NAIVE BAYES CLASSIFIER

Machine Learning 2018/19: Assignment 2

Torielli Davide Fusaro Fabio
EMARO - European Master on Advanced Robotics
DIBRIS - Dip. Informatica, Bioingegneria, Robotica, Ing. Dei Sistemi
Università Degli Studi Di Genova, November 2, 2018

I. INTRODUCTION

CLASSIFICATION is a model for recognition. With a given input, the role of the *classifier* is to categorize this input in one of the suitable class. For example, we have a patient record as input and we want to put the patient in one of the categories *healthy* or *at risk* or *ill*.
To do this, first we need a *training* phase with an already classified data. Then, we construct a rule that minimize a certain cost function (to be chosen).
In this work we implement in Matlab the *Naive Bayes Classifier* with and without the *Laplace additive smoothing* and then we show the results.

II. THE THEORY

A. The Naive Bayes Classifier

Given an input (observation) $\boldsymbol{x}$ we want to categorize it in one of the category *learned* with the training set. So we build a rule (the classifier) $y()$ that receives the observation $\boldsymbol{x}$ and output a *class* (the category) $\omega = y(\boldsymbol{x})$. To do this, the classifier must minimize the *expected probability of error* which means that it must choose the *class* which, given the information of the training set, has *probably* the minimum error. The Naive Bayes Classifier does this using a *naive* assumption (which means theoretically wrong):

$$Pr(x_1, \dots, x_k, \dots, x_d|\omega_i) \;=\; Pr(x_1|\omega_i) \dots Pr(x_k|\omega_i) \dots Pr(x_d|\omega_i),$$

$$k = 1 \dots d, \;\; i = 1 \dots N$$

Where each x_k is one of the attribute k of the observation $\boldsymbol{x}$, and ω_i is the i-th classification.
The i-th *decision boundary* $g_i(\boldsymbol{x})$ is:

$$g_i(\boldsymbol{x}) \;=\; Pr(\omega_i)Pr(\omega_i|\boldsymbol{x}) \;=\; Pr(x_1, \dots, x_d|\omega_i)$$

But, given the *naive assumption* $g_i(\boldsymbol{x})$ becomes:

$$g_i(\boldsymbol{x}) \;=\; Pr(\omega_i)[Pr(x_1|\omega_i) \dots Pr(x_d|\omega_i)] \;=\; Pr(\omega_i) \textstyle\prod_{j=1}^{d} Pr(x_j|\omega)$$

Now, given N different boundaries $g_i(\boldsymbol{x})$, we choose the one with the higher value and classify $\boldsymbol{x}$ with ω_i label. This means that ω_i is the most probable class (according to the trained set) for the observation. Obviously this classification can be not correct.

B. Training Phase

To do all the computations described in the previous subsection, we first have to *train* the classifier. This means to find the probabilities $Pr(\omega_i)$ and $Pr(x_k|\omega_i)$.
Due to the definition of prior probability of classes, $Pr(\omega_i)$ is:

$$Pr(\omega_i) = \frac{number\ of\ observations\ in\ class\ \omega_i}{number\ of\ observations\ in\ the\ training\ set}$$

and, due to the definition of conditional probability, $Pr(x_k|\omega_i)$ can be found counting how often each value of x_k occurs in class ω_i in the training set:

$$Pr(x_k = l_{q,k}|\omega_i) = \frac{number\ of\ times\ x_k = l_{q,k}\ in\ class\ \omega_i}{number\ of\ observations\ in\ class\ \omega_i},$$
$$k = 1 \dots d,\ \ i = 1 \dots N,\ \ q = 1 \dots M_k$$

where $l_{q,k}$ is the q-th possible value of the attribute k and M_k is the number (not known a priori) of these values for the attribute k , e.g. if attribute x_k has two possible values in the training set, $true$ and $false$, we have $M_k = 2$ (known only after we have checked all the training set), $l_{1,k} \equiv true$ and $l_{2,k} \equiv false$.

C. Laplace (additive) Smoothing

It may be the case that some values of some attributes appear in the observation $\boldsymbol{x}$ (that has to be classified) but do not appear in the training set. This would mean that these values would have zero probability, even if we know in advance that they can appear. For example, if in the training set I have only $true$ values for a certain x_k but I know in advance that these values can also be $false$, with the vanilla Naive Bayes Classifier I would have the probability of $false$ equal to zero.
The Laplace smoothing helps dealing with this problem. As said, we are assuming that we know M_k: the number of possible values for each attribute x_k. When we compute the conditional probability $Pr(x_k|\omega_i)$ in the training phase, we add the smoothing terms a and aM_k :

$$Pr(x_k = l_{q,k}|\omega_i) = \frac{(number\ of\ times\ x_k = l_{q,k}\ in\ class\ \omega_i) + a}{(number\ of\ observations\ in\ class\ \omega_i) + (aM_k)},$$
$$k = 1 \dots d,\ \ i = 1 \dots N,\ \ q = 1 \dots M_k$$

where a is a factor that represents the confidence we have in the data; $a < 1$ means that we trust more in data than the prior belief, $a > 1$ means we trust more in the prior belief.
Thanks to this a factor, now each probability will never be zero, so we will consider all the possible values of the attribute, even if they are not all present in the training data.

