Machine Learning: Assignment 1
Naive Bayes

Filippo Gandolfi 4112879

23/10/2019

Abstract

This assignment is based on naive Bayes classifier that is a probabilistic machine learning algorithm based on 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]

1 Introduction*

This assignment is based on Naive Bayes algorithm. This probabilistic algorithms is one of the most popular in machine learning and it's based on the conditional probability. The conditional probability is a measure of the probability of an event occurring given that another event has occurred. Also, in the assignment it is required a Laplace smoothing algorithm, that it is a technique used to smooth categorical data. Given an observation from a multinomial distribution with N trials, it returns a "smoothed" version of the data.

So, these are the goals of this assignment, we have to build a naive Bayes classifier in MATLAB and to test it with different data sets, and then we have also to build an improved version of this classifier adding a Laplace smoothing.

2 Data Sets

Our first dataset is a very simple database, that contains some old records of when it was possible to play golf and when it was impossible, depending on the weather, the temperature, the humidity and the wind, that are the 4 features. We have only 14 observations and it has only 2 classes that are "TRUE" or "FALSE" (possible or not). Our databse is defined like this:

```
Data Set Characteristics:  Multivariate, supervised
```

A multivariate database is essentially a combination of between-subjects and within-subjects database structures, and supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.

Technically the data file has in the first line the column names, Columns 1-4 are the input features and Column 5 is the target class.

The 4 attributes are described like this:

```
1) Outlook (overcast, rainy, sunny)
2) Temperature (hot, cool, mild)
3) Humidity (high, normal)
4) Windy (FALSE, TRUE)
```

[1] * Written with: Francesca Canale

So, the first implementation was to transform all the attributes and their values in *int*eger, starting from 1 to avoid problem with the 0. To do that we could implement it manually (with a simple CTRL-F manual input) or using a simple script algorithm (obviosuly raccomended for bigger datasets). One we have our database clean and ordered we can start our computations.

The first implementation with the database is to split it into two different datasets, one used for training and the other one for testing the algorithm.

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.

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, so we are finding the probability of event H, given that event X is true.
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*

[2]

3.2 Implementation

After some check, for being sure that we don't use a dirty database or ones with wrong sentences, we need the probability of each event. So we search the max value for each column, counting how many times it appears and pounded the value for the number of rows.

```
max_value(i) = max(trainingset(:,i));
prob_classes(j,i) = sum(trainingset(:,i)==j);
prob_events(j,i) = prob_classes(j,i)/n;
```

Then we create n matrices, with n represents the number of classes, in this first database, only two. This are the probability for each events to affect the ability to play or not. Then we divide them by the number of classes and we obtain the probability pounded by the possible solutions. Now our code for the training is set, we need to start develop the test part, based on the Naive assumption:

$$g_i(x) = P(\omega_i)[P(x_1|\omega_i)P(x_2|\omega_i)...P(x_d|\omega_i)] = P(\omega_i)\prod_{j=1}^{d} P(x_j|\omega_i) \tag{2}$$

[2]* Written with: Francesca Canale

First, we need to take the values present in the test set and match with the ones in the training set in order to obtain its probability. Then we work on this part of the algorithm

$$\prod_{j=1}^{d} P(x_j|\omega_i) \tag{3}$$

We compute the probability for each row (which represents a possible scenario):

```
prob_scenario(i,k) = prob_scenario (i,k) * prob_tot(val,i,k);
```

For the sake of simplicity we normalize the value simply by divide it by the sum of all the possibile $g_i(x)$ At the end we verify our results with the ones known in the training set and calculate the error rate of our machine.

4 Improving the classifier with Laplace smoothing

The third task of the assignment use the Laplace Smoothing, that it is a technique to smooth categorical data. Laplace Smoothing is introduced to solve the problem of zero probability. The "smoothness" returns the estimator:

$$\hat{\theta}_i = \frac{x_i + \alpha}{N + \alpha d} \qquad (i = 1, \ldots, d) \tag{4}$$

where α (in the code "a") is the smoothing parameter and n is the number of values of attribute x.

4.1 Implementation

For this we simply use the previuos code and substitute the new formula:

```
prob_tot_task3(j,i,k) = (prob_tot_task3(j,i,k)+a)/(prob_classes(k,d)+(a*n3(j)))
```

5 Results

At the end of our assignment we try to understand more deeply how much the algorithm could be reliable and what affects it. We run the algorithm several times (around one hundred) and the results are:

- naive Bayes Classifier (no smoothing): error rate $\sim 33\%$
- Classifier with Laplacian smoothing: error rate $\sim 38\%$

We understand that with a training set so little like this, it's very difficult to have precise and reliable results. The machine it isn't sufficient trained. Also, the Laplacian smoothing doesn't help to improve the results, because it works only in the case there are missed attributes, where the not smoothing fails.

References

https://towardsdatascience.com/introduction-to-naïve-bayes-classifier-fa59e3e24aaf

https://www.scalelive.com/multivariate-database.html

https://en.wikipedia.org/wiki/Additive_smoothing