March 20, 2017

Naive Bayes (NB) Classifiers

Douglas S. Rubin

1 The NB Assumption

The goal of any probability based classifier is to classify a new data point as belonging to class y (which is a member of a finite set) from its observed set of features, $x_1, x_2, ..., x_n$ by obtaining an estimate of $p(y|x_1, x_2, ..., x_n)$ based off of training data. Sometimes it is easier to model the distribution of features given each class $p(x_1, x_2, ..., x_n|y)$, and then invert this to obtain an estimate $p(y|x_1, x_2, ..., x_n)$ with Bayes rule:

$$p(y|x_1, x_2, ..., x_n) = \frac{p(x_1, x_2, ..., x_n|y)p(y)}{p(x_1, x_2, ..., x_n)}. \tag{1}$$

Thus, the prior distribution, $p(y)$ must be modeled as well. Often, however, it is difficult to model $p(x_1, x_2, ..., x_n|y)$ exactly. This is because, from the chain rule $p(x_1, x_2, ..., x_n|y) = p(x_1|y)p(x_2|y, x_1)...p(x_n|y, x_{n-1}, ..., x_1)$, and it is usually unclear how to model x_j conditioned on y as well as the previous set of features. The complexity of the problem reduces significantly if we use *Naive Bayes assumption*, which is to assume that the probabilities that $X_1 = x_1$, $X_2 = x_2$,... are conditionally independent given y:

$$p(x_1, x_2, ..., x_n|y) = \prod_{j=1}^{n} p(x_j|y). \tag{2}$$

Even though this greatly simplifies the problem, in some situations the NB assumption clearly does not hold. For example, consider spam/not-spam emails. If $y \in \text{spam}, \text{not} - \text{spam}$ and the x_js indicate the presence of a word in the email, given that $y = \text{not} - \text{spam}$ and that the email contains the word "Alice" (the name of a co-worker) it is much more likely that the word "Bob" (another co-worker) is also in the email. However, in practice this assumption usually works quite well.

1.1 A NB classifier for binarized features

In this section, I explain how one can construct a spam/non-spam classifier by considering the words contained in an email. Let $d = \{\text{a}, \text{running}, \text{can}, \text{insurance}, ...\}$ be a dictionary of size n all possible words that could be contained in an email (in practice this is usually the set of words, not including un-informative "stop" words in your training email set). Let's model the process by which emails are sent by the following sequence of events. First, someone has an unfair spam/non-spam coin with probability ϕ of landing heads. The person flips the coin, and if it lands heads they go to a box containing n "spam coins". Each coin corresponds to word, j, in the dictionary and has probability $\phi_{j|1}$ of landing heads (with coins representing very "spammy" words having a high probability of landing heads). The person flips each coin, and if the coin lands heads, the person writes that word in the email. After

going through all of the spam coins, the person then sends off the spam email. Conversely, if the original coin flip landed tails, the person goes to a box containing n "non-spam" coins for all the words in the dictionary, each with probability $\phi_{j|0}$ of landing heads (with coins representing very "non-spammy" words having a high probability of landing heads). The person performs the same procedure to write the email and then sends off the non-spam email.

Given a dataset of spam and non-spam emails, our goal is to estimate ϕ to estimate the probability of spam $p(y)$, and $\phi_{j|0}$ and $\phi_{j|1}$ to estimate the probability of word j conditioned on the email being spam or non-spam $p(x_j|y)$. Given these estimated prior probabilities of the distribution of y and the conditional distribution of the features, x_j, we can then implement Bayes rule to obtain an estimate of the probability of being spam/non-spam given the observed features $p(y|x_1, x_2, ..., x_n)$.

