Report Lab1 for the Machine Learning course

Nicholas Attolino

October 2023

Contents

Introduction

This work is a report for the first laboratory of the Machine Learning course.
The subject of this laboratory is a **Naive Bayes Classifier** using MATLAB, a versatile platform that serves as a numerical computation and statistical analysis environment, as well as a programming language[3].
The chapters are divided in:

1. Goal;
2. Methods used;
3. Results.

1 Goal

The goals of this project are:

- Data pre-processing;
- Build a Naive Bayes Classifier;
- Improve the classifier with Laplace (additive) smoothing

Bayesian classifiers assign the most likely class to a given example described by its feature vector.
Simplifying the learning process of these classifiers becomes easier by assuming that the features are unrelated to each other, then:

$$P(X|C) = \prod_{i=1}^{n} P(X_i|C) \tag{1}$$

where $X = (X_1, .., X_n)$ is a feature vector and C is a class.
Despite this unrealistic assumption, the resulting Naive Bayes classifier proves to be highly effective in real-world scenarios, frequently rivaling more complex methods[1].

1.1 Data Files overview

```
- Attolino-Lab1.zip
    - Attolino-Lab1
        - assignment1_ML.m
        - weather_data.txt
```

The *assignment1_ML* is the main program while *weather_data* is the dataset used.

2 Methods used

2.1 Dataset elaboration

The classifier will undergo testing using a Weather dataset.
Similar to any dataset, this one requires preprocessing before it can be utilized by automated programs.
Specifically, this dataset contains categorical features that must be transformed into numeric formats immediately upon loading the raw data into the MATLAB workspace.
Following these conversions, the entire data structure is altered, as the function employed to load the data produces a cell matrix that is challenging to handle. This approach proves significantly faster compared to manual conversion, particularly with datasets containing hundreds of thousands of observations.
From this dataset, both the training and test sets are randomly extracted.
This task is accomplished using a MATLAB function that generates random numbers serving as indices.

2.2 Classification

Before the program starts the classification, it should check the given data are eligible to said classification.
First of all the test set's number of features should match the training set's one. The number of columns could not match only when the training set has one more. This happens when the test set doesn't come with the target.
It's also important the single units of data are consistent. Since the data set is preprocessed converting every possible level to a specific integer, no one of them should be lesser than one. The program needs to know how many different classes it's going to classify the observations into.
To do that, it simply count the number of unique values in the target column of the training set. In order to compute the likelihood for each possible level of each attribute for every class, the software follows this algorithm:

Algorithm 1 Likelihood P(x=v|c) estimation

```
1: for each class C do
2:     compute the frequency of the class as number of occurrences
3:     - divided by the number of observations
4:     for each variable x do
5:         for each each possible value v for variable x do
6:             the number of instances of class c
7:             - that have variable x == value v
8:             - divided by the number of instances of class c
9: return likelihood
```

Practically, the returned likelihood matrix is a cell matrix with as many rows as the number of classes and as many columns as the number of attribute.
Each value of the matrix is an array that stores the frequency of each possible level of the specific attribute in the specific class. With this data, it's possible to compute the overall probability that a given observation belong to a class rather than another.

Algorithm 2 Discriminant function

```
1: for each observation $X$ do
2:     for each class $C$ do
3:         set the discriminant function of $[X, C]$ equal to the frequency of $C$
4:         for each variable $d$ do
5:             retrieve the frequency of the possible levels of $d$
6:             - when it belongs to $C$
7:             update the discriminant function of $[X, C]$ multiplying it for
8:             - the frequency of the level of $d$ in $X$, when the class is $C$
9: return discriminant function
```

Now it's possible to compare the values of the discriminant function for each observation. The probable belonging class is going to be the one with a greater value.
Once the program computes and stores the estimated target in a matrix, it's finally possible to elaborate the error rate. This action is obviously executed only if the test set target is given. It's equal to the number of incorrect classification divided by the number of observations.

2.3 Laplace Smoothing

Occasionally, especially when dealing with a small dataset like the one under consideration, certain combinations exclusively appear in the test set. In such instances, the program might attempt to calculate the frequency of a value that has never been encountered before.
Consequently, a specific statement might assign its probability as zero. While this approach prevents errors in the code, it is inaccurate. Absence of a particular value in the training set doesn't necessarily imply its probability is zero.
To rectify this behavior, the classifier function allows the option, when called, to employ the Laplace (Additive) Smoothing algorithm. This method presents an alternative approach for computing the probability of observing a specific value of a random variable x.
Knowing there have been N experiments and that the value i occurs n_i times, then:

$$P(x = i) = \frac{n_i}{N} \tag{2}$$

With Laplace Smoothing, the probability of observing value i is given by:

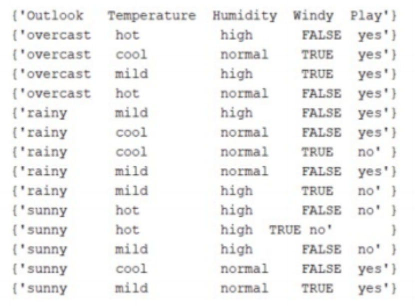
$$P(x = i) = \frac{n_i + a}{N + av} \tag{3}$$

where v is the number of values of the attribute x and a is a parameter that express how the data needs to be trusted from the program.
With a value of a greater than zero, it's possible to avoid the problem of zero probability. In order to implement this algorithm, a few changes to the code are necessary:

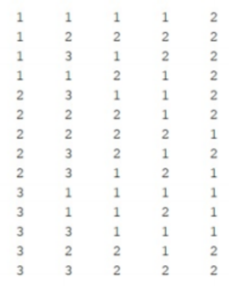
- First of all, the program needs to know the number of levels of each attribute. This information shall be given as an additional row of the training set;
- The program must remove the row with the number of levels before it starts to work on the training set. Also, if the program doesn't know at prior the number of levels for each attribute, it shall count the number of unique level from the training set;
- Ultimately, if the Laplace smoothing function is active, the updated way of computing the probability of observing a specific value needs to be used.

