Theory: Naive Bayes classifier

In this topic, we'll discuss the theory behind the Naive Bayes classifiers.

The Naive Bayes classifiers are a set of classification algorithms for predictive modeling based on **Bayes Theorem** and probability theory. According to the principle that they're built on, every pair of features being classified is independent of each other.

A Naive Bayesian model is one of the easiest to build and it may be used for a wide variety of classification tasks.

Current topic:

Naive Bayes classifier ...

Topic depends on:

✕ Bayes theorem ...

✕ Classification performance metrics ...

§1. Bayes theorem

Bayes theorem finds the probability of an event given the probability of another event that has already occurred. This equation shows how Bayes theorem is stated mathematically:

$$P(A \mid B) = \frac{P(B|A)P(A)}{P(B)}$$

In general, it's used to find the probability of the A event, given that the B event has already occurred. Where:

- B is the **evidence** (an attribute value of an unknown instance) and the A is the hypothesis.
- $P(A)$ is the **"a priori" probability** of the A (the probability of an event before the evidence is seen).
- $P(A|B)$ is the **"a posteriori" probability** of the A (the probability of event after the evidence is seen).
- $P(B)$ is the a priori probability of B.
- $P(B|A)$ is the "a posteriori" probability of B.

§2. Naive Bayes classifier

A Naive Bayes classifier is a probabilistic machine learning model based on Bayes theorem that is used for various classification tasks. The assumption here is that the predictors (features) are **independent**. What does it mean? Well, two features are 'independent' if one doesn't affect the other.

Imagine you have a dataset containing some information about patients: weight, height, pulse, cholesterol level, and their postcode. For this dataset, it's reasonable to suggest that weight and postcode are independent because these features are not connected. Are all the features independent? Unfortunately, no. We can suggest that as the cholesterol level increases, up goes the weight.

That's why the method is called '*naive*' — the assumption that all the features are independent is quite naive.

Let's have a look at the detailed example where we consider the problem of receiving a bonus payment. We have a dataset represented as a table:

#	Skillset	Relationships with the boss	Performance	Bonus payment
1	Low	Bad	Low	No
2	High	Good	High	Yes
3	High	Good	Low	Yes
4	Low	Good	Low	No
5	High	Bad	High	Yes
6	Low	Bad	High	No

Here we determine if an employee will get a bonus according to their particular features. The columns here show these features and the rows show the individual entries. For example, looking at the #1 row, we see that the employee is not going to get the bonus because of their poor skills, performance, and relationships with the boss. We can consider that these predictors are independent, meaning if the skills aren't good enough it does not necessarily mean that the performance will be bad. The second assumption made here is that all the predictors have an equal effect on the outcome. It means that all predictors are important.

We've already seen Bayes theorem above, but since we have this example, we need to recast the formula:

$$P(y \mid X) = \frac{P(X|y)P(y)}{P(X)}$$

The variable y is a class variable. It shows whether an employee deserves a bonus payment. The variable X shows the features.

We can represent X as $X = (x_1, x_2, x_3, \ldots, x_n)$, where x_n represents a particular feature (here: performance/skills/relationships with the boss). The classifier needs to accomplish the classification task based on feature values. Firstly, the algorithm processes the dataset to approximate the probability of a y class for a given set of x_n feature values that can be expressed as the following equation:

$$P(y \mid x_1, ..., x_n) = \frac{P(x_1 \mid y)P(x_2 \mid y)...P(x_n \mid y)P(y)}{P(x_1)P(x_2)...P(x_n)}$$

As a result, we can get the values by looking at the dataset and by substituting them in this equation. The denominator remains for each entry. We can get rid of it and simplify the equation to a proportionality:

$$P(y \mid x_1, ..., x_n) \propto P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

In this equation, each $P(x_i \mid y)$ can be calculated based on the assumption of the distribution of the features.

In our dataset, there are only two y outcomes (yes or no), but if the classification is multivariate, then we have to find the y class that has the maximum probability. The algorithm estimates whether the probability of a given sample with the known feature values belongs to a certain class by choosing the y that leads to the largest $P(x_i \mid y)P(y)$ value.

That's how we can get the class given the predictors:

$$y = argmax_y \, P(y) \prod_{i=1}^{n} P(x_i \mid y)$$

It's called **the maximum a posteriori decision rule**, it means that the probabilistic classifier predicts the class with the highest conditional probability for a given feature vector.

Naive Bayes classifier processes the dataset to calculate the $P(y)$ class probabilities as well as the conditional probabilities, which define the frequency of each feature value for a given class value divided by the frequency of instances with that class value.