Let's first model the prior probability that the email is spam. Since the probability that the email is spam/non-spam is simply given by a coin flip, the proper distribution of $p(y)$ to use is a Bernoulli. For a particular email, let

$$\begin{cases} p(y=1) = \phi \\ p(y=0) = 1-\phi, \end{cases} \tag{3}$$

where ϕ is the probability of spam, and $y \in 0, 1$ where 0 denotes non-spam and 1 denotes spam. Putting these equations together we arrive at the prior $p(y)$:

$$p(y;\phi) = \phi^y(1-\phi)^{1-y}. \tag{4}$$

Now to model probabilities of the features conditioned on spam/non-spam, for a particular email, let $x_{\text{running}} = 1$ denote that the word "running" is in the email (for ease we take $x_{\text{running}} = 1$ even if the word "running" appears more than once in the email) and $x_{\text{running}} = 0$ denote that it does not. Since whether or not the word appears in the email conditioned on spam/non-spam is again a coin flip, the proper distribution to use is a Bernoulli defined by the following probabilities:

$$\begin{cases} p(x_j = 1|y=1) = \phi_{j|1} \\ p(x_j = 0|y=1) = 1-\phi_{j|1} \\ p(x_j = 1|y=0) = \phi_{j|0} \\ p(x_j = 0|y=0) = 1-\phi_{j|0}. \end{cases} \tag{5}$$

Putting these together in a single equation, we arrive at $p(x_j|y)$:

$$p(x_j|y;\phi_{j|1},\phi_{j|0}) = \phi_{j|1}^{x_j \wedge y}(1-\phi_{j|1})^{\neg x_j \wedge y}\phi_{j|0}^{x_j \wedge \neg y}(1-\phi_{j|0})^{\neg x_j \wedge \neg y}, \tag{6}$$

where standard Boolean/logic notation has been used ($1 \wedge 1 = 1$, $0 \wedge 1 = 0$, $1 \wedge 0 = 0$, $0 \wedge 0 = 0$). Thus, the joint probability of receiving a single spam/non-spam email with or without words $1, 2, ..., n$ is $p(x_1, x_2, ..., x_n, y; \phi, \vec{\phi}) = p(x_1, x_2, ..., x_n, |y; \vec{\phi})p(y;\phi)$, where $\vec{\phi} \equiv (\phi_{1|1}, \phi_{2|1}, ..., \phi_{n|1}, \phi_{1|0}, \phi_{2|0}, ..., \phi_{n|0})$. For our entire training set of emails, we can thus write down the joint likelihood function, and maximize it to obtain our best estimates of ϕ and $\vec{\phi}$:

$$\begin{aligned}
\mathcal{L}(\phi, \vec{\phi}) &= \prod_{i=1}^{m} p(x_1^{(i)}, x_2^{(i)}, ..., x_n^{(i)}, y^{(i)}; \phi, \vec{\phi}) \\
&= \prod_{i=1}^{m} p(x_1^{(i)}, x_2^{(i)}, ..., x_n^{(i)} | y^{(i)}; \vec{\phi}) p(y^{(i)}; \phi) \\
&= \prod_{i=1}^{m} p(y^{(i)}; \phi) \prod_{j=1}^{n} p(x_j^{(i)} | y^{(i)}; \vec{\phi}), \qquad (7)
\end{aligned}$$

where the superscript (i) denotes a single training example, and m is the number of emails in our training set. In going from the 2nd to 3rd line, we have employed the NB assumption. We now find the log likelihood, which will be easier to maximize:

$$\begin{aligned}
\ell(\phi, \vec{\phi}) &= \log \prod_{i=1}^{m} p(y^{(i)}; \phi) \prod_{j=1}^{n} p(x_j^{(i)} | y^{(i)}; \vec{\phi}) \\
&= \sum_{i=1}^{m} \log p(y^{(i)}; \phi) + \sum_{i=1}^{m} \log \prod_{j=1}^{n} p(x_j^{(i)} | y^{(i)}; \vec{\phi}) \\
&= \sum_{i=1}^{m} \log p(y^{(i)}; \phi) + \sum_{i=1}^{m} \sum_{j=1}^{n} \log p(x_j^{(i)} | y^{(i)}; \vec{\phi}), \qquad (8)
\end{aligned}$$

