Naive Bayes Classifier
Assignment 1 of the Machine Learning 1 course 2022/2023

Nicchiarelli Gabriele

DIBRIS - Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi
Università Degli Studi di Genova

1 Introduction

In this article, we will discuss the classification problem, in particular with the **Naive Bayes Classifier**.
In *classification* we have an object, called *classifier*, that associates each input with an output (also called class). In order to classify correctly the inputs, the classifier needs to study a set of inputs along with their respective classes. With the *training* process, the classifier acquires the knowledge to assign the new input to the corresponding classes.
There are many different types of classifiers, but the simplest, still efficient, is the so-called Naive Bayes Classifier, which is a statistical classification technique based on the Bayes Theorem.

2 Theory of Naive Bayes Classifier

A general classification problem states that if we have an observation (*x*) as input, *y()* will produce an output *t* called *class*: $t = y(x)$.
In order to predict the correct class, we have to choose a classifier with minimum error probability as it is for the Naive Bayes Classifier.
This classifier is built using "wrong" discriminant functions based on a "naive" assumption:

$$P(\mathbf{x}|t_i) = P(x_1|t_i)P(x_2|t_i)...P(x_d|t_i) \tag{1}$$

This states that we are pretending that the input variables are all independent of each other.
To predict the classes we use a discriminant function $g_i(x)$ defined as follows:

$$\begin{aligned} g_i(x) &= P(t_i)[P(x_1|t_i)P(x_2|t_i)...P(x_d|t_i)] = \\ &= P(t_i)\prod_{j=1}^{d} P(x_j|t_d) \end{aligned} \tag{2}$$

After computing all the discriminant functions, the classifier takes the one with the highest value and classifies *x* with *t*.
Now we can start the training session, where the classifier will acquire data from the training set and computes every conditional probability, using this formula:

$$P(x_j = v_k|t_i) = \frac{\text{number of } x_j = v_k \text{ in class } t_i}{\text{number of observations of class } t_i} \tag{3}$$

where v_k is the possible value for the attribute *k*.

2.1 Improvements with Laplace Smoothing

If we use a small data set we may encounter a problem, some combinations that appear in the test set may not be encountered in the training set, so these probabilities will be zero. This causes a big problem since just one zero will cause the overall results to be zero.
The *Laplace Smoothing* takes into account those attributes that could not appear in the training set but are present in the overall data set, it introduces a *trust factor* (α) in the conditioned probability computation such that is will never be equal to zero, even if some values do not appear in the training set.

$$P(x_j = v_k|t_i) = \frac{\text{(number of } x_j = v_k \text{ in class } t_i) + \alpha}{\text{(number of observations of class } t_i) + \alpha \text{v}} \tag{4}$$

3 Assignment

I decided to use *Python* programming language for the assignment since it provides a lot of useful tools for machine learning. Such as the *Pandas* library for data pre-processing, which provides functionalities for converting *csv* files in *dataframes*. A dataframe is an object used to store data sets, it contains methods and attributes suitable for working with data sets.
The assignment is divided into 3 steps:

1. Data pre-processing
2. Build the Naive Bayes Classifier
3. Improve the classifier with Laplace smoothing

For this assignment we choose to use a weather data set (fig. 1) composed of:

- 14 observations;
- 4 attributes (*outlook*, *temperature*, *humidity*, *windy*);
- 2 classes (YES and NO for the *play* attribute).

The first four columns are chosen as attributes, while the last column is chosen as the target column.

```
#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
```

Figure 1. Wheather dataset

3.1 Data pre-processing

In machine learning datasets are usually provided in *csv* format. All values of the data set are categorical, so they need to be converted into numerical values, afterward, I split the data set into a test set and a training set, respectively 30% and 70% of the data set. I also separated the target column from the dataset. To be sure that the split was successful I performed a check that the test and train matrices have the correct dimensions.

3.2 Build the Naive Bayes Classifier

The naive Bayes classifier is implemented as a class with two methods:

- a *fit* method that takes care of the training session;
- a *predict* method that makes the classification of the predicted classes;
- the *error* method that computes the error rate of the prediction.

In particular, the fit method calculates the prior and the conditional probabilities for each feature, which will then be used by the predict method to compute the posterior probabilities.
Finally the predict method returns the predicted classes that will then be printed on the console.
The error method takes care of computing the error rate, which is computed as the number of wrong predictions divided by the number of total observations.

3.3 Improve the classifier with Laplace smoothing

We take an α factor of 1 defined as a global constant for the Laplace smoothing. Then the classifier is pretty much the same as before, we just have to add the α factor to the conditional probabilities calculation.

4 Results

When we run the program, it will print in the console three parameters: the actual values of the classes, the result of our prediction, and the error rate. An example of a possible output is shown in the following figures:

```
True output:  ['yes', 'yes', 'no', 'yes']
Predicted output:  ['yes', 'no', 'no', 'no']
Error: 50%
```

Figure 2. Prediction results (50% error rate)

```
True output:  ['yes', 'yes', 'no', 'yes']
Predicted output:  ['yes', 'yes', 'yes', 'yes']
Error: 25%
```

Figure 3. Prediction results (25% error rate)