COMP551 A2

Yusen Tang, Keyu Yao, Jaylene Zhang

November 2022

Abstract

The assignment task is to investigate the performance of two machine learning models, logistic regression and multi-class regression on the two different datasets. We find that the logistic and multi-class regression approach achieve better accuracy than KNN but is significantly slower to train.

Introduction

In this assignment, we implement two machine learning models, logistic regression on the IMDB reviews dataset and multi-class regression on the 20-news group dataset.

The IMDB dataset contains movie reviews along with their associated binary sentiment polarity labels. It is intended to serve as a benchmark for sentiment classification. The sentiment includes positive and negative sentiment and the overall distribution of labels is balanced (25k pos and 25k neg). In the labeled train/test sets, a negative review has a score ≤ 4 out of 10, and a positive review has a score ≥ 7 out of 10. Thus reviews with neutral ratings are not included in the training/test sets.

On the other hand, the 20-news group dataset consists of 18000 newsgroups posts on 20 topics. The data is divided into training and testing samples and the separation is based on messages posted before and after a specific date. We will choose 4 categories that are 'comp. graphics', 'rec.sport.hockey', 'sci.med', 'soc.religion.christian' for our experiments.

Citation

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

https://www.analyticsvidhya.com/blog/2017/06/a-comprehensive-guide-for-linear-ridge-and-lasso-regression/

https://datatofish.com/multiple-linear-regression-python/

Datasets

Preprocessing IMDB Data

To preprocess the IMDB dataset, we import from sklearn.datasets and load svmlight file using the function load svmlight file() to load all the features' counts as X, and sentiment benchmark as Y. We firstly transfer Y data into a binary numpy array, making positive review(score ≥ 7) as 1 and negative review(score ≤ 4) as 0. Since X has an array attribute called **indices** consisting of all the word indices in each review file, we then group indices by features and obtain the number of files that each word occurs in. We filter out the words appearing in less than 0.01 of the files and those more than 0.5, preventing us from choosing wrong features in later processing steps. Then we use the standardized X and calculate the z-score of each rows by using the formula:

$$z = \frac{X^T Y}{\sqrt{N}} \tag{1}$$

We choose a total of 500 features with the highest z-score since the greater the z-score the more likely we are to reject the null hypothesis. However, we will experiment with varying feature sizes in later steps.

Preprocessing 20 New Groups Data

To filter out irrelevant words in 20 new group data, we directly set the condition when applying CountVectorizer() to obtain occurrences for each word. We first intended to standardize the features using the same method as in the last dataset. However, in our later process, we found that the scale of data is still way too big to converge, causing the overflow problem when computing exponential. As a result, we adopted another way of standardization that ensures our data falls in the range [0, 10]. To choose the features that are the most relevant to our selected topics, for the four categories, we compute the Mutual Information between all the features and one-hot-encoded y value and obtain the union set over the top 100 features for each category. Finally, 308 features are selected for the 20 new groups dataset.

Selecting the Hyper-parameter

In order to select the optimum learning rate we further split the training set into training and validation samples while keeping the proportions of positive and negative reviews the same. The accuracy is shown in the table below:

learning rate	1	0.1	0.5	0.01	0.001
validation accuracy	0.7406400	0.7409600	0.7406400	0.7488000	0.7450400

From the table, we can see that our results do not fluctuate much with the decreasing learning rate, and we will continue with alpha=0.01 in our following experiments. We repeat the same procedure on the second dataset and find alpha=0.005, with the test accuracy hitting 0.80.

