Training of a Multinomial Logistic Regression Classifier using scikit-learn

Leonardo Bonilla Escobar

Abstract

We train a logistic regression classifier for a multinomial topic classification task by using scikit-learn's implementation of a Stochastic Gradient Classifier and the 20 Newsgroups dataset. We conduct parameter hypertuning and conclude with the comparison of performance metrics for different parameter combinations.

1 Introduction

1.1 The 20 Newsgroups dataset

"The 20 Newsgroups Dataset is a collection of approximately 20,000 newsgroup documents", with each document consisting of (1) a news text extract, and (2) a label assigning it to one of 20 categories or 'newsgroups'. Version XX.YY of the dataset, used for this report, contains:

- Number of instances: 17683.
- Number of sentences in corpus: 207802.
- AvgSentences per instance: 11.75.
- Number of tokens in corpus: 4762051.
- AvgTokens per instance: 269.30.

The dataset contains both a training and a test split. For our purposes, we further divided the former into training and validation sets, containing respectively 80% and 20% of the training split's instances (see Table 1).

Instances are almost equally distributed across labels, with most labels in each set receiving around 5% (i.e. a twentieth) of the total instances, ±0.5%. The distribution of labels across set splits is also quite uniform. Exceptions are the labels 'alt.atheism' and 'talk.religion.misc', which are underrepresented in all three splits at ~4.3% and ~3.3%, respectively; and 'talk.politics.misc', which is underrepresented in both the training and the test splits, but not in the validation one. Something equally particular only to the validation split is the overrepresentation of the category 'sci.space', with over 6% of this split's instances receiving this label, against little over 5% in the other two splits.

Dataset Split	Instance Count
Training	8531
Validation	2133
Test	7019
Total	17683

Table 1: Instance counts in the dataset's splits

2 Training the classifier

2.1 Preparing the data

The validation and test sets were not used during the training stage. Only the training split was used to train the model. This allows us a meaningful use of the remaining partitions of the dataset during the testing stage, as the two other partitions remain "unseen" to the model during training.

The text in the training split instances was tokenized using the NLTK Punkt tokenizer, and then successively purged of stopwords, punctuations and "uncommon words " (defined here as "words happening less than N times across all instances of the corpus"). This reduced our number of unique tokens from a huge 138.436 to a very manageable 804 words with a minimum of occurrences N = 200, which constitutes our vocabulary.

We then calculated the document frequency *df* and inverse document frequency *idf* for each token in this vocabulary. Tokens with a high *df*/low *idf* include the letter 'n', the personal pronoun 'I', the letter 'T', and the pronoun 'us'. Uncommon tokens include the sequences '55.0', '2DI', 'GIZ', 'A86' and

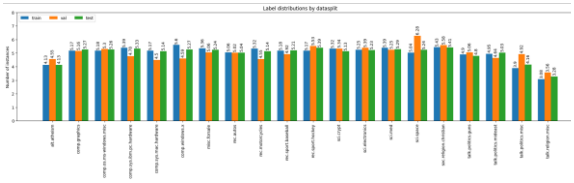

Figure 1: Label distributions across splits.

'Q,3', which made their way into the vocabulary even though they don't seem to be words at all.

Since the scikit-learn implementation of a logistic regression classifier expects vectors filled only with numerical values, we had to map our dataset instance labels to integers by assigning a different number to each label, and added a feature to each instance containing the *tf.idf* for each token in the instance's text, calculated for a term t and a document d using the formula shown below:

$$tf.idf(t,d) = tf(t,d) * idf(t) \qquad (1)$$

2.2 Using the SGC

We ran the scikit-learn Stochastic Gradient Classifier implementation on our prepared data with several parameter combinations.

- 804 token vocabulary, 15 epochs, and learning rate $LR = 0.0001$ (nicknamed 'Vanilla' for being our base configuration)
- 804 token vocabulary, 20 epochs, and learning rate $LR = 0.0001$
- 804 token vocabulary, 15 epochs, and learning rates $LR = 0.00001$ (nicknamed 'LR*0.1') and $LR = 0.001$ (nicknamed 'LR*10')
- 631 token vocabulary, 15 epochs, and learning rate $LR = 0.001$ (nicknamed 'RV' for "reduced vocabulary")

Other parameter combinations were also explored but are not listed since their results are not presented in this report.

3 Results

Learning curve graphs for the "Vanilla" configuration, the different learning rates, and the different vocabulary sizes are shown below (Figures 2,3). We would have loved to include a table showing per-label recall and precision for different parameter configurations, but space constraints and the power of Microsoft Word really made it impossible, so we have to make do with a puny table and a couple of macro figures (Table 2).

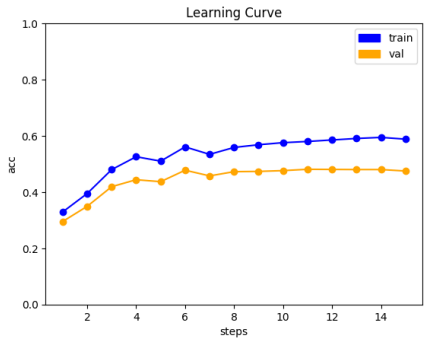

Figure 2: Learning curve for the "Vanilla" parameter configuration.

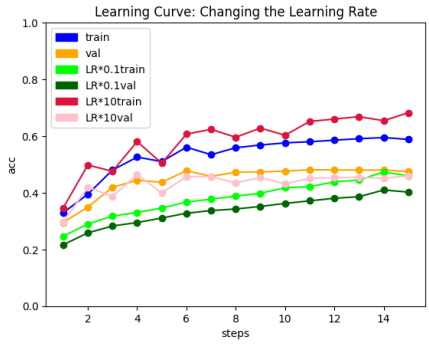

Figure 3: Learning curve for different Learning Rates.

Settings	Overall Accuracy	Precision	Recall	F1 Scores
"Vanilla"	0.442	0.455	0.431	0.442
RV	2133	0.429	0.396	0.406
LR*0.1	7019	0.424	0.414	0.422

Table 2: Accuracy, Precision, Recall, F1 for different configurations.

4 Conclusions

We really need to put in the work and learn how to use LaTeX.