Notes on Naive Bayes classifiers for spam filtering

Jonathan Lee

Consider the following problem involving Bayes' theorem:

40% of all emails are spam. 10% of spam emails contain the word "viagra", while only 0.5% of non-spam emails contain the word "viagra". What is the probability that an email is spam, given that it contains the word "viagra"?

Let S be the event that a given email is spam, and let V be the event that the email contains the word "viagra". We would like to know $\mathbb{P}(S|V)$, the probability that an email is spam given that it contains "viagra". A straightforward application of Bayes' theorem tells us that

$$\begin{aligned}\mathbb{P}(S|V) &= \frac{\mathbb{P}(V|S)\mathbb{P}(S)}{\mathbb{P}(V|S)\mathbb{P}(S) + \mathbb{P}(V|\neg S)\mathbb{P}(\neg S)} \\ &= \frac{0.1 \times 0.4}{0.1 \times 0.4 + 0.005 \times 0.6} \\ &\approx 0.93\end{aligned}$$

It's easy enough to derive this sort of probability when you're only examining one word. We would like to extend this sort of thinking to classify emails as either spam or not-spam (also known as "ham"), by examining every word in the email, building a very general, powerful spam filter along the way.

Naive Bayes in theory

Consider representing an email as the set of distinct words $(x_1, \ldots, x_n)$ contained in the text. We are interested in computing

$$\mathbb{P}(S|x_1, \ldots, x_n)$$

where S is the event that the email is spam, and H is the event that the email is ham (not-spam, equivalent to S^c). By Bayes' theorem, this is equivalent to

$$\frac{\mathbb{P}(x_1, \ldots, x_n|S)\mathbb{P}(S)}{\mathbb{P}(x_1, \ldots, x_n)}$$

$$= \frac{\mathbb{P}(x_1, \ldots, x_n|S)\mathbb{P}(S)}{\mathbb{P}(x_1, \ldots, x_n|S)\mathbb{P}(S) + \mathbb{P}(x_1, \ldots, x_n|H)\mathbb{P}(H)}$$

Ignoring the denominator for a minute, by definition of conditional probability,

$$\mathbb{P}(x_1, \ldots, x_n|S)\mathbb{P}(S) = \mathbb{P}(x_1, \ldots, x_n, S)$$

Using the chain rule to decompose,

$$\begin{aligned}\mathbb{P}(x_1, \ldots, x_n, S) &= \mathbb{P}(x_1|x_2, \ldots, x_n, S)\mathbb{P}(x_2, \ldots, x_n, S) &(1)\\ &= \mathbb{P}(x_1|x_2, \ldots, x_n, S)\mathbb{P}(x_2|x_3, \ldots, x_n, S)\mathbb{P}(x_3, \ldots, x_n, S) &(2)\\ &= \ldots &(3)\\ &= \mathbb{P}(x_1|x_2, \ldots, x_n, S)\mathbb{P}(x_2|x_3, \ldots, x_n, S) \ldots \mathbb{P}(x_{n-1}|x_n, S)\mathbb{P}(x_n|S)\mathbb{P}(S) &(4)\end{aligned}$$

This is a correct formula, but it's not a particularly useful one for us! The problem is that terms like $\mathbb{P}(x_1|x_2, \ldots, x_n, S)$ have so many conditions, that it's generally not possible to accurately calculate their probability.

Think about what that term is really trying to calculate: what's the probability that the word x_1 occurs in a spam email, given that $(x_2, \ldots x_n)$ also occur? Unless there is another spam email with all of those words $(x_2, \ldots x_n)$ in it that you've already looked at (which is unlikely unless you have absurd amounts of data), there's no way to know that probability already. We need to simplify the problem!

Naive Bayes "fixes" this by making the assumption that the words in an email are **conditionally independent of each other, given that we know whether or not the email is spam**. In other words, it assumes that knowing that an email has the word "viagra" in it, doesn't tell us anything about whether it has the word "pills" in it, **if we already factor in whether the email is spam or not**. This is why it's called Naive Bayes: it naively assumes independence where it might not actually exist.

This assumption, of course, is **not true** in reality: words often appear together, and English is not just a random walk through the dictionary. But it's a useful simplification of the problem that can lead to practical results.

Let's rewrite those chain rule probabilities using the conditional independence assumptions.

$$\begin{aligned}
\mathbb{P}(x_1, \ldots, x_n, S) &= \mathbb{P}(x_1|x_2, x_3, \ldots, x_n, S)\mathbb{P}(x_2|x_3, \ldots, x_n, S) \ldots \mathbb{P}(x_{n-1}|x_n, S)\mathbb{P}(x_n|S)\mathbb{P}(S) && (5)\\
&\approx \mathbb{P}(x_1|S)\mathbb{P}(x_2|S) \ldots \mathbb{P}(x_{n-1}|S)\mathbb{P}(x_n|S)\mathbb{P}(S) && (6)\\
&= \mathbb{P}(S)\prod_{i=1}^{n}\mathbb{P}(x_i|S) && (7)
\end{aligned}$$

By a similar argument,

$$\mathbb{P}(x_1, \ldots, x_n, H) \approx \mathbb{P}(H)\prod_{i=1}^{n}\mathbb{P}(x_i|H)$$

Putting it all together,

$$\mathbb{P}(S|x_1, \ldots, x_n) \approx \frac{\mathbb{P}(S)\prod_{i=1}^{n}\mathbb{P}(x_i|S)}{\mathbb{P}(S)\prod_{i=1}^{n}\mathbb{P}(x_i|S) + \mathbb{P}(H)\prod_{i=1}^{n}\mathbb{P}(x_i|H)}$$

Thus, we can calculate the probability of an email being spam if we know the prior probability of an email being spam, and the probabilities of seeing a particular word in either a spam or ham email!