Results

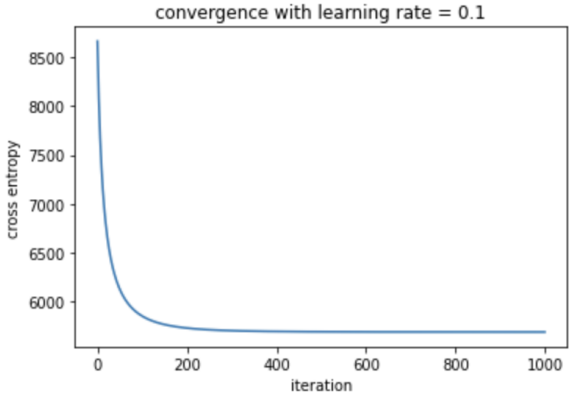
Logistics Regression

features number	train accuracy	test accuracy
100	0.6478	0.6344
300	0.7383	0.7189
500	0.7901	0.741
700	0.8191	0.7766
900	0.85	0.7943
1100	0.8701	0.8095

Using their absolute z-score associations, the requirement to choose the top features is

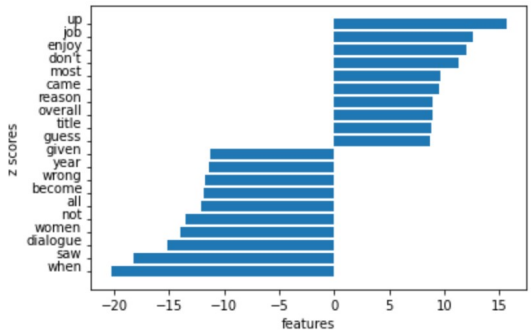
$$D \in [100, 1000]$$

by their absolute z-score associations. From 100-1000 features, we choose 500 features with the highest z-scores to determine the X train data set. The training accuracy is 0.7901 and the test accuracy is 0.7410. We also try smaller numbers and a bigger number of features. With the increase in a number of features, the test accuracy gradually increases. We check the gradient computed by ourTotions using a small perturbation, which is 1.7990106514914935e-09, within the range. Aftvalidationmented the logistics class and running the experiments, we can check gradients to see if the model is correct. Here we can see the gradient converges to 0 after 1000 iterations.

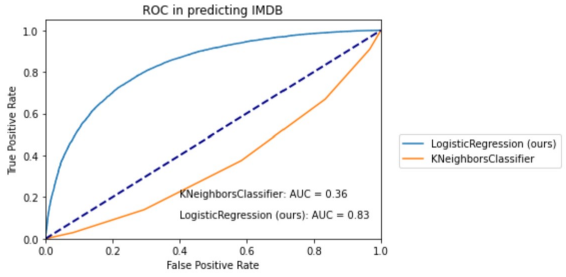


We draw a horizontal bar plot showing the top 20 features (10 most positive and 10 most negative) from the Simple linear regression on the IMDB data with the z-scores as the x-axis and

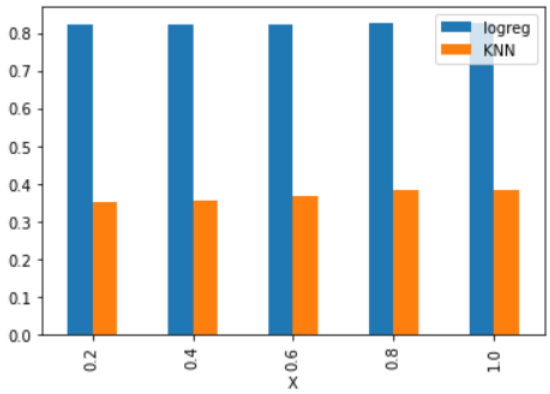
the feature names (i.e., words) as the y-axis. The 10 positive features are "up", "job", "enjoy", "don't", "most", "came", "reason", "overall", "title", and "guess". The 10 negative features are "when", "saw", "dialogue", "women", "not", "all", "become", "wrong", "year", and "given".



