Naive Bayes classifier

Student Gianluca Galvagni, S5521188, Università degli Studi di Genova

***Abstract*—In the first month of lessons, we talked about probability. The probability is how many times something can or can not happen and it is a branch of mathematics. There are many kinds of probabilities but in our assignment we have studied the Naive Bayes classification. Our goal for this assignment was created a code which can "learn" by a data set and then give us "answers" in according with what the code had learned before. This process is a beginning of artificial intelligence.**

***Index Terms*—Probability, Bayesian probability, Matlab.**

I. Introduction

THE word probability has been used in a variety of ways since it was first applied to the mathematical study of games of chance. Does probability measure the real, physical, tendency of something to occur, or is it a measure of how strongly one believes it will occur, or does it draw on both these elements? In answering such questions, mathematicians interpret the probability values of probability theory.
However, when it comes to practical application, there are two major competing categories of probability interpretations, whose adherents hold different views about the fundamental nature of probability.

A. Frequentist probability

It is an interpretation of probability of an event's probability as a generalization of frequency of occurrence of that event in infinite repetitions of an experiment or many trials.

B. Subjective probability

It is a type of probability derived from an individual's personal judgment or own experience about whether a specific outcome is likely to occur. The most popular version of subjective probability is Bayesian probability.

II. Bayesian probability

The naive Bayes classifier greatly simplify learning by assuming that features are independent given class. Although independence is generally a poor assumption, in practice naive Bayes often competes well with more sophisticated classifiers. Let $X = x_1, x_2, ..., x_n$ be a sample, whose components represent values made on a set of n attributes. In Bayesian terms, X is considered "evidence". Let H be some hypothesis, such as that the data X belongs to a specific class C. For classification problems, our goal is to determine $P(H|X)$, the probability that the hypothesis H holds given the "evidence", (i.e. the observed data sample X). $P(H|X)$ is the a posteriori probability of H conditioned on X. In contrast, P(H) is the a priori probability of H, which is independent of X. Similarly with this, there is P(X), it is the a priori probability of X.
According to Bayes' theorem, the probability that we want to compute $P(H|X)$ can be expressed in terms of probabilities P(H), $P(X|H)$, and P(X) as

$$P(H|X) = \frac{P(X|H) \cdot P(H)}{P(X)}$$

and these probabilities may be estimated from the given data.

A. Naive Bayesian Classifier

The naive Bayesian classifier works as follows:
1) Let T be a training set of samples, each with their class labels. There are t classes, $T_1, T_2, ..., T_k$. Each sample is represented by an n-dimensional vector, $X = x_1, x_2, ..., x_n$, depicting n measured values of the n attributes, $A_1, A_2, ..., A_n$, respectively.
2) Given a sample X, the classifier will predict that X belongs to the class having the highest a posteriori probability, conditioned on X. That is X is predicted to belong to the class T_i if and only if

$$P(T_i|X) > P(T_j|X) \quad for 1 \leq j \leq m, j \neq i.$$

Thus we find the class that maximizes $P(T_i|X)$. The class T_i for which $P(T_i|X)$ is maximized is called the maximum posteriori hypothesis. By Bayes' theorem

$$P(T_i|X) = \frac{P(X|T_i) \cdot P(T_i)}{P(X)}.$$

3) As P(X) is the same for all classes, only $P(X|T_i) \cdot P(T_i)$ need be maximized. If the class a priori probabilities,$P(T_i)$, are not known, then it is commonly assumed that the classes are equally likely, that is, $P(T_1) = P(T_2) = ... = P(T_k)$, and we would therefore maximize $P(X|T_i)$. Otherwise we maximize $P(X|T_i) \cdot P(T_i)$.
4) Given data sets with many attributes, it would be computationally expensive to compute $P(X|T_i)$. In order to reduce computation in evaluating $P(X|T_i) \cdot P(T_i)$, the naive assumption of class conditional independence is made. This presumes that the values of the attributes are conditionally independent of one another, given the class label of the sample. Mathematically this means that

$$P(X|T_i) \approx \prod_{k=1}^{n} P(X_k|T_i).$$

The probabilities $P(x_1|T_i), P(x_2|T_i), ..., P(x_n|T_i)$ can easily be estimated from the training set. Recall that here x_k refers to the value of attribute A_k for sample X.

