CSE 446: Machine Learning **Winter 2020**

Section 6: Naive Bayes

Lecturer: Byron Boots *Date 27 Jan 2020*

Agenda:

1. Announcements
2. Review Naive Bayes classifier.

Naive Bayes Classifier

Naive Bayes is a probabilistic machine learning algorithm that can be used in a wide variety of classification tasks. Typical applications include filtering spam e-mails, classifying documents, etc. Today, we will look at Naive Bayes classifiers in the context of spam classification for e-mails. Let us represent an e-mail, or any document we wish to classify as either spam or not spam, as the set $\{x_1, \ldots, x_n\}$ of distinct words in the text. That is, we desire to compute

$$P(S \mid x_1 \ldots x_n).$$

We write S to denote the event that our document is spam and $\overline{S}$ to denote the event that our document is not spam. Then, by Bayes' theorem we have

$$P(S \mid x_1 \ldots x_n) = \frac{P(x_1 \ldots x_n \mid S)P(S)}{P(x_1 \ldots x_n)} = \frac{P(x_1 \ldots x_n \mid S)P(S)}{P(x_1 \ldots x_n \mid S)P(S) + P(x_1 \ldots x_n \mid \overline{S})P(\overline{S})}.$$

By the chain rule of probability, we can decompose the numerator as

$$P(x_1 \ldots x_n \mid S)P(S) = P(x_1 \mid S)P(x_2 \mid x_1, S) \ldots P(x_n \mid x_1 \ldots x_{n-1}, S)P(S).$$

However, this expression can be difficult to calculate. We simplify the problem by making the assumption that each of the words are conditionally independent of each other if we already know whether the document is spam or not spam. While this assumption is not true, it does provide a useful relaxation to our problem. Hence,

$$P(x_1 \ldots x_n \mid S)P(S) = P(S)\prod_{i=1}^{n} P(x_i \mid S)$$
$$P(x_1 \ldots x_n \mid \overline{S})P(\overline{S}) = P(\overline{S})\prod_{i=1}^{n} P(x_i \mid \overline{S})$$

Returning to the expression we originally derived from Bayes' Rule, we have

$$P(S \mid x_1 \ldots x_n) = \frac{P(S)\prod_{i=1}^{n} P(x_i \mid S)}{P(S)\prod_{i=1}^{n} P(x_i \mid S) + P(\overline{S})\prod_{i=1}^{n} P(x_i \mid \overline{S})}.$$

Laplace Smoothing

If a given token and class never occur together in the training data, then the probability estimate for that token conditioned on that class will be zero. Recall, the probability estimate is directly proportional to the number of times a given token occurs. Discuss why this scenario is problematic for our classifier and how an adversarial agent (spam generator) might take advantage of it.

One solution is to smooth all our word probabilities upwards by a count of 1. That is, we assume we saw each token in each token for each class once more than we actually did. Specifically, our probability for some token u goes from

$$P(u \mid S) = \frac{\text{number of spam e-mails containing } u}{\text{number of spam e-mails}}$$

to a smoothed version of

$$P(u \mid S) = \frac{1 + \text{ number of spam e-mails containing } u}{2 + \text{ number of spam e-mails}}.$$

Similarily,

$$P(u \mid \overline{S}) = \frac{1 + \text{ number of 'not spam' e-mails containing } u}{2 + \text{ number of 'not spam' e-mails}}.$$

The reason we add 2 to the denominator when we smooth is because any token can take on either take on the value of belonging to 'spam' or 'not spam'.

Algorithm

Let V be the set of unique tokens in the training set.

Algorithm 1: Naive Bayes Classifier

```
for $\forall u \in V$ do
    Compute and store $P(u \mid S) = \frac{1 + \text{ number of 'spam' e-mails containing } u}{2 + \text{number of 'spam' e-mails}}$
    Compute and store $P(u \mid \overline{S}) = \frac{1 + \text{ number of 'not spam' e-mails containing } u}{2 + \text{number of ' not spam' e-mails}}$
end
Compute $P(S) = \frac{|\text{'spam emails'}|}{|\text{'spam' emails}| + |\text{'not spam' emails}|}$ and $P(\overline{S}) = 1 - P(S)$
for each $email \in test\ set$ do
    Create set $\{x_1 \dots x_n\}$ of distinct words in e-mail, ignoring words not seen in labeled training data.
    if $P(S)\prod_{i=1}^{n} P(x_i \mid S) > P(\overline{S})\prod_{i=1}^{n} P(x_i \mid \overline{S})$ then
        Classify e-mail as 'spam'.
    else
        Classify e-mail as 'not spam'
    end
end
```

Although we are nearly done, there is one issue of practical concern. Namely, calculating either of the expressions in the above inequality is likely to cause numerical underflow issues. As you might recall from lecture, we solve this issue by computing the inequality in the log-space. For two values a, b, we have

$$a > b \iff \log(a) > \log(b).$$

So,

$$P(S)\prod_{i=1}^{n} P(x_i \mid S) > P(\overline{S})\prod_{i=1}^{n} P(x_i \mid \overline{S}) \iff \log(P(S)) + \sum_{i=1}^{n} \log(P(x_i \mid S)) > \log(P(\overline{S})) + \sum_{i=1}^{n} \log(P(x_i \mid \overline{S})).$$

Now, we simply modify our algorithm by switching the product-based inequality for the log-space inequality.