and plugging in our models for the probability distributions we arrive at

$$\begin{aligned}
\ell(\phi, \vec{\phi}) &= \sum_{i=1}^{m} y^{(i)} \log(\phi) + (1 - y^{(i)}) \log(1 - \phi) \\
&+ \sum_{i=1}^{m} \sum_{j=1}^{n} (x_j^{(i)} \wedge y^{(i)}) \log \phi_{j|1} + (\neg x_j^{(i)} \wedge y^{(i)}) \log(1 - \phi_{j|1}) + (x_j^{(i)} \wedge \neg y^{(i)}) \log \phi_{j|0} \\
&+ (\neg x_j^{(i)} \wedge \neg y^{(i)}) \log(1 - \phi_{j|0}). \qquad (9)
\end{aligned}$$

To obtain our best estimate values of ϕ, $\phi_{j|1}$ and $\phi_{j|0}$, we maximize the likelihood function by taking derivatives and setting the corresponding equations to zero. For ϕ,

$$\frac{\partial \ell(\phi, \vec{\phi})}{\partial \phi} = \sum_{i=1}^{m} \frac{y^{(i)}}{\phi} - \frac{1 - y^{(i)}}{1 - \phi}, \qquad (10)$$

which, when set to 0 and solved for ϕ reduces to our best estimate for ϕ:

$$\hat{\phi} = \frac{1}{m} \sum_{i=1}^{m} y^{(i)}. \qquad (11)$$

One can do the same for the $\phi_{k|1}$ (I've switched from j to k since I am using j as a dummy variable in the sum). Notice how much easier taking the derivative is by taking the log of the likelihood function. For $\phi_{k|1}$, the sum over j gets killed except when $j = k$:

$$\frac{\partial \ell(\phi, \vec{\phi})}{\partial \phi_{k|1}} = \sum_{i=1}^{m} \frac{x_k^{(i)} \wedge y^{(i)}}{\phi_{k|1}} - \frac{\neg x_k^{(i)} \wedge y^{(i)}}{1 - \phi_{k|1}}, \tag{12}$$

and solving for $\phi_{k|1}$ we obtain our best estimate for $\phi_{k|1}$:

$$\hat{\phi}_{k|1} = \frac{\sum_{i=1}^{m} x_k^{(i)} \wedge y^{(i)}}{\sum_{i=1}^{m} x_k^{(i)} \wedge y^{(i)} + \sum_{i=1}^{m} \neg x_k^{(i)} \wedge y^{(i)}} = \frac{\sum_{i=1}^{m} x_k^{(i)} \wedge y^{(i)}}{\sum_{i=1}^{m} y^{(i)}} \tag{13}$$

The best estimate for $\phi_{k|0}$ can be found similarly:

$$\hat{\phi}_{k|0} = \frac{\sum_{i=1}^{m} x_k^{(i)} \wedge \neg y^{(i)}}{\sum_{i=1}^{m} x_k^{(i)} \wedge \neg y^{(i)} + \sum_{i=1}^{m} \neg x_k^{(i)} \wedge \neg y^{(i)}} = \frac{\sum_{i=1}^{m} x_k^{(i)} \wedge \neg y^{(i)}}{\sum_{i=1}^{m} \neg y^{(i)}}. \tag{14}$$

The formulas for the best estimates for $\hat{\phi}$, $\hat{\phi}_{k|1}$ and $\hat{\phi}_{k|0}$ make a lot of intuitive sense. The formula for $\hat{\phi}$, which represents the prior probability of spam is just the fraction of emails in the training set which are spam. The formula for $\hat{\phi}_{k|1}$, which represents the probability of word k appearing in an email given that it is spam is just the fraction of times that word k appears in all spam emails. Similarly the formula for $\hat{\phi}_{k|0}$ is just the fraction of times that word k appears in all non-spam emails

