THE NAIVE BAYES CLASSIFIER

November 24, 2019

Alberto Grillo ID: 4229492
Alberto Ghiotto ID: 4225586

Abstract— **Nowadays, the Naive Bayes Classifier has a widespread set of application: from email spam detection, to sentiment analysis, digit recognition, medical diagnosis and much more. Its algorithm works on the principles of conditional probability as given by the Bayes' theorem, which gives the probability that some hypothesis is true given an event that was tested positive for such hypothesis.**

The aim of this laboratory was to implement a Naive Bayes classifier in order to get a solid grasp of the math behind it.

I. INTRODUCTION

In this assignment some guidelines on how to develop the project were given, along with two different datasets on which to perform the testing phase. The first step was to manipulate the given datasets in order to make them readable by the MATLAB pre-processing script that was gonna be used to and feed them to the Naive Bayes classifier. In practice, this task consisted in enumerating each possible value of each attribute with increasing integers starting from 1: this assumption is useful either for using them as matrix indexes or for computing the number of possible values for each attribute.
The next step was to actually build the Naive Bayes classifier as a MATLAB function. The guidelines required that the program would take as mandatory parameters the training and test set, along with the class label vector regarding the test set as an optional parameter. In particular, the classifier function would have to check the consistency of the data contained in the dataset by verifying that the matrix dimensions adds up and that no entry of the matrixes would be lower than 1. Then it would have to train the classifier on the training set and return the inferred classification of the test set, along with the error rate in the case that the class label parameter was provided.
The third and last step consisted in improving the classifier in order to deal with the case of test sets including attribute values that were not in the training set. This was done by adding Laplace smoothing which means adding some terms that account for a prior belief asserting, since nothing is known about those missing values,that such values are equally probable.

II. BACKGROUND

Bayes' theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event.
Its formula:

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

gives the probability of a hypothesis after seeing an experimental observation.
A classifier is a rule y() that receives an observation x and outputs a class w = y(x).
A Naive Bayes classifier is a probabilistic machine learning model that's used for classification tasks based on applying Bayes' theorem with strong independence assumptions between the features.
The term Naive refers to the assumption that the features that go into the model are independent from one another. Meaning that changing the value of one feature, does not directly influence or change the value of any of the other features used in the algorithm. Moreover, features are considered to provide an equal contribution to the outcome, meaning that none of the attributes is irrelevant and assumed to be contributing equally to the outcome.
This type of classifier is particularly handy when it comes to evaluate features that are binary. Its learning process consists in counting how often each value of an observation occurs

in a given class of the training set. With the increase in the number of features the naive assumption can be considered approximately correct.

III. IMPLEMENTATION

As mentioned in the introduction section, the actual implementation started from the analysis and the manipulation of the dataset, which was done in a pre-processing script.
The data file is divided into two parts: the feature matrix, which consists of all the dataset minus the last column, and the class labels which consist of the column left over from the previous division.
The feature matrix consists in the row vectors containing the value of the features representing the condition related to the event, which is in turn contained in the class label.
Once having adjusted the dataset in the dedicated script, the implementation of the classifier function would start by simply checking the dimension and content of the data matrixes, as mentioned in the introduction section.
After that, the actual training would take place by counting the occurrences of each different classes and for each different levels of each attributes, in order to being able to compute the likelihood (conditional probability) of each attribute and the a piori probability of the single classes.
Then, thanks to the previously computed variables, it would be possible to compute the a posteriori probability by applying Bayes' formula.

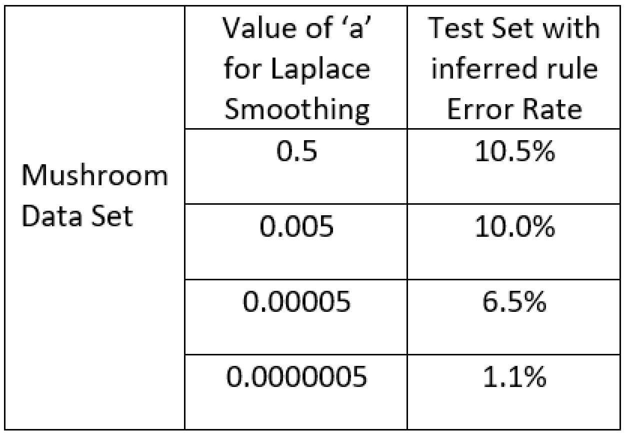
$$P(A \mid B) = P(B \mid A)\, P(A)$$

In this particular case the denominator is removed, since, for all entries in the dataset, it does not change and therefore it can be omitted. After having computed the a posteriori probability one could even decide to normalize the resultant values in an interval from 0 to 1 but that is optional and not mandatory for the purpose of this project. Eventually, by analyzing the a posteriori probability it would be easy to compute a vector of prediction from which to, if the ground truth vector was provided, extrapolate the error rate by simply compare it with the class label vector and counting the differences between them. Done that, the classifier function would return the results to the pre-processing script which would print the outcome of the classification.

As mentioned in the introduction section, the third step required the implementation of Laplace smoothing in order to deal with the case of test sets including attribute values that were not in the training set and to add the information about the number of levels for each feature. The former is done by adding some terms that account for your prior belief, while the latter is performed by adding a row vector to the dataset containing the information about the number of levels of each attribute during the pre-processing data step.

IV. RESULTS

In order to test the classifier, as mentioned in the introduction, two datasets were given. The first one is a small dataset designed is such a way that it was possible to manually verify the results but since the dataset entries weren't enough, they weren't trustworthy. The second one is a much larger dataset which gives more reliable results but needs Laplace smoothing to work properly.
It's important to notice that applying a Laplace smoothing makes the error rate slightly increase since the probability of absent values are assumed in order to make the computation possible. However, this process warps the real probabilities in the case when they are computable. This issue is minimized with a weight parameter of less than one.
In the table below the average error rates produced by different values of the weight parameter 'a' are shown.



	Value of 'a' for Laplace Smoothing	Test Set with inferred rule Error Rate
Mushroom Data Set	0.5	10.5%
	0.005	10.0%
	0.00005	6.5%
	0.0000005	1.1%

Fig. 1. Results with randomized training and test set and Laplace smoothing

V. CONCLUSIONS

The naive Bayes classifier is a good starting point for learning the basics of Machine Learning. Not only it's easy to implement, but also the naivety assumption is often approximately correct. Nevertheless, it still has some flaws that can be improved with simple adjustments.
First of all, as it has been done in the third task, it is possible to overcome the eventuality of missing attribute values with data processing and Laplace smoothing.
Another weakness of the classifier implemented in this assignment is the pre-processing phase required in order to make the classifier works. It's possible to relax the integer conversion assumption exploiting some MATLAB functions extracting the information needed directly from the dataset.
Eventually, it should be considered the implementation of an algorithm to cope with the unavailability of some attribute values. This algorithm should decide whenever discard observations or input variables or, if possible, impute the missing values.