Naïve Bayes

Tutorial/Lab work: Week 4

Instructions: This is a coding exercise. Kindly stage+commit+push the `MATLAB` code to the "`week04`" directory on your INM431 Github repository. Grading: All attempts will get 1 point. **Please make sure that the document is pushed to Github by 03/11/2021**.

Background

Naïve Bayes is a simple technique for building classifiers. Such models assign class labels to instances represented as vectors of feature values, where the class labels are drawn from some finite set. Naive Bayes is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a particular feature is independent of the value of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round and roughly three inches in diameter. A Naïve Bayes classifier assumes that each of these features contribute independently to the probability that the fruit is an apple, regardless of any possible correlations between colour, roundness and diameter. For some types of probability models, naive Bayes can be trained very efficiently in a supervised learning setting. In many practical applications, parameter estimation for Naïve Bayes uses maximum likelihood. Despite their simple design and apparently oversimplified assumptions, naive Bayes classifiers have worked quite well in many complex real-world situations. In 2004, an analysis of the Bayesian classification problem showed that there are sound theoretical reasons for the apparently implausible efficacy of naive Bayes classifiers[1]. Still, a comprehensive comparison with other classification algorithms in 2006 showed that Naïve Bayes classification is outperformed by other approaches, such as Boosted Trees or Random Forests[2](we will be covering this week07). A major advantage of Naïve Bayes is that it only requires a small amount of training data to estimate the parameters for classification.

[1] Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004

[2] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168, 2006

Setup:
Consider the following vector:
[likes shortbread, likes lager, eats porridge, watches England playing football, nationality]
A vector $x = [1, 0, 1, 0, 1]$ would describe that a person likes shortbread, does not like lager, eats porridge, does not watch England playing football, and is a national of Scotland.
The right-most element in the vector denotes the class that we want to predict, and it can take two values: 1 for Scottish, 0 for English.
We will perform simple naive Bayes to predict such data.

- Copy `naiveBayesIntro.m` into any folder of your choice;
- Open `MATLAB` and inside it click on `naiveBayesIntro.m`;
- Left-click the editor and press $F5$ to run the code.

You should see two results: one represents $P(\text{Scottish}|X)$ and the other represents $P(\text{English}|X)$.

Can you work out what the `MATLAB` code does?

Change the test point on line 25 to use different values and check the results.

Gaussian Naïve Bayes

Let us use the Gaussian Naïve Bayes to classify whether a person is male or female based on the following features: height, weight and foot size, with training set:

gender	height(feet)	weight(lbs)	footsize(inches)
male	6	180	12
male	5.92(5′11")	190	11
male	5.58(5′7")	170	12
male	5.92(5′11")	165	10
female	5	100	6
female	5.50(5′6")	150	8
female	5.42(5′5")	130	7
female	5.75(5′9")	150	9

Assuming a Gaussian distribution, calculate the mean and variance of each feature given male or female.

From the frequencies in the training data or your knowledge of the frequencies in the larger population, calculate $P(\text{male})$ and $P(\text{female})$.

Given the information in the test data `Sample1` and `Sample2` below to be classified each either as male or female, calculate $P(C = \text{male}|\text{Sample1})$, $P(C = \text{female}|\text{Sample1})$, $P(C = \text{male}|\text{Sample2})$, $P(C = \text{female}|\text{Sample2})$, using:

$$P(C_k|x_1, \cdots, x_n) = \frac{1}{Z} P(C_k) \prod_{i=1}^{n} P(x_i|C_k)$$

	gender	height(feet)	weight(lbs)	footsize(inches)
Sample1	?	6	130	8
Sample2	?	5.6	167	9

The equation of the Gaussian distribution can be applied using `normpdf` with mean and variance calculated earlier from the training data, to obtain the conditional probability densities, e.g. $P(\text{height=6}|\text{C=male})$, which could be a number greater than 1. Alternatively, use `cdf` from `MATLAB` to calculate the area under the curve as the probability of height being approximately 6, which cannot be a number greater than 1.

Apply the Maximum a Posteriori (MAP) decision rule to decide whether `Sample1` and `Sample2` are male or female

Hints

The Naïve Bayes code can return `NaN` as the class conditional probabilities of the two classes $P(\text{female}|x)$ and $P(\text{male}|x)$ given a new test input x. Notice that the function corresponding to $P(x|\text{female})$ and $P(x|\text{male})$ needs to be changed since we are now dealing with continuous (real-valued) data as opposed to discrete (categorical) data which was originally the case in the code.

You are required to compute the probability of each of the attributes of the input feature x assuming that these arise out of a Gaussian distribution. As stated in the question, this would involve first computing the mean and standard deviation of each of the attributes for each of the two classes, i.e. for the distributions $\mathcal{N}(\mu_{\text{height}}, \sigma^2_{\text{height}}|\text{female})$, $\mathcal{N}(\mu_{\text{height}}, \sigma^2_{\text{height}}|\text{male})$ (and the equivalent distributions for the other two attributes). Following this, you will be able to compute $P(x|\text{female})$ and $P(x|\text{male})$ for using these 6 distributions (over 2 classes, and 3 attributes) which you can then apply to the Naive Bayes equation as

$$P(\text{female}|x) = \frac{P(x|\text{female})P(\text{female})}{\sum_{\text{gender}\in(\text{female,male})} P(x|\text{gender})P(\text{gender})}$$

and,

$$P(\text{male}|x) = \frac{P(x|\text{male})P(\text{male})}{\sum_{\text{gender}\in(\text{female,male})} P(x|\text{gender})P(\text{gender})}$$

The values of $P(x|\text{female})$ and $P(x|\text{male})$ according to the Gaussian density (i.e. continuous) function can be approximated by binning (or discretising) the Gaussian function over a large number of points uniformly drawn from the distribution. This is similar to calculating, by using the `MATLAB` function `cdf`, the area under the curve as the probability that x will assume approximately a given value. Alternatively, the `MATLAB` function `normpdf` can be used to produce density values from the Gaussian, which can be numbers greater than one (not to be confused with probabilities), which can be used directly with the Maximum A Posteriori (MAP) decision rule, that is, `argmax`.

References

Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168, 2006.

Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.