Table of contents:

§3. The Naive Bayes classificator in use

Let's look closer at the probability calculation. For example, we have a fruit dataset with 3 classes (orange, banana, etc.) and 3 features (it may be sweet/yellow/long). Therefore, we have 300 oranges (150 of them are sweet), 500 bananas with 450 of them being yellow and 350 sweet, and 200 fruits of other kinds:

Class	Yellow	Long	Sweet	Total
Orange	300	0	150	300
Banana	450	400	350	500

Other	50	100	150	200
Total	800	500	650	1000

If we get the features only (sweetness/length/color), without knowing the class, we can calculate the probability for the fact that the fruit is an orange/banana/other fruit.

Imagine, that our fruit is yellow, long, and sweet. Probability calculation consists of 4 steps:

1. Calculate the probability of whether the unknown fruit is a banana. The probability calculation for the 'Banana' class is based on three features ('sweet', 'yellow', 'long'):

$$P(\text{Banana}|\text{Long, Sweet, Yellow})$$

2. Let's start with the numerator and substitute all the values in the equation:

$$P(\text{Long}|\text{Banana}) = 400/500 = 0.8$$

$$P(\text{Sweet}|\text{Banana}) = 350/500 = 0.7$$

$$P(\text{Yellow}|\text{Banana}) = 450/500 = 0.9$$

$$P(\text{Banana}) = 500/1000 = 0.5$$

By multiplying the values according to the equation, we will get:

$$0.8 * 0.7 * 0.9 * 0.5 = 0.252$$

3. Ignore the denominator, as it will be the same for all the subsequent calculations.

4. Let's do the same calculations for other classes:

$$P(\text{Orange}|\text{Long, Sweet, Yellow}) = 0$$

$$P(\text{Other}|\text{Long, Sweet, Yellow}) = 0.01875$$

Since 0.252 is bigger than 0.01875, a naive Bayesian algorithm classifies this long, sweet, yellow fruit as a banana.

This method requires training as the algorithm uses a labeled dataset to build a table.

As we can see in the example above, the algorithm consists of simple calculations — multiplication and division. Once the frequency arrays have been calculated, the classification of the unknown fruit involves only calculating the probabilities for all classes and then choosing the highest probability.

Despite its simplicity, a naive Bayesian algorithm can be surprisingly accurate. For example, it has been found that it can be successfully used to filter spam.

§4. Spam filtering

Spam comes from messaging systems that are sending unsolicited messages, usually. It's advertising or fraud messages aimed to gather personal information. Let's see how the filtering works.

For example, you received a letter with the text "Please enter your bank account number here" and then marked it as "spam". The mail server remembered it, each word from the letter was taken and put in the array. Then you received a mail with the text "The meeting is in the bank tomorrow, please come." It is legit. The server took it into account; it went over every

word and marked them as well. The result is shown in the chart. For example, you clicked "spam" on the first letter — each word from this letter got one point in the column of the same name. The "legit" section works the same way.

	Spam	Legit
Please	1	1
Enter	1	0
Your	1	0
Bank	1	1
Account	1	0
Number	1	0
Here	1	0
Meeting	0	1
Tomorrow	0	1
Come	0	1

The next mail you get is "Please, send me your bank account number before the meeting". So what the server would do? It would take every word from the mail and look at the statistics. The words "account", "number" are spam. The words "please" and "bank" are both spam and "legit" with the same score — the server will consider it spam. The server doesn't know the rest of the words. Total — 2 spam words ("account", "number") versus 1 non-spam ("meeting"), 3 words in total.

According to Bayes' theorem, the probability that this is spam — 66.6%. The email will be moved to the "Spam" folder. And if the user clicks the "Legit" button, the algorithm will also remember this, and it will work more accurately next time.

That would be machine learning. Now it is ready to protect you from spam. The more mail you receive, the higher spam filtering accuracy would be.

§5. Summary

Summarizing the above, we can say that:

- The Naive Bayes algorithms are classification algorithms based on Bayes theorem used for probabilistic modeling. It's fast and easy but it also can be very accurate.
- Naive Bayes algorithms are widely used in *spam filtering* (identifying spam e-mail), but other than that it can be used in *text classification*, *sentiment analysis (*social media analysis) to identify positive and negative customer sentiments, and for *recommendation systems* (filter unseen information and predict whether a user would like a given resource or not).
- Naive Bayes classifier is based on supervised learning.
- To get better results, the predictors have to be independent.

Now, let's practice!

Report a typo

10 users liked this theory. **0** didn't like it. **What about you?**

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion