Machine Learning - Robotics Engineering
Report Assignment 1
Naive Bayes classifier

Francesca Canale 4113133

21/10/2019

Abstract

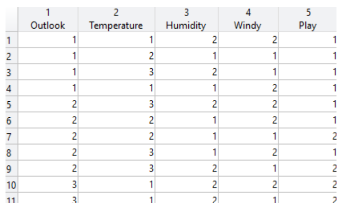
This assignment is based on naive Bayes classifier that is a probabilistic machine learning algorithm based on the Bayes Theorem. It is called "*naive*" because it operates on the assumption of independence of the features that is not always true. The requirement was to implement its algorithm as a MATLAB function. We had also to implement in a second function an improved version of the classifier that tries to overcome some of its limitations. The final results show that the first function works well in normal situations, but when we have in input a particular data set the improved version works better.

1 Introduction

The goal of this lab assignment was to build a naive Bayes classifier in MATLAB and to test it with two possible data sets. We had also to build an improved version of the classifier adding a Laplace smoothing.

2 Data Sets

We have been given two different data sets to work with: one about the influence of the weather on the decision whether to go play outdoor or not, the other about understanding if a mushroom is edible or poisoned having some information about its aspect. The first thing we had to do with these data sets was to convert their attribute values into integers so to be used to index matrices. In Fig. 1 there is an example of the first data set:



	1 Outlook	2 Temperature	3 Humidity	4 Windy	5 Play
1	1	1	2	2	1
2	1	2	1	1	1
3	1	3	2	1	1
4	1	1	1	2	1
5	2	3	2	2	1
6	2	2	1	2	1
7	2	2	1	1	2
8	2	3	1	2	1
9	2	3	2	1	2
10	3	1	2	2	2
11	3	1	2	1	2
12	3	3	2	2	2
13	3	2	1	2	1
14	3	3	1	1	1

Figure 1: Weather data set

- each column of the matrix represents a different feature of the weather;
- the same integer value in different columns has a different meaning, for example 1 in the first column means *"overcast"* and in the second column means *"hot"*;

- the last column is the one of the classes (in this case 1 means *"play"* and 2 means *"don't play"*);
- each row represents a different observation of the weather.

As we can see in Fig. 1 this weather data set has 4 features and 14 observations. The other data set, the mushroom one, is bigger and has 22 features and 8124 observations. After loading the data sets on a MATLAB script we had to split them randomly into training set and test set: two matrices that contain different rows of the data set. They have different row dimension and the test has one column less than the training set because it doesn't have the classes column. At this point the data is ready to be analyzed by the classifier.

2.1 Mushroom data set

For the mushroom data set it was considered necessary to apply some modifications. In this data set the classes column was the first one, so it has been moved to the end of the matrix in order to have the same format of the other data set. It was also considered necessary to delete the 11^{st} column because there were many missing values that altered the results.

3 Naive Bayes classifier

Naive Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem, used in a wide variety of classification tasks. The naive Bayes classifier operates on a strong independence assumption so that the probability of each feature being classified does not affect the probability of the other. Nevertheless, the results of the naive Bayes classifier are often correct and typical applications of this algorithm include filtering spam, classifying documents, sentiment prediction etc. [1][2]

3.1 Bayes' theorem

Bayes' Theorem finds the probability of an event occurring given the probability of another event that has already occurred. Bayes' theorem is stated mathematically as the following equation:

$$P(H|X) = \frac{P(X|H)P(H)}{\sum_i P(X|H_i)P(H_i)} \tag{1}$$

where H is a class variable and X is an observation vector, so we are finding the probability of event H, given that event X is true. [3]
In equation (1):

- $P(H|X)$ is the *a posteriori probability* of hypothesis H after observing X;
- $P(X|H)$ is the *likelihood* of observing X when H holds;
- $P(H)$ is the *a priori probability* of hypothesis H;
- the denominator is called *partition function*

In the case of the naive Bayes classifier we are making the naive assumption that:

$$P(X|H) = P(x_1, x_2 ... x_{\text{n}}|H) = P(x_1|H)P(x_2|H)...P(x_{\text{n}}|H) \tag{2}$$

3.2 Implementation

In order to implement a naive Bayes classifier in MATLAB, I created a function *naiveBayesClassifier()* that takes in input a training set (a nxd matrix), a test set (a mx$(d-1)$ matrix) and optionally the test set ground truth (a mx1 matrix). The function returns the classification and the error rate.
This function firstly check if the test set has one column less than the training one and if all the values in the matrices are integer grater or equal to 1. After that it starts computing all the probabilities necessary to train a naive Bayes classifier on the training set.

