Assignment1: Naive Bayes classifier

Saeed Kalateh, MSc Student, Robotic Engineering,

Abstract—In this assignment, a naive bayes classifier bayes classifier is trained using weather data set. First, data is preprocessed and imported to MATLAB software as a matrix. Next, the naive classifier is trained and its accuracy is evaluated. Finally, the classifier is improved using Laplace smoothing. To make code easier to test, the MATLAB script is written in the sections. These sections are explained in this report.

Index Terms—Naive Bayes, Bayes Theorem

1 Data preprocessing

Input data for this assignment is the Weather data set and the Weather data description. Data includes four attributes and two classes. All attribute values have been converted into integers more than or equal to one using text editor. The numbering is as follows:

- Outlook (overcast, rainy, sunny) : (1, 2, 3)
- Temperature (hot, cool, mild) : (1 ,2 , 3)
- Humidity (high, normal) : (1, 2)
- Windy (FALSE, TRUE) : (0, 1)

In order to check the calculations, a table have been drawn that depicts the values number of instances for each feature (Figure 1). Dividing the ”yes” columns by 9, which is the total number of ”yes”s, will result in conditional probabilities assuming the answer is ”yes”. Likewise, dividing ”No columns by 4, will result in conditional probabilities assuming the answer is ”No”.
Modified Weather data is imported into MATLAB. The \t is used as the delimiter and the table numbers are assumed to be float. Finally, data is saved in a matrix named ”data” and its size is saved in ”row” and ”column” variables, respectively.

		Play	
		Yes	No
outlook	1. Overcast	4	0
	2. rainy	3	2
	3. sunny	2	3

		play	
		Yes	No
Temperature	1. hot	2	2
	2. cool	3	1
	3. mild	4	2

		Play	
		Yes	No
Humidity	high	3	4
	normal	6	1

		Play	
		Yes	No
Windy	FALSE	6	2
	TRUE	3	3

Fig. 1. Number of instances for each feature

2 Dividing data set into train and test

This section asks for the number of training examples. Then it checks if the input number is less than ”rows” and more than 3. If this condition is satisfied, it assigns the ”n” number of data to training and the rest (”rows-n”) to the test. The train data is selected randomly, there fore, the training might yield different results at each run.

3 Training: Creating Lookup Matrices

Two matrices are created in this section:

3.0.1 f_prob

The probability of each feature is stored in this matrix. Each column is dedicated to features of one attribute in the same order of the numbers in section 1.

3.0.2 lookup_table

This matrix includes the conditional probabilities. Each attribute has two columns, one with ”yes” assumption and th other with ”No” assumption.

4 Test our Naive Bayes

In this section, Naive Bayes formula is implemented. Each test data iterates in the outer for loop. It flows into two inner for loops, where the prob_yes=P(Yes|conditions) and prob_no=P(No|conditions) are both calculated. Then prob_yes and prob_no are compared and the larger one is chosen as algorithm prediction. A validation matrix is also created, which is exactly like test_matrix, except it has a 6th column which is our predicted number.

5 Print the results

Finally, the 5th and 6th columns of validation matrix are compared and divided by the total number of rows. This yields the accuracy for our Naive Bayes algorithm which is printed in the end.

6 Laplace (additive) smoothing

The code for this task is pretty similar to what mentioned above. Only number of levels needs to be calculated. The number of levels for all attributes is saved to a matrix named "levels". It is simply the maximum number in each column, according to our modifications in section 1.
Before calculation of conditional probability matrices, the v parameter is updated using "levels" vector and it is added to the formula.