- If A_k is categorical, then $P(x_k|T_i)$ is the number of samples of class T_i in N having the value x_k for attribute

A_k, divided by $freq(T_i, N)$, the number of sample of class T_i in N.

- If A_k is continuous-valued *(think further)*, then we typically assume that the values have a Gaussian distribution with a mean μ and standard deviation σ defined by

$$g(x, \mu, \sigma) = \frac{1}{\sqrt{2\phi\sigma}} \cdot \exp(-\frac{(x-\mu)^2}{2\sigma^2}),$$

so that

$$p(x_k|T_i) = g(x_k, \mu T_i, \sigma T_i).$$

We need to compute $\mu \cdot T_i$ and $\sigma \cdot T_i$, which are the mean and standard deviation of values of attribute A_k for training samples of class T_i.

III. Laplace smoothing

The Laplace smoothing or Laplace correction is way of dealing with zero probability values. This estimator does not to improve the performance of an existing classifier. It is a way to solve a problem of missing data, that would prevent you from building a classifier.
Recall that we use the estimation

$$P(X|T_i) \approx \prod_{k=1}^{n} P(X_k|T_i).$$

based on the class independence assumption.
What if there is a class T_i, and X has an attribute value, x_k, such that none of the samples in T_i has that attribute value? In that case $P(x_k|T_i) = 0$, which results in $P(X|T_i) = 0$ even though $P(x_k|T_i)$ for all the other attributes in X may be large. But what happened if, say, there are no training samples? A zero probability cancels the effects of all of the other a posteriori probabilities on T_i.
We have to use the Laplace smoothing to avoid this problem. We can assume that our training set is so large that adding one to each count that we need would only make a negligible difference in the estimated probabilities, yet would avoid the case of zero probability values. If we have q counts to which we each add one, then we must remember to dd q to the corresponding denominator used in the probability calculation.

IV. Lab activity

In this assignment i have used Matlab R2022b to make my program. I started from the data processing, I build a naive Bayes classifier and then I improved the classifier with Laplace smoothing.

1) *TASK 1: Data pre-processing*
   I opened the data set, a file .txt, into Excel to change the values into integers greater or equal than 1. I chose the maximum values of the data set is 3, and the minimum is 1. In the *Fig.1* there are all values of our data set, and before to create the matrix there is a control to alert if there are values smaller than 1 or values outside the referee value, in this case for value bigger than 3.
2) *TASK 2: Build a naive Bayes classifier*
   This task is the most difficult because I created all matrices to use the Bayes theorem and runs the theorem with many for cycles.
   Firstly, I create a vector with four random numbers, I remove this lines by the data set and with this way I generate the training set, *Fig.2*.
   Then, I generate the test set, *Fig.3*, like difference between the data set and the training set.

   Secondly, I calculated three kinds of probabilities; priori, maximize and likelihood using the training set. I made two matrices for the last two probabilities, *Fig.5 and Fig.6*, and two variables for the priori probabilities,

Matrixset
14x5 double

	1	2	3	4	5
1	2	3	3	2	3
2	2	1	2	3	3
3	2	2	3	3	3
4	2	3	2	2	3
5	1	2	3	2	3
6	1	1	2	2	3
7	1	1	2	3	2
8	1	2	2	2	3
9	1	2	3	3	2
10	3	3	3	2	2
11	3	3	3	3	2
12	3	2	3	2	2
13	3	1	2	2	3
14	3	2	2	3	3

Fig. 1. Data set, all values for training set and test set

Trainingnset
10x4 double

	1	2	3	4
1	2	3	3	2
2	2	1	2	3
3	1	1	2	2
4	1	1	2	3
5	1	2	3	3
6	3	3	3	2
7	3	3	3	3
8	3	2	3	2
9	3	1	2	2
10	3	2	2	3