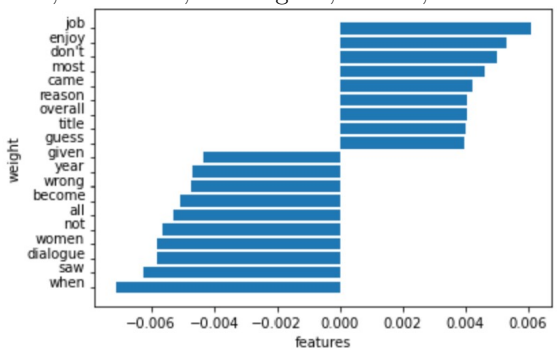
We draw a single plot containing two ROC curves of logistic regression and sklearn-KNN on the IMDB test data. The logistics regression remains higher all the time compare with the KNN curve



We draw a bar plot that shows the AUROC of logistic regression and KNN on the test data (y-axis) as a function with training, size equals to 0.2, 0.4, 0.6, 0.8, 1.0 We could see that the AUROC value of KNN remains low but gradually increases. The AUROC value of logistic regression is much higher, remaining at around 0.82.



We draw a horizontal bar plot showing the top 20 features (10 most positive and 10 most negative) from the logistic regression on the IMDB data with the coefficient as the x-axis and the feature names (i.e., words) as the y-axis. The 10 positive features are "job", "enjoy", "don't", "most", "came", "reason", "overall", "title", and "guess". The 10 negative features are "given", "year", "wrong", "become", "all", "not", "woman", "dialogue", "saw", and "when".

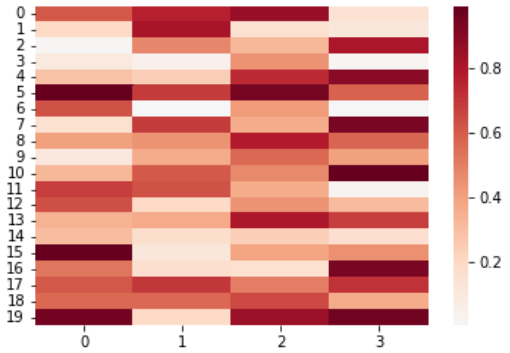


We could find that the top features are same with the top features chosen by using linear regression. They have different the most positive feature but "when" is the most negative for the two models

Multiclass Regression

The first time run the experiment, we find the accuracy is much lower than we expected. After comparing the result we get from our model and the one from the sklearn model, we learned that the cause of low accuracy is due to the nan of our gradi-

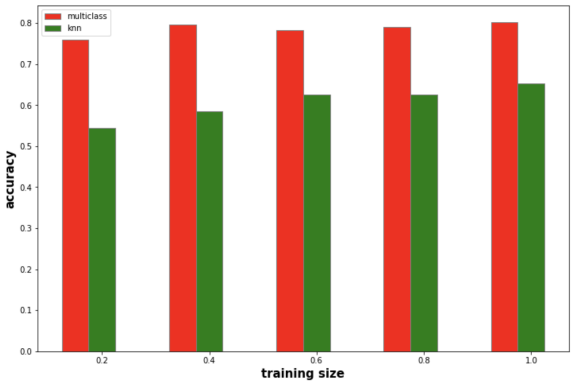
ent. We then notice that some of x still has a large value for certain features, which leads to the wrong calculation of our model. Based on this observation, we further calculated a smaller range. For the training data, we get an accuracy of 0.85 and 0.80 for testing data. Here is the heat map plot showing the top 5 most positive features as rows for each class as columns in the multi-class classification on 4 of the chosen classes from the 20-news group datasets:



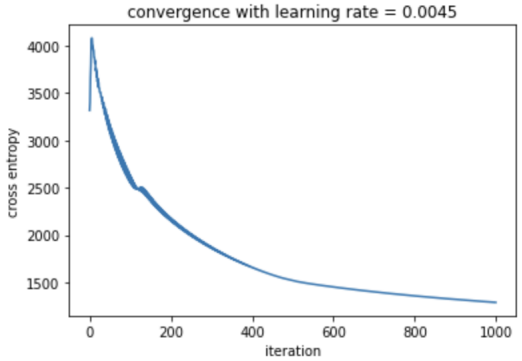
We further do experiments on different training sizes, assizes below:

training size	multiclass	knn
100%	0.8	0.65
80%	0.79	0.62
60%	0.78	0.62
40%	0.79	0.58
20%	0.75	0.54

We observe that comparing multiclass regression is not as easy as affected by the training size. Even if we train the model using a small training size, it still performs well. While KNN's result can be easily affected by the size of the training data.



