CS 252 - Naïve Bayes
Additional Reading 4
and
Homework problems

4 Naïve Bayes

In our treatment of probabilistic algorithms, we were able to make some strong statements about our confidence in their accuracy and the complexity of attaining that confidence by assuming a guaranteed bound on the error ϵ. In cases when we can obtain this guarantee, we can rely on such results. But what about situations in which we can not bound the error? Or situations in which there are more than two possible outcomes and in which only a plurality (rather than a majority) vote can be obtained? Or situations in which the answer is not deterministic (e.g. a string may be accepted sometimes and rejected others). In such situations, what we can guarantee about the correctness of answers is much more limited, but there still are algorithms that can be efficient and effective. One general class of such algorithms is the Bayesian network. In the context of language membership that has defined our study of algorithms, a Bayesian network is a model that tries to compute the most likely language membership value for a string w. That is, it tries to decide for a string w whether it is more likely in some language L or whether it is more likely not. While describing this general class of algorithms is beyond the scope of this reading, we will describe a special case called naïve Bayes.

To begin with, let's define a bit of notation. Let us consider a situation in which string w's membership may not be determined with certainty; instead we can consider it to have some probability of being in language L. Let $p(w)$ be the probability of encountering a string w (this is often called the *marginal*), $p(d)$ be the *prior* probability of decision d being the correct decision (about a string's membership in a language), $p(w|d)$ be the probability of encountering string w if decision d holds (often called the *likelihood*) and $p(d|w)$ be the *posterior* probability that decision d is correct. What we are interested in is this last probability. That is, given a string w, we'd like to know the probability that w is in L. But how do we compute this? Bayes' Law tells us that

$$p(d|w) = \frac{p(d)p(w|d)}{p(w)}$$

Does this help us any? Well, sort of. It is hard to know what $p(w)$ might be, but, if our goal is just to choose d such that $p(d|w)$ is highest, we can ignore this denominator and write

$$p(d|w) \propto p(d)p(w|d)$$

If we were given $p(d)$ and $p(w|d)$, we could then decide what the most likely value for d is given w. In the case of language membership, d can take only two values: Y or N. $p(d)$ is called the prior distribution because it is our best guess at the probability of a string being in L before we know what string we are talking about. Barring any information about L, we might decide to use a prior that weights both decisions equally (if we have some information about L that gives us a better guess at $p(d)$, we should of course make use of that). Now, suppose that $p(w|d)$ is somehow given to us. We can then compute $p(d|w)$ for any w as

$$\operatorname*{argmax}_d p(d)p(w|d)$$

4.0.1 Example

Let $\Sigma = \{0, 1\}$, $w \in \Sigma^*$, $n = |w|$ and k be the number of 1's in the w. Suppose $p(d)$ is given to us as

$p(d)$	Y	N
	0.3	0.7

and that $p(w|d)$ is given to us as

$p(w|d)$	Y	N
	$\frac{k}{m}$	0.25

What is the best answer for the question, "Is string $w = 11000$ in L?" It is the decision d that maximizes the posterior probability, $p(d|w)$; that is, it is

$$\operatorname*{argmax}_d p(d)p(11000|d)$$

So, since,

$$p(Y)p(11000|Y) = (0.3)(\frac{2}{5}) = 0.12$$

and

$$p(N)p(11000|N) = (0.7)(0.25) = 0.175$$

the most likely answer is $w \notin L$. On the other hand, for the string $w = 10111$,

$$p(Y)p(10111|Y) = (0.3)(\frac{4}{5}) = 0.24$$

and

$$p(N)p(10111|N) = (0.7)(0.25) = 0.175$$

and the most likely answer is $w \in L$.

