Machine Learning

A practical explanation of a Naive Bayes classifier

by Bruno Stecanella

May 25, 2017 · 6 min read

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 tag of a text (like a piece of news or a customer review). They are probabilistic, which means that they calculate the probability of each tag for a given text, and then output the tag 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.

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

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 data has 5 sentences:

Text	Tag
"A great game"	Sports
"The election was over"	Not sports
"Very clean match"	Sports
"A clean but forgettable game"	Sports
"It was a close election"	Not sports

Now, which tag does the sentence *A very close game* belong to?

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 tag 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 text 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

So what do we do? Simple! We use **word frequencies**. That is, we ignore word order and sentence construction, treating every document as a set of the words it contains. Our features will be the counts of each of these words. Even though it may seem too simplistic an approach, it works 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 tag has a bigger probability, we can discard the divisor —which is the same for both tags— and just compare

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

with

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

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* tag, 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 data, so this probability is zero. Unless every sentence that we want to classify appears in our training data, the model won't be very useful.

Being Naive

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 well with little data or data that may be mislabeled. The next step is just applying this to what we had before:

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

And now, all of these individual words actually show up several times in our training data, 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 data.

First, we calculate the *a priori* probability of each tag: for a given sentence in our training data, 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* texts (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* text! 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 a multiplication, if one of the terms is zero, the whole calculation is nullified. Doing things this way simply doesn't give us any information at all, so we have to find a way around.

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

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	$\frac{2+1}{11+14}$	$\frac{0+1}{9+14}$

Now we just multiply all the probabilities, and see who is bigger:

$$P(a|Sports) \times P(very|Sports) \times P(close|Sports) \times P(game|Sports) \times P(Sports)$$
$$= 2.76 \times 10^{-5}$$
$$= 0.0000276$$

$$\mathrm{P(a} \quad |Not\,Sports) \times P(very|Not\,Sports) \times P(close|Not\,Sports) \times P(game|Not\,Sports) \times P(Not\,Sports)$$
$$= 0.572 \times 10^{-5}$$
$$= 0.00000572$$

Excellent! Our classifier gives "A very close game" the **Sports** tag.

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 classification, 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 same word. So election, elections, elected, and so on would be grouped together and counted as more appearances of the same word.
- Using n-grams. Instead of counting single words like we did here, we could count

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.

Machine Learning

Posts you might like...

Machine Learning

Have something to say?

blanc • a year ago

Hi Bruno, Thanks for the post!
What if the test document contains words not in the vocabulary which was created during training phase?

11 • Reply • Share ›

Faiyaad Alam • a year ago

Thank you for your post. Can you help me understand the difference between the total word count being 10 and possible words being 14? It just has me a little confused because shouldn't the total word count = possible words? Thanks.

1 • Reply • Share ›

Sabbir Ahmed • 3 months ago

Thanks a lot for this great tutorial.

• Reply • Share ›

loveleen Kumar • 3 months ago

Thank you Bruno for the post, As I am a learner, I have one question regarding Laplace smoothing. What is the need to add number 14 numbers in the denominator ? In which case it's probability will be more than 1 ?

• Reply • Share ›

Harold Inacay • 6 months ago

Thanks Bruno for this clear explanation of Naive Bayes.

• Reply • Share ›

Kai Wang • 10 months ago

Hi Bruno, very nice post!
One question: ' To balance this, we add the number of possible words to the divisor, so the division will never be greater than 1. '

word types from Sports AND Not Sports to divisor? but not just the ones from 'Sports'?
Thanks

• Reply • Share ›

Fred • a year ago

Great summary. I'm taking an intro machine-learning course on Udacity and this is decent supplemental reading. Cheers!

• Reply • Share ›

Niyongabo Eric • a year ago

hi Bruno, thanks for the article.
i understand well naive theory, next time i think it would be better to see both theory and practice(code). why to do so is i am new to machine learning so i just need to understand both

• Reply • Share ›

Mihai • a year ago

Hi Bruno,
This is an awesome guide! I think I am missing something obvious, but how did you get to 2.76 and 0.572 after multiplying all the probabilities? also, why did you multiply them by 10 raised at power -5? (just before explaining the advanced techniques).

Thanks!