To check if the coefficients are correctly computed, we run the model for 1000 times and record the cross entropy at each iteration. We first used the optimum learning rate 0.005 found in the validation step but the model tends to converge slowly and too many fluctuations were observed, so we reduce alpha to 0.0045 so that there is little impact on the accuracy.

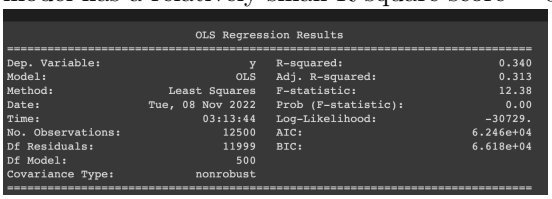


As seen from the above plot, at about the first 100 iterations the loss of entropy tends to fluctuate since our learning rate and overshoots the correct answer to a small extent. After 500 iterations the plot starts to converge to 0. We also calculate the error between the derived gradient and the true value and obtain a difference equal to 5.029e-10. Since the difference is less than 1e-10 we can thus conclude that our gradient calculation is correct.

Comparing with Other Models

Since we are curious about how the linear model performs on the first dataset, we apply the Multiple Linear Regression after restoring y values to the rat-

ings ranging from 1 to 9. It can be observed that the model has a relatively small R square score = 0.340.



OLS Regression Results

Dep. Variable:	y	R-squared:	0.340
Model:	OLS	Adj. R-squared:	0.313
Method:	Least Squares	F-statistic:	12.38
Date:	Tue, 08 Nov 2022	Prob (F-statistic):	0.00
Time:	03:13:44	Log-Likelihood:	-30729.
No. Observations:	12500	AIC:	6.246e+04
Df Residuals:	11999	BIC:	6.618e+04
Df Model:	500		
Covariance Type:	nonrobust		

However, the mean square error calculated for the model is rather large, suggesting that the linear fitting has a low variance but significant bias. We suspect that this is due to over-simplification of the linear model, therefore it should be made more complex to fit our datasets.

Moreover, we doubt that features with larger weight may have greater influence on the result, so we decide to try the Ridge Regression as well as LASSO. The MSE and R2 score obtained for the first data set are 0.426 and 0.272 respectively for Ridge, while the statistics become 0.432 and 0.253 while switching to LASSO. From the result we can see that the variance decreases by a quite amount and accuracy does improve.

Discussion and Conclusion

To sum up what we have done on the two samples, undoubtedly both logistics and multiclass regression work well on our samples, especially for the later. To be more specific, logistic regression performs better with increasing number of features while multiclass regression has a higher accuracy when using larger training sample. We can see that KNN is not the best model for both of our datasets, and we think that at the low accuracy of KNN is caused by the high dimension data (simply too many features compared to the previous experiments in A1). We also discover that the our models are quite robust to the varying learning rate. Therefore we can conclude that selecting optimum features is the dominating factor to success in both models. However, we should avoid using a large learning rate since it may overshoot the correct weight.

As we discussed in the previous section we investigate the performance of some linear regression models on the first dataset. We learn that linear models are not suitable for IMDB data but regularization does enhance our result. On contrary, since the 20 new groups are typical categorical data it is impossible to apply the linear regression to it. Therefore, for fututhe re research, we may want to try the decision tree on the second data and compare the accuracy with our multiclass logistic model.

Statement of Contributions

Yusen Tang: data preprocessing, logistic regression implementation.

Keyu Yao: data preprocessing, multi-class regression implementation.

Jaylene Zhang: data preprocessing, debugger