Precision : 99.3421052631579

Recall : 48.089171974522294

F1 : 64.8068669527897

PS C:\MLHW1A> python ber\_NB.py ./hw1

TP : 303

TN : 128

FP : 45

FN : 2

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Metrics in Discrete Naive Bayes-Bernoullli

Accuracy : 90.1673640167364

Precision : 87.06896551724138

Recall : 99.34426229508196

F1 : 92.80245022970904

PS C:\MLHW1A> python ber\_NB.py ./enron1

TP : 278

TN : 149

FP : 29

FN : 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Metrics in Discrete Naive Bayes-Bernoullli

Accuracy : 93.64035087719299

Precision : 90.55374592833876

Recall : 100.0

F1 : 95.04273504273503

PS C:\MLHW1A> python ber\_NB.py ./enron4

TP : 143

TN : 379

FP : 9

FN : 12

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Metrics in Discrete Naive Bayes-Bernoullli

Accuracy : 96.13259668508287

Precision : 94.07894736842105

Recall : 92.25806451612904

F1 : 93.15960912052118

PS C:\MLHW1A> python bow\_lr.py ./hw1

The Final lambda is 0.5

Metrics in Logistic Regression-Bag Of Words

Accuracy : 92.88702928870293

Precision : 91.66666666666666

Recall : 98.4567901234568

F1 : 94.94047619047619

PS C:\MLHW1A> python bow\_lr.py ./enron1

The Final lambda is 0.1

Metrics in Logistic Regression-Bag Of Words

Accuracy : 91.22807017543859

Precision : 90.55374592833876

Recall : 96.19377162629758

F1 : 93.28859060402685

PS C:\MLHW1A> python bow\_lr.py ./enron4

The Final lambda is 0.3

Metrics in Logistic Regression-Bag Of Words

Accuracy : 95.39594843462247

Precision : 83.55263157894737

Recall : 100.0

F1 : 91.0394265232975

PS C:\MLHW1A> python ber\_lr.py ./hw1

The Final lambda is 0.5

Metrics in Logistic Regression-Bag Of Words

Accuracy : 94.76987447698745

Precision : 95.6896551724138

Recall : 97.08454810495627

F1 : 96.38205499276411

PS C:\MLHW1A> python ber\_lr.py ./enron1

The Final lambda is 0.9

Metrics in Logistic Regression-Bag Of Words

Accuracy : 95.39473684210526

Precision : 95.43973941368078

Recall : 97.66666666666667

F1 : 96.54036243822077

PS C:\MLHW1A> python ber\_lr.py ./enron4

The Final lambda is 0.9

Metrics in Logistic Regression-Bag Of Words

Accuracy : 96.50092081031308

Precision : 87.5

Recall : 100.0

F1 : 93.33333333333333