III. Our Work

Our work consists in the development of a Naive Bayes Classifier on a *weather data set*. So the goal is to give a description of the weather and to decide whether to go outdoor to play (e.g. tennis) or not. This was implemented with three tasks:

- **Task 1**: Data preprocessing.
- **Task 2**: Build a Naive Bayes Classifier.
- **Task 3**: Improve the classifier with Laplace (additive) smoothing.

A. Task 1

The data that we analyzed is a weather data set; the data is composed by 14 instances, 4 attributes and 2 classes. The first four columns are the features (in particular *Outlook, Temperature, Humidity, Windy*), the fifth is the target class (*Play*) as we can see in the Table I. The *Outlook* and the *Temperature* can assume three different values (*overcast*, *rainy* or *sunny* for the first one and *hot*, *cool* or *mild* for the second one). Instead the *Humidity* and the *Windy* have two different values (*high* or *normal* for the first one and *false* or *true* for the second one). The target class (*Play*) is binary: *yes* or *no*.
The task 1 processes the data converting all attribute values in numbers so they can be easily analyzed with Matlab.

Outlook	**Temperature**	**Humidity**	**Windy**	**Play**
overcast	hot	high	FALSE	yes

TABLE I: *Example of one instance of the weather data set*

B. Task 2

In this task we implemented the Naive Bayes Classifier. This was done applying the theory (II-A) the classifier is based on; in particular the task trains the classifier on the training set using, as said before, the last column as target. Then, the test set is classified with the *decision boundaries* $g_i(\boldsymbol{x})$ given the *naive assumption*. At the end, if the test set has the same number of columns as the training set, the last column is used to calculate the error rate between the obtained classifications and the real results.

C. Task 3

In this task, the previous algorithm is improved introducing the additive smoothing or Laplace smoothing. It is only modified the conditional probability formula adding a factor a for the trustiness and the known number of possible values for each attribute, so the probability will never be zero as explained in the theory (II-C).

IV. Experimental Results

Outlook	Temperature	Humidity	Windy	Play (test target)
rainy	mild	high	FALSE	yes
sunny	cool	normal	FALSE	yes
overcast	hot	normal	FALSE	yes
sunny	hot	high	FALSE	no

TABLE II: *First experiment test set*

Play (test target)	Play (without smoothing)	Play (a = 0.5)	Play (a = 1)	Play (a = 1.5)
yes	no	no	yes	yes
yes	yes	yes	yes	yes
yes	yes	yes	yes	yes
no	no	no	no	no

TABLE III: *First experiment results with different Naive Bayes Classifiers*

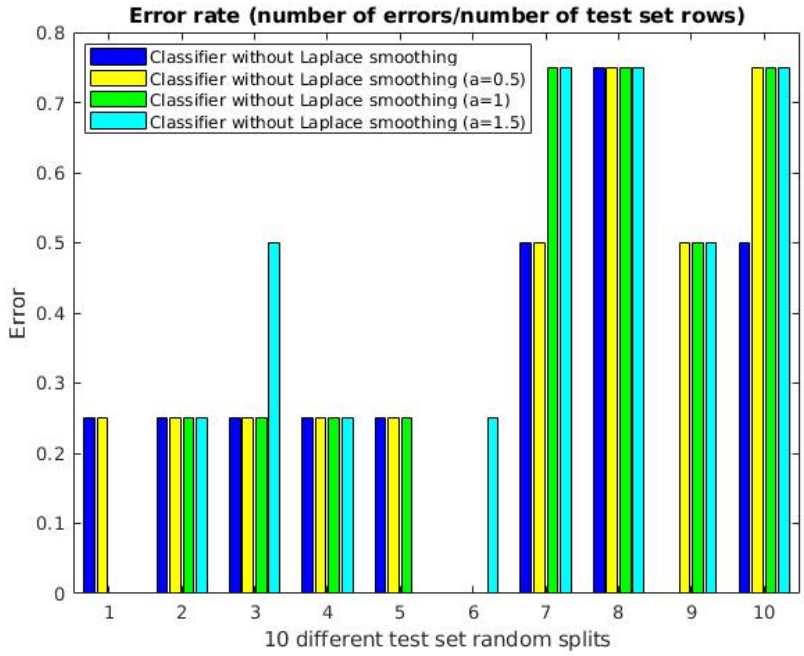

Fig. 1: *Error rate between prediction (classification) and real target*

The data, as described in III-A, has 14 instances; 10 chosen as training set and the remaining 4 as test set. This distinction is computed randomly 10 times. For the first experiment we can see the test set in Table II and the classification results for different kinds of Naive Bayes Classifier in Table III. In this case it is convenient to believe more in the prior belief than the data ($a \geq 1$).
In all the cases tried, the probabilities found with the training set are always different from 0 for at least one of the target *yes* or *no*. This means that the vanilla classifier never fails to give a classification, because when only one decision boundary is equal to zero, the reasonable choice is the other target.
So, the differences between using Laplace smoothing or not are not evident, even with different values of the trustiness. As we can see in the Fig. 1, sometimes Laplace smoothing improves the results but other times it worsens them. Moreover, big differences are not noticeable because training and testing set are very limited (and so the errors can have only 4 values from 0 to 1).