- The *a priori probability* of hypothesis H is computed as the ratio of the number of times the H class appears and the total number of observations in the training set. There will be an a priori probability for each class. For example if I assume to have the training set equal to the weather data set, I will have:

$$P(play = 1) = \frac{9}{14}, \qquad P(paly = 2) = \frac{5}{14}$$

- The *likelihood* of observing X when H holds is computed as the ratio of the number of times that X appears while H is true and the number of time the H class appears. In the same example of before I will have:

$$P(outlook = 2|play = 1) = \frac{3}{9}, \quad P(temperature = 1|play = 2) = \frac{2}{5}, \quad etc..$$

  I decided to save all the likelihoods in a 3-dimensional matrix: the function generates a matrix for each class, and in each matrix in position (i,j) there is P(feature j = i|H) as shown in the example tables (1) and (2).

4/9	2/9	6/9	3/9
3/9	3/9	3/9	6/9
2/9	4/9	0	0

Table 1: P(X|1)

0	2/5	1/5	3/5
2/5	1/5	4/5	2/5
3/5	2/5	0	0

Table 2: P(X|2)

Once the training is ended the function has to classify the test set: for each observation in the test set it returns the most probable class value. According to the Bayes theorem it has to compute the a posteriori probability for each observation vector and for all class values. So assuming to have as observation in the test set the last row of the weather data set, I will have:

$$P(play = 1|X) = \frac{P(outlook = 3|1)P(temp. = 3|1)P(humidity = 1|1)P(windy = 1|1)P(play = 1)}{P(X|play = 1) + P(X|play = 2)}$$

$$P(play = 2|X) = \frac{P(outlook = 3|2)P(temp. = 3|2)P(humidity = 1|2)P(windy = 1|1)P(play = 2)}{P(X|play = 1) + P(X|play = 2)}$$

Done this the function computes which a posteriori probability has higher value and the respective class value will be the classification. The error rate is computed only if there is the optional third input that is used as the target. The error is obtained as the ratio of the number of times in which the classification is different from the target, and the number of observations in the test set.
The *naiveBayesClassifier()* function does not make any assumption regarding the inputs, so can be used with all types of data sets.

4 Improving the algorithm

It is possible to improve the classifier by considering solution for some particular cases. For example if some values of some attributes appear in the test set but not in the training set: in this case the *naiveBayesClassifier()* function is not able to classify the test set and returns an error. An instance could be if an attribute x can be either 1 or 2 but in the training set only observations with $x = 1$ appear.
There are two ways to overcome this problem:

a. It is possible to add as first row of both training and test set a special row that contains the number of values of each attribute, in order to know if some value is missing. For example for the weather data set would be:
$$[3, 3, 2, 2, 2]$$

b. It is possible to introduce Laplace smoothing in the computations of the probabilities: so adding something that takes into account your prior belief and since you don't know anything the prior belief is that all data are equally probable. So the likelihood computation become:
$$P(X|H) = \frac{NumberOfTimesThatXAppearWhenHIsTrue + a}{NumberOfTimesHClassAppear + an}$$
Where n is the number of values of the X attribute and a express the level of trust: $a > 1$ means that you trust more your prior belief that the data and $a < 1$ means the opposite.

4.1 Implementation

To improve the classifier I implemented a MATLAB function *naiveBayesClassifierSmooth()* that works mainly as the previous function but has the upgrade (a) and (b) just explained.

5 Results

To evaluate the results I have run the script 1000 times in order to test my functions with different training and test sets. The resultant average error rates are:

- For the weather data set 0.3558 and 0.4093 with the smoothing;
- For the mushroom data set 0.0026 and 0.0355 with the smoothing;

It is clear that the Laplacian smoothing does not improve the results: this is because if the input training set has all the possible values of each feature, adding a Laplacian smoothing means distorting the real probabilities. Contrary, if the input training set has some missing values that instead appear in the test set, the result with the Laplacian is better than the result without it. Indeed I tried to build a training set based on the mushroom data set where the observations that have a 1 as value of the first feature where all in the test set: so in the training set it never appears the value 1 in the first column. In this case, while with the function *naiveBayesClassifier()* it is not possible to classify the test set, with function *naiveBayesClassifierSmooth()* the test set can be classified and the error rate is of 0.1297.

References

[1] Intrusion Detection using Naive Bayes Classifier with Feature Reduction - Dr. Saurabh Mukherjeea, Neelam Sharmaa - http://bit.ly/2Mzfevj

[2] How Naive Bayes Algorithm Works? - https://www.sciencedirect.com/science/article/pii/S2212017312002964

[3] Naive Bayes Classifiers - https://www.geeksforgeeks.org/naive-bayes-classifiers/

***** This report in its general parts was written together with Filippo Gandolfi-4112879 *****