To predict whether a new email, with feature values $(x_1, x_2, ..., x_n)$ is spam or not spam, we use our best estimate probabilities in Bayes rule to obtain $p(y|x_1, x_2, ..., x_n)$:

$$p(y|x_1, x_2, ..., x_n; \hat{\phi}, \hat{\vec{\phi}}) = \frac{p(x_1, x_2, ..., x_n|y; \hat{\vec{\phi}})p(y; \hat{\phi})}{p(x_1, x_2, ..., x_n; \hat{\vec{\phi}})}. \tag{15}$$

The estimated class of the new email is thus:

$$\hat{y} = \text{argmax}_{y \in \{0,1\}} \left\{ \frac{p(x_1, x_2, ..., x_n|y; \hat{\vec{\phi}})p(y; \hat{\phi})}{p(x_1, x_2, ..., x_n; \hat{\vec{\phi}})} \right\} = \text{argmax}_{y \in \{0,1\}} \left\{ p(x_1, x_2, ..., x_n|y; \hat{\vec{\phi}})p(y; \hat{\phi}) \right\}, \tag{16}$$

and plugging in our Bernoulli formulas for the prior probabilities and using the NB assumption, we arrive at the classification:

$$\hat{y} = \text{argmax}_{y \in \{0,1\}} \left\{ \hat{\phi}^y (1 - \hat{\phi})^{1-y} \prod_{j=1}^{n} \hat{\phi}_{j|1}^{x_j \wedge y} (1 - \hat{\phi}_{j|1})^{\neg x_j \wedge y} \hat{\phi}_{j|0}^{x_j \wedge \neg y} (1 - \hat{\phi}_{j|0})^{\neg x_j \wedge \neg y} \right\}. \tag{17}$$

1.2 The Multi-variate Bernoulli event model

We now take a slightly different approach to model the process by which a spam/non-spam email is sent. The beginning of the process is still the same, a person has an unfair spam/non-spam coin with probability ϕ of landing heads and flips the coin. But if the coin lands heads,

they now go to a multi-faceted "spam" die with as many faces as there are words in the dictionary. The spam die is unfair, with faces with very "spammy" words weighted most heavily, with the probability of word j landing face up given by $\phi_{j|1}$. If the email contains N words, the person flips the n sided die N times and writes the corresponding words in an email and sends off the spam email. Conversely, if the original coin toss lands tails then the person flips an unfair multi-faceted "non-spam" die, with very "non-spammy" words weighted most heavily. The probability of word j landing face up for this die is $\phi_{j|0}$.

To mathematically model this process, the probability distribution of the first process, the coin flip is still given by a Bernoulli(ϕ) distribution. However, for the second stage of rolling either a spam or non-spam die, this corresponds to a multinomial distribution. Let x_j now correspond to the jth word in the email, and let $p(x_j = p|y)$ correspond to the probability that the jth word in the email takes on the value pth value from the dictionary $p \in \{1, 2, ..., n\}$. Therefore, for word j of an email:

$$\begin{cases} p(x_j = 1|y = 1) = \phi_{1|1} \\ p(x_j = 2|y = 1) = \phi_{2|1} \\ ... \\ p(x_j = n|y = 1) = 1 - \sum_{\lambda=1}^{n-1} \phi_{\lambda|1} \\ p(x_j = 1|y = 0) = \phi_{1|0} \\ p(x_j = 2|y = 0) = \phi_{2|0} \\ ... \\ p(x_j = n|y = 0) = 1 - \sum_{\lambda=1}^{n-1} \phi_{\lambda|0}. \end{cases} \tag{18}$$

