CS6375 - Project 1: Naive Bayes and Logistic Regression for Text Classification

Khoi P. N. Nguyen

This document reports the experiment of conducting spam classification using two class of models – Naive Bayes and Logistic Regression. In spam classification, given a piece of text representing the email, the model is asked to generate a “spam” or “ham” label for the email.

1 Experimental Setup

This section describes the experiment procedure. The source code can be found submitted together with this report.

1.1 Datasets

Three datasets were provided: `enron1`, `enron2`, and `enron4`. The statistics are shown in Figure 1.

To vectorize these datasets, two encoding methods were used: **Bernoulli** and **Bag of Words** (henceforth, BOW). While the former encodes whether a word appears in a text, the latter further encodes the number of occurrences of that word in the text. In both of these methods, all sentences are first tokenized using the recommended work tokenizer by the Natural Language Toolkit (NLTK). After that, a vocabulary is first constructed from the training set, where each token is assigned an index. The vocabulary size ranges from 10K to 17K across the datasets.

Size of	enron1	enron2	enron4
training set	450	463	535
test set	556	478	543
vocabulary	10111	10449	17890

Table 1: Size of the datasets

1.2 Models

Three classes of models were used: Naive Bayes (NB), Logistic Regression (LR), and Stochastic Gradient Descent Classifier (SGDClassifier). While the first two were implemented from scratch by the author, the last one is from the `sklearn` package [1].

For Naive Bayes to work with both types of coding, we implemented two variants: the **discrete** variant to work with Bernoulli encoding, and the **multinomial** variant to work with BOW encoding.

For Logistic Regression, L2 regularization was used. There are two hyperparameters: the regularization constant λ and the learning rate η. We explored learning rate $\lambda \in \{0.001, 0.01, 0.1\}$ and regularization constant $\eta \in \{0.001, 0.01, 0.1\}$. The best parameters were selected via a validation set, which is a random 30%-subset of the training set. Gradient Descent was used to learn the weights for a maximum of 100 iterations with early stopping implemented (min_delta=10^{-6} and patience=10).

For SGDClassifier, parameter search was done via the `GridSearchCV` utility of `sklearn`. Specifically, we search for loss $\in \{\text{hinge}, \log\}$, penalty $\in$ {l2, l1, elasticnet}, alpha (regularization constant) $\in \{0.001, 0.01, 0.1\}$, eta0 (initial learning rate) in $\{0.001, 0.01, 0.1\}$, and learning_rate (learning-rate adaptation strategy) in {constant, optimal, invscaling, adaptive}. Notice that we use the same set of values for regularization constant and learning rate as in Logistic Regression for valid comparisons.

2 Results

The results are shown in Figure 1.

3 Discussion

We now answer the instructor's questions.

Which data representation and algorithm combination yields the best performance (measured in terms of the accuracy, precision, recall and F1 score) and why? Among all models, SGDClassifier consistently yields the highest performance across all metrics and datasets. Among its variants, the one paired with Bernoulli encoding performs better. Looking closely into the loss function chosen by grid search, we see that "hinge" is always selected. "Hinge" stands for the linear SVM method, which is not covered in class yet. However, this result suggests that Linear SVM is a very robust and performant method to study.

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

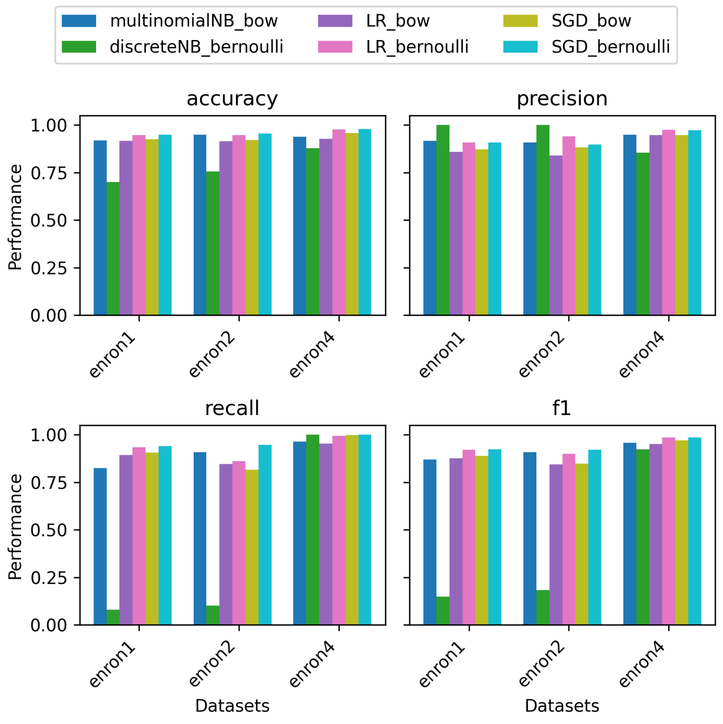

Figure 1: Performances of six models on the three datasets (enron1, enron2, and enron4) measured by four metrics (accuracy, precision, recall, and F1).

your yes/no answer. On Bag of Words, Multinomial NB performs on par with SGDClassifier and even outperforms LR. This shows that the conditional independence assumption is being held in this data representation.

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. Discrete NB performs much more poorly compared to LR and SGDClassifier. This suggests that the conditional independence assumption of NB does not hold when we only consider whether a word appears in the text.

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, my LR implementation performs *competitively* with SGDClassifier in most settings. In particular, it performs almost equally in accuracy and F1, sometimes outperforming in precision (enron2), lacking behind in recall. This shows that my implementation is unlikely to be buggy.

References

[1] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.