How spammy is a word?

We have a way to compute the probability of an email being spam using relatively simple terms, but it's not yet clear how to compute those conditional probabilities. This classifier works by taking a large number of emails that have already been hand-labelled as *spam* or *ham*, and using that data to compute word spam probabilities, by counting the frequency of each word.

Imagine we're given a training set of 5000 randomly chosen emails. We examine them and label 2000 of them as spam emails and 3000 as ham emails (so $\mathbb{P}(S) = 0.4$, and $\mathbb{P}(H) = 0.6$). We'd like to calculate $\mathbb{P}(\text{"viagra"}|S)$ and $\mathbb{P}(\text{"viagra"}|H)$. The easiest thing to do would be to count how many spam emails have "viagra" and divide that by the total number of spam emails (and do the same thing for ham). So, if there were 216 spam emails with "viagra" and 12 ham emails with "viagra", we'd say

$$\mathbb{P}(\text{"viagra"}|S) = \frac{216}{2000} \approx 11\%$$

$$\mathbb{P}(\text{"viagra"}|H) = \frac{12}{3000} \approx 0.4\%$$

Smoothing

This is the right idea, but there's a small problem: what if there's a word (say, "Pokemon") that we've only ever seen before in ham emails, and not spam? In that case, $\mathbb{P}(\text{"Pokemon"}|S) = 0$, and the entire spam probability will go to zero, because we're multiplying all of the word probabilities together and we've never seen "Pokemon" in spam mail before. A malicious spammer, knowing this weakness, could just slip the word "Pokemon" in his pill-pushing email, and it would get right past our classifier. We would like to be robust to words we haven't seen before, or at least words we've only seen in one setting.

The solution is to never let any word probabilities be zero, by smoothing them upwards. **Instead of starting each word count at 0, start it at 1**. This way none of the counts will ever have a numerator of 0. This overestimates the word probability, so we need to **add 2 to the denominator**. (We add 2 because we're implicitly keeping track of 2 things: the number of emails that contain that word, and the number that don't. The sum of those two things should be in the denominator, and the 2 accounts for starting both the counters at 1.)

The smoothed word probabilities for the previous example are now

$$\mathbb{P}(\text{"viagra"}|S) = \frac{217}{2002} \approx 11\%$$

$$\mathbb{P}(\text{"viagra"}|H) = \frac{13}{3002} \approx 0.43\%$$

And our estimate for "Pokemon" in spam emails would now be $\frac{1}{2002}$, instead of 0.

This technique is called **Laplace smoothing** and was used in the 18th century to estimate the probability that the sun will rise tomorrow!

Naive Bayes algorithm

1. Iterate over the labelled spam emails, and for each word w in the entire training set, count how many of the spam emails contain w. Compute $\mathbb{P}(w|S) = \frac{|\text{spam emails containing w}|+1}{|\text{spam emails}|+2}$
2. Compute $\mathbb{P}(w|H)$ the same way for ham emails.
3. Compute $\mathbb{P}(S) = \frac{|\text{spam emails}|}{|\text{spam emails}|+|\text{ham emails}|}$
4. $\mathbb{P}(H) = \frac{|\text{ham emails}|}{|\text{spam emails}|+|\text{ham emails}|}$
5. Given a set of unlabelled test emails, iterate over each:
   (a) Create a set $\{x_1, \ldots, x_n\}$ of the distinct words in the email. Ignore the words that you haven't seen in the labelled training data.
   (b) Compute
   $$\mathbb{P}(S|x_1, \ldots, x_n) \approx \frac{\mathbb{P}(S)\prod_{i=1}^{n}\mathbb{P}(x_i|S)}{\mathbb{P}(S)\prod_{i=1}^{n}\mathbb{P}(x_i|S) + \mathbb{P}(H)\prod_{i=1}^{n}\mathbb{P}(x_i|H)}$$
   (c) If $\mathbb{P}(S|x_1, \ldots, x_n) > 0.5$, output "spam", else output "ham"

Avoiding floating point underflow

Multiplying a set of small probabilities together in step 5(b) above will probably result in floating point underflow, where the product will become too small to represent and will be replaced by 0. Instead of calculating

$$\mathbb{P}(S)\prod_{i=1}^{n}\mathbb{P}(x_i|S)$$

(which may underflow), consider computing the logarithm of this,

$$\log\left(\mathbb{P}(S)\prod_{i=1}^{n}\mathbb{P}(x_i|S)\right)$$

which can be written equivalently as

$$\log(\mathbb{P}(S))+\sum_{i=1}^{n}\log(\mathbb{P}(x_i|S))$$

Which will certainly not underflow. Then, realize that if

$$\log(\mathbb{P}(S))+\sum_{i=1}^{n}\log(\mathbb{P}(x_i|S))>\log(\mathbb{P}(H))+\sum_{i=1}^{n}\log(\mathbb{P}(x_i|H))$$

then, because $\log(x)>\log(y)$ iff $x>y$,

$$\mathbb{P}(S)\prod_{i=1}^{n}\mathbb{P}(x_i|S)>\mathbb{P}(H)\prod_{i=1}^{n}\mathbb{P}(x_i|H)$$

and therefore

$$\mathbb{P}(S|x_1,\ldots,x_n)>0.5$$

and thus the email should be classified as spam (and as ham otherwise).