A practical explanation of a Naive Bayes classifier

The simplest solutions are usually the most powerful ones, and Naive Bayes is a good proof of that. In spite of the great advances of the Machine Learning in the last years, it has proven to not only be simple but also fast, accurate and reliable. It has been successfully used for many purposes, but it works particularly well with natural language processing (NLP) problems.

Naive Bayes is a family of probabilistic algorithms that take advantage of probability theory and Bayes' Theorem to predict the category of a sample (like a piece of news or a customer review). They are probabilistic, which means that they calculate the probability of each category for a given sample, and then output the category with the highest one. The way they get these probabilities is by using Bayes' Theorem, which describes the probability of a feature, based on prior knowledge of conditions that might be related to that feature.

Search ...

Recent Posts

169 Shares

- An introduction to Support Vector Machines (SVM) 97
- A practical explanation of a Naive Bayes classifier 25
- Analyzing 10 years of startup news with Machine Learning 7
- Creating machine learning models to analyze startup news 1
- Filtering startup news with Machine Learning

Categories

- Applications
- Guides

We're going to be working with an algorithm called **Multinomial Naive Bayes**. We'll walk through the algorithm applied to NLP with an example, so by the end not only will you know *how* this method works, but also *why* it works. Then, we'll lay out a few advanced techniques that can make Naive Bayes competitive with more complex Machine Learning algorithms, such as SVM and neural networks.

A simple example

Let's see how this works in practice with a simple example. Suppose we are building a classifier that says whether a text is about sports or not. Our training set has 5 sentences:

Text	Category
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
	Not sports

y close game belong to?

How To
News
Text Classification

Since Naive Bayes is a probabilistic classifier, we want to calculate the probability that the sentence "A very close game is Sports", and the probability that it's *Not Sports*. Then, we take the largest one. Written mathematically, what we want is $P(Sports|a\ very\ close\ game)$— the probability that the category of a sentence is *Sports* given that the sentence is "A very close game".

That's great, but how do we calculate these probabilities?

Let's dig in!

Feature engineering

The first thing we need to do when creating a machine learning model is to decide what to use as features. We call **features** the pieces of information that we take from the sample and give to the algorithm so it can work its magic. For example, if we were doing classification on health, some features could be a person's height, weight, gender, and so on. We would exclude things that maybe are known but aren't useful to the model, like a person's name or favorite color.

In this case though, we don't even have numeric features. We just have text. We need to somehow convert this text into numbers that we can do calculations on.

quencies. That is, we ignore word
y document as a set of the words it
ch of these words. Even though it
surprisingly well.

Bayes' Theorem

Now we need to transform the probability we want to calculate into something that can be calculated using word frequencies. For this, we will use some basic properties of probabilities, and Bayes' Theorem. If you feel like your knowledge of these topics is a bit rusty, read up on it and you'll be up to speed in a couple of minutes.

Bayes' Theorem is useful when working with conditional probabilities (like we are doing here), because it provides us with a way to reverse them:

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

In our case, we have $P(sports|a\ very\ close\ game)$, so using this theorem we can reverse the conditional probability:

$$P(sports|a\ very\ close\ game) = \frac{P(a\ very\ close\ game|sports) \times P(sports)}{P(a\ very\ close\ game)}$$

Since for our classifier we're just trying to find out which category has a bigger probability, we can discard the divisor —which is the same for both categories—

$rts) \times P(Sports)$

$$P(a\,very\,close\,game|Not\,Sports) \times P(Not\,Sports)$$

This is better, since we could actually calculate these probabilities! Just count how many times the sentence *"*A very close game*"* appears in the *Sports* category, divide it by the total, and obtain $P(a\,very\,close\,game|Sports)$.

There's a problem though: "A very close game" doesn't appear in our training set, so this probability is zero. Unless every sentence that we want to classify appears in our training set, the model won't be very useful.

Being Naive

So here comes the *Naive* part: we assume that every word in a sentence is **independent** of the other ones. This means that we're no longer looking at entire sentences, but rather at individual words. So for our purposes, "this was a fun party" is the same as "this party was fun" and "party fun was this".