Fig. 2. Training set, one part of the data set with a random choice

Testset
4x5 double

	1	2	3	4	5
1	2	2	3	3	3
2	2	3	2	2	3
3	1	2	3	2	3
4	1	2	2	2	3

Fig. 3. Test set, the rested part of the data set, it is a different between the data set and the training set

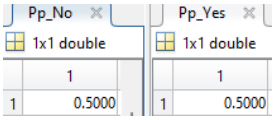

Fig. 4. Priori probability

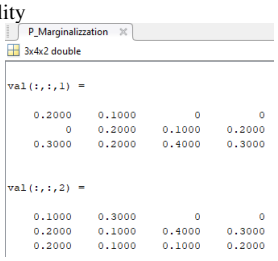

Fig. 5. Maximize probability

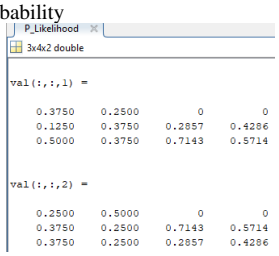

Fig. 6. Likelihood probability

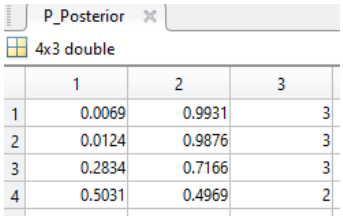

Fig. 7. Posteriori probability calculated with the other three ”probabilities”

Fig.4.

Thirdly, with all of those matrices i can calculate the posteriori probability, using the test set without the last column where there are the random answers from the data set, and then normalize the results for having a nice point of view over the Bayes's answers. The normalize changes the values between 0.0 and 1.0.
In the third column there are the values like the data set. This means, if you have '3' the answer for the case on the same row is ”YES” and whether you have '2' the answer is ”NO” *Fig. 7*.

Finally, in the *command window* (*Fig.8*) we can see the result of the test set said and what the program calculates. Essentially, there is a confront between the fifth Test set column (*Fig.3*) and the third posteriori probability (*Fig.7*). At the beginning in the command window (*Fig.8*), the program asks the size of the data set and the maximum values of the columns. It checks unless there are any errors inside the data set, for instance, if there are values smaller than 1. And then, whether it is all well, it compares the values and calculates the

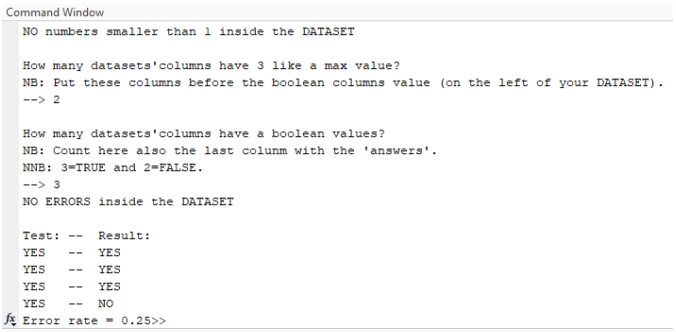

Fig. 8. Command window, the result from a test

error rate. The error rate is a number between 0 and 1 and it is how many times the program answers not well.

V. Conclusion

At the end of this first assignment, there are some points of view to analyze.

1) Without using the smoothing technique it will be difficult to answer to test set because sometimes it happened that inside likelihood matrix and maximize matrix to have a 0. This state generated an error in the posteriori matrix. All of this happened because our data set has few dates and it may not have all combination which the posteriori probability needs.
2) In this running there is a 0.25 error rate. This means the 25% of the answers are wrong. In our case only the fourth rows of the test set and the posteriori matrix have different answer.
3) This program could be deepened with many task. For example, it could ask the maximum values of the data set, in our case 3 but it may be more, it will build the answer of this values and it will answer how many column have this values.

I reported only one of the possible running and I saw in all of the run sections that there is an error rate between 25% and 75%. Up to now, I assume that results acceptable for this assignment.