3 Results

As already mentioned, the data set used for valuating the classifier is the Weather data set. It has 14 instances, 4 attributes and 2 classes. It represents the decision problem of whether to go play outdoor or not, given a description of the weather.



```
{'Outlook   Temperature  Humidity  Windy  Play'}
{'overcast   hot          high      FALSE  yes'}
{'overcast   cool         normal    TRUE   yes'}
{'overcast   mild         high      TRUE   yes'}
{'overcast   hot          normal    FALSE  yes'}
{'rainy      mild         high      FALSE  yes'}
{'rainy      cool         normal    FALSE  yes'}
{'rainy      cool         normal    TRUE   no' }
{'rainy      mild         normal    FALSE  yes'}
{'rainy      mild         high      TRUE   no' }
{'sunny      hot          high      FALSE  no' }
{'sunny      hot          high  TRUE no'       }
{'sunny      mild         high      FALSE  no' }
{'sunny      cool         normal    FALSE  yes'}
{'sunny      mild         normal    TRUE   yes'}
```



```
1    1    1    1    2
1    2    2    2    2
1    3    1    2    2
1    1    2    1    2
2    3    1    1    2
2    2    2    1    2
2    2    2    2    1
2    3    2    1    2
2    3    1    2    1
3    1    1    1    1
3    1    1    2    1
3    3    1    1    1
3    2    2    1    2
3    3    2    2    2
```

Figure 1: Weather dataset, before and after the pre-process phase

3.1 Standard Naive Bayes Classifier

The first test involves the standard configuration of the classifier (i.e. the one without the Laplace Smoothing function). There have been 100 calls to the classifier with different training and test set each time. The error rate ranges from 0 to 1 with an average of 0.44.

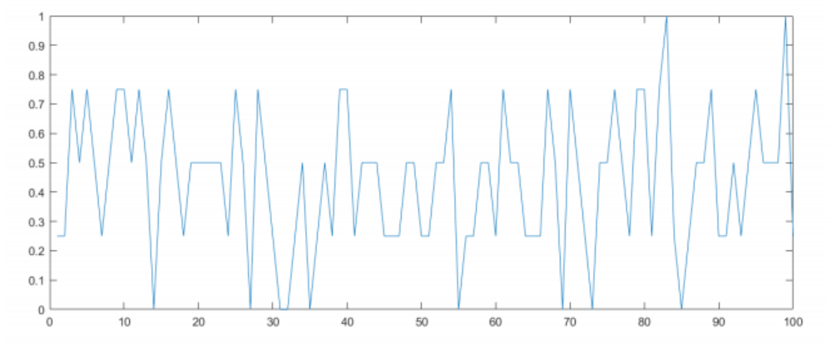


Figure 2: Error rate during the 100 episodes

3.2 Naive Bayes Classifier with Laplace Additive Smoothing

In order to see how the error rate changes with the variation of the a parameter, there have been 100 calls for each a in the range [0:0.1:2]. The minimum (best) average error rate was obtained for $a = 2$.
An a greater than 1 means that a prior belief of equally probable values is more trusted than the probability extracted from the data. The result then has its sense: the computed likelihood is based on a very poor training set, just 10 observations.

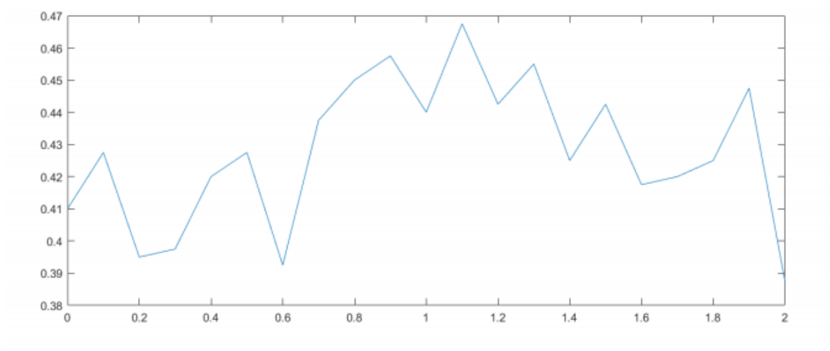


Figure 3: Error rate average with $a = [0 : 0.1 : 2]$

4 Conclusions

Assuming independence among features, the Naive Bayes classifier represents an optimal model, wherein no other classifier is anticipated to attain a lower misclassification error rate[2].
However, when this assumption is not true and practically in the vast majority of real-world classification problems, the naive Bayes classifier has demonstrated excellent performance.

References

[1] I. Rish, *An Empirical Study of the Naïve Bayes Classifier*. Jan. 2001, vol. 3.

[2] D. Berrar, *Bayes' Theorem and Naive Bayes Classifier*. Jan. 2018, ISBN: 9780128096338. DOI: `10.1016/B978-0-12-809633-8.20473-1`.

[3] T. M. Inc., *MATLAB Version: 23.2.0.2365128 (R2023b)*. The MathWorks Inc., 2023.