• Reply • Share ›

Bruno Stecanella ➚ Mihai • a year ago

The results of the multiplication are 0.0000276 and 0.00000572 -- I wrote them first in scientific notation so they would be easier to compare.

Hope this helps

• Reply • Share ›

Afshin • a year ago

It is one of the best practical explanation of a Naive Bayes classifiers.
Thanks

• Reply • Share ›

Sufian Abu • a year ago

Bruno, This is an excellent explanation. Finally, I really

Ananya shrivastava • a year ago

Great Job!! It was really helpful for me to understand "Naive Bayes". Thankyou:)

Reply • Share ›

Praveen • a year ago

Made it simple, great articulation! thanks for sharing!

Reply • Share ›

mayank • a year ago

Really helpful thanks bro ....can u explain the implementation of tf idf in extension of context of this example it would be really heplful.. Thanks

Reply • Share ›

Bruno Stecanella ➚ mayank • a year ago

Alright, to make it clear, first a bit on tf-idf. Check out this extremely short explanation on how it works. The important part is the example they give at the end, which I'll just shamelessly copy here:

> *Consider a document containing 100 words wherein the word cat appears 3 times. The term frequency (i.e., tf) for cat is then (3 / 100) = 0.03. Now, assume we have 10 million documents and the word cat appears in one thousand of these. Then, the inverse document frequency (i.e., idf) is calculated as log(10,000,000 / 1,000) = 4. Thus, the Tf-idf weight is the product of these quantities: 0.03 * 4 = 0.12.*

Now, how do we put this to use in Naive Bayes?

see more

2 Reply • Share ›

Vikram • a year ago

Really helpful

Reply • Share ›

Rameez Nizar • a year ago

Excellent article. Found it super helpful..Thanx

Reply • Share ›

classifier", available online at:
https://monkeylearn.com/blo... [&#8230]

• Reply • Share ›

Francisco Ottonello • a year ago

Good post, it helped me to understand naive Bayes.

• Reply • Share ›

Knut • 2 years ago

Hi Bruno, great post. Just out of curiosity, how would the divisor [P(a very close game)] be calculated? Thanks

• Reply • Share ›

Bruno Stecanella ➚ Knut • a year ago

Remember, we never actually calculate this divisor. The actual number would be [how many times 'a very close game' appears in the dataset] / [total number of sentences in the dataset]. However, this just yields 0 if the sentence doesn't actually appear in the dataset, so it wouldn't be very useful. Luckily for us, we can cancel out the divisor: since max(A/C, B/C) = max (A, B) / C we can compare A and B directly. We are interested in who is larger, and not in the actual value of the max

Hope this helped!

• Reply • Share ›

Rendy • 2 years ago

Hey Bruno, if I have a dataset for sentences and words. Is it better using words as a dictionary rather than sentences? I tried to classify emotion based on text, and I found incorrect result using sentences as my dataset.

• Reply • Share ›

Bruno Stecanella ➚ Rendy • 2 years ago

I'm not sure I understand your question. Could you explain a little more about your problem? Cheers!

• Reply • Share ›

pavan • 2 years ago

Great help to understand Navie Bayes from scratch.

Cristopher Garduno • 2 years ago

This was great for someone with no experience with Naive Bayes, like myself. Thank you!

Reply • Share ›

Juergen Trittin • 2 years ago

I think there is a typo in the section "A simple example".

It should be:
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.

Instead of:
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.

Reply • Share ›

Federico Pascual ➚ Juergen Trittin • 2 years ago

Thanks! Fixed!

Reply • Share ›

Sumit Roy • 2 years ago

I have used the Stop words and tf/idf techniques while working on data clustering algorithms like 10 years back. Works like a charm.

Reply • Share ›

Artur Janowiec • 2 years ago

I got some different numbers in my calculations. For example: ((3/25)*(2/25)*(1/25)*(3/25)*(3/5)) = 0.0000276
and ((2/23)*(1/23)*(2/23)*(1/23)*(2/5)) = 0.00000571.
The conclusion that "sports" is the predicted label is still correct. Am I missing something or is this a calculation error that the author made?

Reply • Share ›

Text Analysis with Machine Learning

Subscribe

MonkeyLearn | Blog