

Introduction to Python for Data Science

Coding the Naive Bayes Algorithm

Dan Wiesenenthal
dan.wiesenenthal@galvanize.com

If you would like to follow along, head here:
<https://github.com/danwiesenenthal/Naive-Bayes-Evening-Workshop>

Naive Bayes

Naive Bayes Classifier: $\operatorname{argmax}_Y P(Y|\vec{X}) = \operatorname{argmax}_Y P(x_1|Y)P(x_2|Y)\dots P(x_3|Y)P(Y)$

$$= \operatorname{argmax}_Y P(Y) \prod_{i=1}^k P(x_i|Y)$$

Quick Probability

$P(a)$: The probability event a occurs

$P(a,b)$: The probability events a and b both occur

$P(a|b)$: The probability event a occurs, given that event b occurs

$$P(a,b) = P(b,a)$$

$$P(a,b) = P(a|b)P(b)$$

$$P(a|b) \neq P(b|a) \quad (\dots\text{why?})$$

Bayes Theorem

I want $P(a|b)$, but I don't have it already.

How could I get it?

$$P(a,b) = P(b,a)$$

$$P(a|b)P(b) = P(b|a)P(a)$$

$$P(a|b) = P(b|a)P(a) / P(b)$$

posterior = likelihood * prior / evidence

Intuitive?

We have a test that's marketed as 99% "accurate" (if you have the condition, you'll test positive 99% of the time; if you don't have the condition you'll test negative 99% of the time). Only 0.5% of the population have the condition (say, for example, lactose intolerance).

We pick Sally at random, and we test her. She tests positive. Does she have the condition?

Answer: Probably not!

It's only 33% likely. It's more likely that she doesn't have the condition than that she does!

$P(\text{cond} \mid \text{pos})$

$$= P(\text{pos} \mid \text{cond}) * P(\text{cond}) / P(\text{pos})$$

$$= P(\text{pos} \mid \text{cond}) * P(\text{cond}) / (P(\text{pos} \mid \text{cond}) * P(\text{cond}) + P(\text{pos} \mid \text{no-cond}) * P(\text{no-cond}))$$

$$= 0.99 * 0.005 / (0.99 * 0.005 + 0.01 * 0.995)$$

$$\approx .33$$

Naive Bayes:

Class | Feature

So for **one** feature:

$$P(\text{class} | \text{feature}) = P(\text{feature} | \text{class}) * P(\text{class}) / P(\text{feature})$$

Don't worry about the denominator for a minute, trust me...

$$= (\# \text{ feat in class} / \# \text{ in class}) * (\# \text{ class} / \# \text{ total data points})$$

For **multiple** features:

$$P(\text{feature} | \text{class}) * P(\text{class})$$

$$= P(c, F_1, F_2, \dots, F_n) \text{ ("joint")}$$

$$= P(c) * P(F_1 | c) * P(F_2 | c, F_1) * P(F_3 | c, F_1, F_2) * \dots$$

But if we assume independence of features given class... $P(F_i | c, F_j) = P(F_i | c)$

$$= P(c) * P(F_1 | c) * P(F_2 | c) * P(F_3 | c) * \dots$$

Which is way more tractable. Sweet!

Naive Bayes: $\text{argmax}(\text{Class})$

Now that we can calculate the numerator for **some** class, we calculate it for **every** class and compare them. The largest one is the most likely class, and therefore the class we should predict.

For each class **c**:

$$\text{pseudo_prob_data_is_in_this_class} = P(\mathbf{c}) * P(F1 | \mathbf{c}) * P(F2 | \mathbf{c}) * P(F3 | \mathbf{c}) * \dots$$

Prediction = the **c** which led to max pseudo_prob_data_is_in_this_class

The denominator stays the same for all classes, so we don't have to calculate it. Awesome.

“Training” is simply the process of counting up occurrences (all the little terms in “(# feat in class / # in class) * (# class / # total data points)”) so we can use them to predict. (In some models/algorithms, training is much, much more complicated.)

Thanks!

Dan Wiesenthal
dan.wiesenthal@galvanize.com