4.1 Estimating the Likelihood

What if we do not know the prior and likelihood probabilities? We've already suggested that the prior might be estimated as equal probability for both decisions, if we don't have any other information to help us. But what about the likelihood? That conditional probability $p(w|d)$ is really the probability $p(a_1 a_2 \dots a_n|d)$, where a_1 is the value of the first symbol of w, a_2 is the value of the second, etc. That is, $p(a_1 a_2, \dots, a_n|d)$ is the *joint* conditional probability of $a_1 \ldots a_n$ (given d). Computing this joint probability, in general, is difficult because

$$p(a_1 a_2 \ldots a_n | d) = p(a_1|d)p(a_2|d, a_1)p(a_3|d, a_1 a_2) \ldots p(a_n|d, a_1 a_2 \ldots a_{n-1})^1$$

Unfortunately, estimating and computing these partial joint probabilities can be very difficult and very expensive, and many algorithmic approaches make use of simplifying assumptions to make the computation tractable. The most drastic of these assumptions is that the joint events are independent; that is, $\forall_{i,j,i \neq j} p(a_i|d) = p(a_i|d, a_j)$. In other words, the probability of value a_i occurring at position i in w (given d) is not affected by the presence (or absence) of value a_j at position j. Given this assumption, computing the joint conditional probability is significantly simplified, becoming

$$p(a_1 a_2 \ldots a_n | d) = p(a_1|d)p(a_2|d)p(a_3|d) \ldots p(a_n|d)$$

So, the likelihood has now been simplified to the product of relatively simple distributions of symbol values for different positions in the string w (given d). How might we estimate these? One good way is to do it empirically from examples. If we can get a hold of a set of words labeled as Y and N instances, we can count different values in different string positions for each type of word and use the counts to estimate the probabilities.

4.1.1 Example

Suppose we have been given two sets of strings,

$A = \{000, 001, 110, 101\}$
$B = \{111, 110, 011, 101\}$

The strings in A have been classified as yes-instances for a language L, and the strings in B as no-instances. We would like to build a naïve Bayes model from these examples and use it to decide if $100 \in L$. To make our decision, we must calculate

$$\underset{d}{\text{argmax}}\, p(d)p(w|d) = p(d)p(100|d) = p(d)p(a_1 = 1|d)p(a_2 = 0|d)p(a_3 = 0|d)$$

As our prior, we will let $p(Y) = p(N) = 0.5$.[2] We will estimate the following probabilities from the sets A and B:

$p(a_1 = 1|Y) = \frac{2}{4} = 0.5$
$p(a_2 = 0|Y) = \frac{3}{4} = 0.75$
$p(a_3 = 0|Y) = \frac{2}{4} = 0.5$
$p(a_1 = 1|N) = \frac{3}{4} = 0.75$
$p(a_2 = 0|N) = \frac{1}{4} = 0.25$
$p(a_3 = 0|N) = \frac{1}{4} = 0.25$

[1] It can be decomposed in many other ways as well.

[2] This seems like a good choice, given that $|A| = |B|$.

Then, computing the argmax,

$$p(Y)p(a_1 = 1|Y)p(a_2 = 0|Y)p(a_3 = 0|Y) = (0.5)(0.5)(0.75)(0.5) = 0.09375$$

$$p(N)p(a_1 = 1|N)p(a_2 = 0|N)p(a_3 = 0|N) = (0.5)(0.75)(0.25)(0.25) = 0.0234375$$

and we see that the most likely decision for the string 100 is that it is a yes-instance.

4.2 Generalization

Note that what we've done easily generalizes to any number of classifications and any number of alphabet symbols. Simply compute the needed conditional probabilities for each value d can take and then compute the argmax over all the possibilities.

4.3 Exercises

Exercise 4.1. You are given three sets of strings A, B and C, classified as type 0, type 1 and type 2 strings respectively. Build a naïve Bayes model from these data and use it to decide a type for the following three strings: $\{0000, 1001, 0011\}$.

$A = \{1011, 0111, 2011\}$
$B = \{0001, 0101, 1101, 2101\}$
$C = \{0010, 0100, 0110, 1000, 1010, 1100, 1110, 1111, 2000, 2001, 2010, 2100, 2110, 2111\}$.

For the prior use the following probabilities:

$p(d)$	*0*	*1*	*2*
	$\frac{3}{21}$	$\frac{4}{21}$	$\frac{14}{21}$

Estimate the naïve likelihood using the data, and for each of the following strings: $\{0000, 1001, 1011\}$

a. Compute the naïve Bayes probability that the string is a *type-0* instance.
b. Compute the naïve Bayes probability that the string is a *type-1* instance.
c. Compute the naïve Bayes probability that the string is a *type-2* instance.
d. Give the most likely output for the string (assuming independence).