We write this as:

$$P(a\,very\,close\,game) = P(a) \times P(very) \times P(close) \times P(game)$$

This assumption is very strong but super useful. It's what makes this model work eled. The next step is just applying

$$\times P(very|Sports) \times P(close|Sports) \times$$

And now, all of these individual words actually show up several times in our training set, and we can calculate them!

Calculating probabilities

The final step is just to calculate every probability and see which one turns out to be larger.

Calculating a probability is just counting in our training set.

First, we calculate the a priori probability of each category: for a given sentence in our training set, the probability that it is *Sports* P(Sports) is ⅗. Then, P(Not Sports) is ⅖. That's easy enough.

Then, calculating $P(game|Sports)$ means counting how many times the word "game" appears in *Sports* samples (2) divided by the total number of words in *sports* (11). Therefore, $P(game|Sports) = \frac{2}{11}$

However, we run into a problem here: "close" doesn't appear in any *Sports* sample! That means that $P(close|Sports) = 0$. This is rather inconvenient since we are going to be multiplying it with the other probabilities, so we'll end up with $P(a|Sports) \times P(very|Sports) \times 0 \times P(game|Sports)$. This equals 0, since in whole calculation is nullified. Doing rmation at all, so we have to find a

How do we do it? By using something called Laplace smoothing: we add 1 to every count so it's never zero. To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1. In our case, the possible words are `['a', 'great', 'very', 'over', 'it', 'but', 'game', 'election', 'clean', 'close', 'the', 'was', 'forgettable', 'match']`.

Since the number of possible words is 14 (I counted them!), applying smoothing we get that $P(game|sports) = \frac{2+1}{11+14}$. The full results are:

Word	P(word | Sports)	P(word | Not Sports)
a	$\frac{2+1}{11+14}$	$\frac{1+1}{9+14}$
very	$\frac{1+1}{11+14}$	$\frac{0+1}{9+14}$
close	$\frac{0+1}{11+14}$	$\frac{1+1}{9+14}$
game	$\frac{2+1}{11+14}$	$\frac{0+1}{9+14}$

see who is bigger:

$$P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times P(Sports)$$
$$= 4.61 \times 10^{-5}$$
$$= 0.0000461$$

$$\mathrm{P(a \text{—} Not\ Sports)} \times P(very|Not\ Sports) \times P(close|Not\ Sports) \times P(game|Not\ Sports) \times P(Not\ Sports)$$
$$= 1.43 \times 10^{-5}$$
$$= 0.0000143$$

Excellent! Our classifier gives “A very close game” the **Sports** category.

Advanced techniques

There are many things that can be done to improve this basic model. These techniques allow Naive Bayes to perform at the same level as more advanced methods. Some of these techniques are:

- **Removing stopwords**. These are common words that don’t really add anything to the categorization, such as a, able, either, else, ever and so on. So for our purposes, *The election was over* would be *election over* and *a very close game* would be *very close game.*
- **Lemmatizing words**. This is grouping together different inflections of the d, and so on would be grouped es of the same word. le words like we did here, we could tch” and “close election”.

- **Using TF-IDF**. Instead of just counting frequency we could do something more advanced like also penalizing words that appear frequently in most of the samples.

Final words

Hopefully now you have a better understanding of what Naive Bayes is and how it can be used for text classification. This simple method works surprisingly well for this type of problems, and computationally it's very cheap. If you're interested in learning more about these topics, check out our guide to machine learning and our guide to natural language processing.

By Bruno Stecanella | May 25th, 2017 | News | 2 Comments

About the Author: Bruno Stecanella

Engineering student and primate enthusiast. I write code and blog about monkeys. Sometimes about other stuff, too.

2 Comments

Jennifer Clark June 3, 2017 at 9:20 am - Reply

Really great post – thank you

Yesudeep Mangalapilly June 20, 2017 at 1:32 pm - Reply

Fantastic explanation. Could you please add this to Wikipedia?
The world needs this there.

Leave A Comment

Comment...

Website

POST COMMENT




169
Shares

97

25

7

1

Signup for our Newsletter

Receive new cool Machine Learning posts and updates!

Subscribe now