Homework 2, Statistical Learning, Fall 2019

Yizhen Dai, s2395479, jeremydai1992@gmail.com

Tasks

In this exercise, we are going to compare the **Naive Bayes (NB) model** to **Logistic Regression (LR)** on the *Car Evaluation*[1] dataset. The dataset was created by Marko Bohanec and included following categorical attributes (with corresponding available values in the brackets) shown in Table 1.

Table 1: Attributes and corresponding values in Car Evaluation Dataset

Variables	Values
• Car acceptability	[unacc, acc, good, vgood]

• Price
  • buying: buying price
  • maint: price of the maintenance
	
	[vhigh, high, med, low]
	[vhigh, high, med, low]

• Technical characteristics
  • doors: number of doors
  • persons: capacity in terms of persons to carry
  • lug_boot: the size of luggage boot
  • safety: estimated safety of the car
	
	[2, 3, 4, 5more]
	[2, 4, more]
	[small, med, big]
	[low, med, high]

The goal is to predict car acceptability based on a total of six characteristics. For the purpose of this exercise, the outcome variable (car acceptability) is recoded as 'Positive' and 'Negative', which correspond to [unacc] and [acc, good, vgood] separately.

Methodology

Models

In this exercise, we will compare Naive Bayes maximum likelihood estimation and Logistic regression conditional likelihood maximization. Both have been implemented in scikit learn (sklearn) package in Python, which will be used in this exercise.

NB algorithm, which is based on Bayes' theorem, predicts the probability of a sample belonging to a class given a feature vector. It assumes each feature is independent from each other, therefore the name of Naive Bayes. **LR method**, on the other hand, infers the odds in stead of probability.

In NB, Laplace smoothing is used to get rid of the probabilities of zero when a certain feature does not appear in a certain class. We will use parameter α to control the smoothing strength. To be specific, as mentioned in the homework introduction, 'for each class y and each feature X_j , the probability that X_j has value k given that the class is y is set to $(n_{k,y} + \alpha)/(n_y + \alpha K_j)$), where $n_{k,y}$ is

[1] Sources:/Users/jeremydai/GitHub/23-Statistical-Learning/data/car.names.txt

the number of data vectors where the feature has value k and the class is y, n_y is the number of data vectors with class y, and K_j is the number of values that feature X_j can take'. In this homework, we will test α = 1, 0.1 and 10 separately.

We are going to train and compare the following models:

- 'NB_1': Naive Bayes with Laplace smoothing parameter α = 1
- 'NB_0.1': Naive Bayes with Laplace smoothing parameter α = 0.1
- 'NB_10': Naive Bayes with Laplace smoothing parameter α = 10
- 'LR_full': Logistic Regression with all six variables, without regularization
- 'LR_2': Logistic Regression with only price variables('buying', 'maint') , without regularization
- 'LR_full_l2': Logistic Regression with all six variables, with L2 regularization (regularization factor C = 1.0)

Methods

To test the two models, the data are split in two equal parts randomly, with the first part as the training dataset and the second part as the test dataset. After splitting the dataset, both training and test datasets contain 864 observations separately.

To study how the models perform with different train data size, we will train models with n data points, with n = 1, 2, …, 864. We will ignore some cases with low n if the models cannot run with such limited dataset.

Evaluation Metric

In this homework, we evaluate the prediction accuracy using the 0/1-error: the average 0/1-loss values over the total dataset. The 0/1-loss function is shown below:

$$L(\hat{y}, y) = I(\hat{y} \neq y)$$

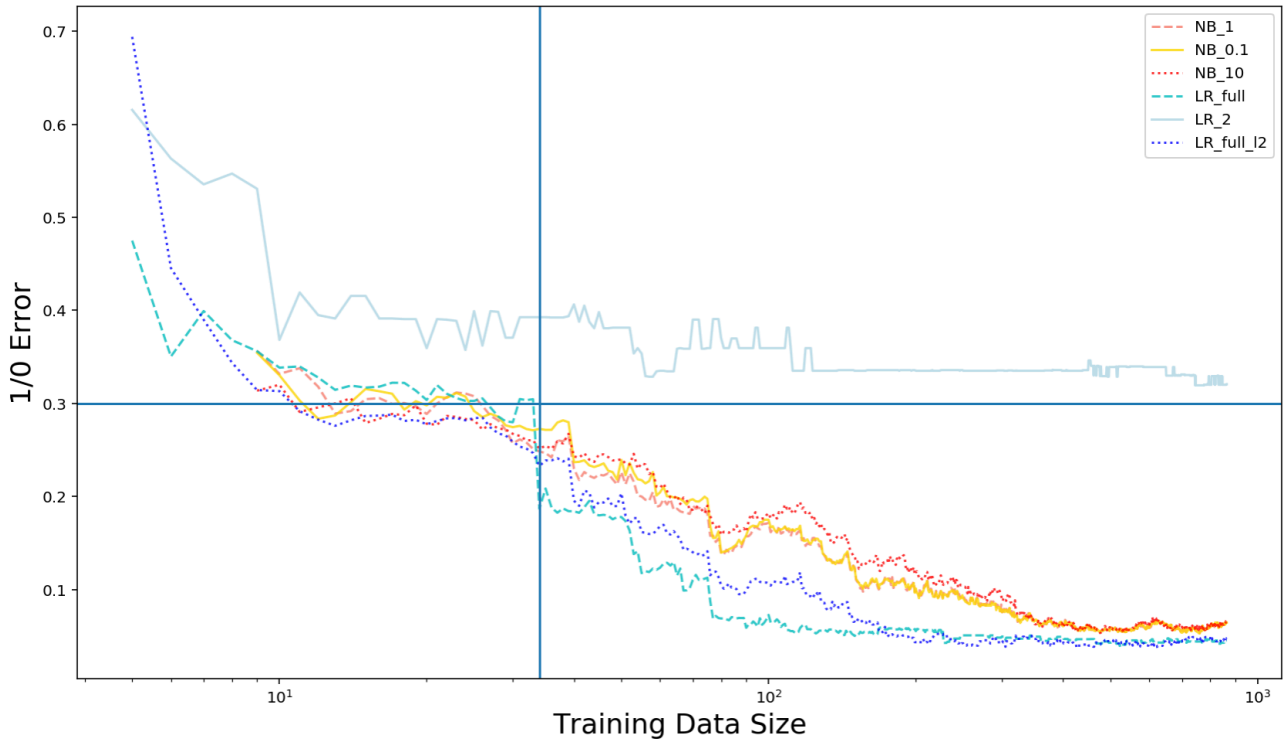
It equals to 1 when the predicted class is not the same as the true class, 0 otherwise. To smooth out some randomness in the 0/1-error, we will repeat the whole procedure 20 times and average the results. Each time, the total dataset will be randomly split into evenly into training and test sets with different random seed.

Results

The sklearn packages require at least 9 data points for NB and 5 for LR for our data set. This leads to NaN values when n is low.

Figure 1 shows the Average 1/0 Error for models ['NB_1', 'NB_0.1', 'NB_10', 'LR_full', 'LR_2', 'LR_full_l2'] over 20 runs. The lines with cold colors are for LR models while those with warm colors are for NB algorithms. The x-axis is on a log scale to help us observe the trends. The vertical line is x = 34, which is where LR_full begins to perform generally better than NB. The horizontal line is y = 0.3, which is baseline for 0-1 error. There are 1728 observations in total with no missing values. 1210 of them have negative car acceptability, which accounts for 70.023 % of total dataset. We can obtain an 0/1 error of around 0.3 by keep betting on 'negative' outcomes.

Figure 1 Average 0/10-error for six models over 20 runs



Discussion

Smoothing parameter α for NB

A bigger α helps improve prediction accuracy for NB when the training data size n is small (n <34). It is because a stronger smoother can help reduce the variance when the data is limited. When n gets bigger, a bigger α cannot help but decrease the prediction accuracy for NB.

Model performance vs Training Data Size

There is no method that performs the best in all situations. When the training dataset size n is limited (< 34), NB performs better than LR without regularization. When n gets bigger, LR has better prediction accuracy.

This result is consistent with the paper by Ng and Jordan[2] and the theoretical insights we have learned about the difference between generative and discriminative models. Given the feature X and label Y, NB and LR are generative-discriminative pair since NB estimate the joint distribution P(X,y) and LR the posterior probability of P(y|X) in same model P.

NB, as a generative model, can reduce variance but introduce bias in theory. It theoretically performs better for small n, because the prior probability in the objective function acts as a regularizer. We can see this trend in our homework, that the information on prior probabilities help in improving the prediction when the n is small.

---

2 On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes

LR, as a discriminative model, has less bias but more variance. It is theoretically a better fit when N is large, because we care more about bias than variance when we have big data set. This trend is also consistent with Figure 1.

In summary, NB reaches its asymptotic performance more quickly (O(log p)) than LR does (O(p)). However, LR performs better when it reaches its asymptotic performance.

Multicollinearity

Since NB assumes the independence between each feature, NB might perform worse if there are highly correlated features. Surprisingly, the features in our dataset are very correlated based on the training dataset in the last run. After recoding '5more' and 'more' into 5 and coding other ordinal features with 1,2,3,4, the correlation between different features based on the training dataset in the last run is shown in Table 2.

Table 2: The correlation between different features based on the training dataset in the last run

	buying	door	lug_boot	maint	per	safety
buying	1.000000	0.017391	0.053936	-0.002390	0.022161	0.013661
door	0.017391	1.000000	-0.000012	0.008427	-0.040273	0.004465
lug_boot	0.053936	-0.000012	1.000000	0.007664	0.008589	0.029644
maint	-0.002390	0.008427	0.007664	1.000000	0.015690	-0.029156
per	0.022161	-0.040273	0.008589	0.015690	1.000000	0.020402
safety	0.013661	0.004465	0.029644	-0.029156	0.020402	1.000000

As shown in Table 2, the absolute correlations are all smaller than 0.06. Failure to meet the independence assumption is not the reason of NB performing worse than LR with big n for our dataset.

Regularization

As stated in the paper, 'logistic regression can often be improved upon in practice by regularization techniques '. Based on Figure 1, the introduction of a L2 regularization term help reduce overfitting and improve the prediction accuracy when training data size n is relatively small (7< n < 34) with regard with the number of features. With large n (>34), the L2 regularization term cannot help LR much. This is consistent with the observation of LR_full model from last run (random seed = 19 during train/test split). The 0/1 error is 0.052 and 0.0428 for train and test datasets separately, which would not be expected if there was overfitting.

Generally undesirable model: LR_2

Except for the case that there are only 5 training data points, LR with only 2 features performs worst. It is even worse than random guesses. Since around 70% of labels in the whole data set are 'negative', we can get a 0/1 error of about 0.3 if we keep betting on 'negative' outcomes.

Also, LR_2 hardly gets any improvement when n is bigger than 100. With only two category features (the maintenance and buying prices), each having four possible values, we have only 16 potential input combination. Without further information (more features), getting more data cannot help with the prediction accuracy. The model is forced to be stable and stably bad at predicting.

Extreme estimated probabilities

To measure how extreme (close to 0 or 1) are the estimated probabilities P(Y|X) for NB and LR on the training data, we calculate the square of the discrepancy in estimated probabilities and sum them up. The equation is shown below:

$$\sum_{i=1}^{n} (P(y_1 \mid X_i) - P(y_2 \mid X_i))^2$$

The results are 522.704 and 751.165 for NB_1 and LR_full respectively on the train data set from last run of the 20 runs. The more extreme estimated probabilities there are, the higher sum of the square it is.

We found that LR estimates more extreme probabilities, which means LR fits better on the training data. This is consistent with the 0/1 errors on the training data, which are 0.059 and 0.052 for NB_1 and LR_full respectively. This may show that LR can concentrates more on data points that are close to the decision boundary.

Dealing with Imbalanced Data

0/1 errors accuracy is not the best metric to evaluate our imbalanced dataset. We plot the confusion matrix (shown in Figure 2) based on the test data results of NB_1 and LR_full from the last run of the 20 runs. We can see that LR performs better than NB for ‘postive’ recall. This is consistent with our intuition. Since NB makes prediction based on the prior probabilities, the difference in class ratios between training and test data sets may lead to a less accurate estimation of NB model.

Figure 2: Confusion matrix on test dataset from last run for NB_2 (left) and LR_full (right)