Putting all of this together, we get the multi-nomial distribution for the conditional probability that word j in the email takes on one of the n words in the dictionary:

$$p(x_j|y; \vec{\phi}) = \phi_{n|1}^{\mathbb{1}(x_j=n \wedge y=1)} \phi_{n|0}^{\mathbb{1}(x_j=n \wedge y=0)} \prod_{\gamma=1}^{n-1} \phi_{\gamma|1}^{\mathbb{1}(x_j=\gamma \wedge y=1)} \phi_{\gamma|0}^{\mathbb{1}(x_j=\gamma \wedge y=0)}, \tag{19}$$

where $\mathbb{I}$ is the indicator function $\mathbb{I}(\text{True}) = 1$ and $\mathbb{I}(\text{False}) = 0$. Also, for ease of notation, I have defined $\phi_{n|1} \equiv 1 - \sum_{\lambda=1}^{n-1} \phi_{\lambda|1}$ and $\phi_{n|0} \equiv 1 - \sum_{\lambda=1}^{n-1} \phi_{\lambda|0}$. I have also re-defined $\vec{\phi}$ as: $\vec{\phi} \equiv (\phi_{1|1}, \phi_{2|1}, ..., \phi_{n-1|1}, \phi_{1|0}, \phi_{2|0}, ..., \phi_{n-1|0})$.

Thus, since we are assuming conditional independence of the die rolls (the NB assumption), given y, for a particular email, the total probability of observing x_1, x_2,... x_N conditioned on y is:

$$p(x_1, x_2, ..., x_N|y; \vec{\phi}) = \prod_{j=1}^{N} \phi_{n|1}^{\mathbb{1}(x_j=n \wedge y=1)} \phi_{n|0}^{\mathbb{1}(x_j=n \wedge y=0)} \prod_{\gamma=1}^{n-1} \phi_{\gamma|1}^{\mathbb{1}(x_j=\gamma \wedge y=1)} \phi_{\gamma|0}^{\mathbb{1}(x_j=\gamma \wedge y=0)}, \tag{20}$$

The joint likelihood function is now

$$\begin{aligned}
\mathcal{L}(\phi, \vec{\phi}) &= \prod_{i=1}^{m} p(x_1^{(i)}, x_2^{(i)}, ..., x_{N_i}^{(i)}, y^{(i)}; \phi, \vec{\phi}) \\
&= \prod_{i=1}^{m} p(x_1^{(i)}, x_2^{(i)}, ..., x_{N_i}^{(i)} | y^{(i)}; \vec{\phi}) p(y^{(i)}; \phi) \\
&= \prod_{i=1}^{m} p(y^{(i)}; \phi) \prod_{j=1}^{N_i} p(x_j^{(i)} | y^{(i)}; \vec{\phi}),
\end{aligned} \tag{21}$$

which is exactly the same as the previous likelihood function, except that n has been replaced by N_i. There the log-likelihood is give as before:

$$\begin{aligned}
\ell(\phi, \vec{\phi}) &= \sum_{i=1}^{m} \log p(y^{(i)}; \phi) + \sum_{i=1}^{m} \sum_{j=1}^{N_i} \log p(x_j^{(i)} | y^{(i)}; \vec{\phi}) \\
&= \sum_{i=1}^{m} y^{(i)} \log(\phi) + (1 - y^{(i)}) \log(1 - \phi) \\
&+ \sum_{i=1}^{m} \sum_{j=1}^{N_i} \mathbb{1}(x_j^{(i)} = n \wedge y^{(i)} = 1) \log \phi_{n|1} + \mathbb{1}(x_j^{(i)} = n \wedge y^{(i)} = 0) \log \phi_{n|0} \\
&+ \sum_{i=1}^{m} \sum_{j=1}^{N_i} \sum_{\gamma=1}^{n-1} \mathbb{1}(x_j^{(i)} = \gamma \wedge y^{(i)} = 1) \log \phi_{\gamma|1} + \mathbb{1}(x_j^{(i)} = \gamma \wedge y^{(i)} = 0) \phi_{\gamma|0}
\end{aligned} \tag{22}$$

Maximizing over ϕ by taking $\partial / \partial \phi$, we get the same solution as before:

$$\hat{\phi} = \frac{1}{m} \sum_{i=1}^{m} y^{(i)}. \tag{23}$$

1.3 Other NB classifiers

1.4 Laplace Smoothing