210709 Machine Learning **Spring 2021**

Lecture 5. Naive Bayes Classifier

Lecturer: Jie Wang Date: April 16, 2021

The major reference of this lecture is [1].

1 Spam Detection

Suppose that you are a software engineer working for Google. You boss asks you to develop an algorithm that can automatically detect spam emails. What are you supposed to do?

The first step is to collect a banch of data (the more, the better) $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where each $\mathbf{x}_i$ is an email consisting a few words and tokens, and $y_i \in \mathcal{C} = \{\text{spam}, \text{not spam}\}$.

This is a typical *classification* problem, as the *target space* is a finite set. In many real applications, the labels have *no order*, but occasionally, they do.

2 Naive Bayes Classifier

We can model this problem by the conditional probability:

$$\begin{aligned}\hat{y} &= \mathop{\mathbf{argmax}}_{c\in\mathcal{C}} P(c|X) \\ &= \mathop{\mathbf{argmax}}_{c\in\mathcal{C}} \frac{P(X|c)P(c)}{P(X)} \qquad \leftarrow \textbf{Bayes Theorem} \\ &= \mathop{\mathbf{argmax}}_{c\in\mathcal{C}} P(X|c)P(c) \\ &= \mathop{\mathbf{argmax}}_{c\in\mathcal{C}} P(X_1,\ldots,X_d|c)P(c). \end{aligned} \tag{1}$$

The term $P(c)$ is the so-called ***prior probability*** of the class c.

Remark 1. For the spam email detection problem, emails have different number of words and tokens, which implies that the dimensions of the input instances can be different. We will see that this will not cause any trouble to the naive Bayes classifier.

To apply (1) to predict the label of a new email, we need to estimate the ***class priors*** $P(c)$ for all $c \in \mathcal{C}$, and $P(X_1,\ldots,X_d|c)$ for all possible attributes' values and class labels. Estimating the class priors $P(c = \textit{spam})$ and $P(c = \textit{not spam})$ from the training set is easy, as $|\mathcal{C}|$—that is, the number of labels in $\mathcal{C}$—is small in real applications. For example,

$$P(c = \textit{spam}) = \frac{\text{the number of spam emails in the training set}}{\text{the number of emails in the training set}}.$$

However, reliable estimations of conditional joint probabilities require a huge amount of training samples. Let $|X_i|$ be the number of possible values of the i^{th} attribute. The number of probabilities we need to estimate is

$$|\mathcal{C}| + |\mathcal{C}|\prod_{i=1}^{d}|X_i|. \tag{2}$$

Notice that, there are approximately 50000 distinct English words in vocabulary, and an email often contains more than 100 words and tokens. This implies that the number of probabilities we need to estimate is astronomical.

To put the above naive Bayes classifier to practical use, we need a few assumptions.

Assumption 1. *The attributes X_i, $i = 1, \ldots, d$, are independent conditioned on the label:*

$$P(X_1, \ldots, X_d|c) = \prod_i P(X_i|c).$$

Assumption 1 transfers the above model to

$$\hat{y} = \underset{c \in \mathcal{C}}{\mathbf{argmax}}\, P(c) \prod_i P(X_i|c).$$

Question 1. How many parameters do we need to estimate under Assumption 1?

Before we answer this question, let us first take a look at a simple example. Consider a short email which says "*laptops with the lowest price*". The corresponding representation is thus $\mathbf{x} = \{laptops, with, the, lowest, price\}$. To predict its label by the above model, we need to compute

$$\hat{y} = \underset{c \in \{spam,\, not\, spam\}}{\mathbf{argmax}}\, P(c)P(X_1 = laptops|c) \cdots P(X_5 = price|c).$$

Then, besides of the class priors, we need to estimate a set of conditional probabilities in the form of $P(X_j = w_k|c)$, where w_k is the k^{th} word in the vocabulary $\mathcal{V}$. Thus, the number of probabilities we need to estimate is

$$|\mathcal{C}| + |\mathcal{C}| \times |\mathbf{x}| \times |\mathcal{V}|,$$

which is considerably smaller than the number computed by Eq. (2), while still quite large in practice ($|\mathbf{x}| = 5$ in this case). To further compress the number of probabilities we need to estimate, we introduce the second assumption as follows.

Assumption 2. *We assume that the attributes are independent and identically distributed (regarding to the word positions) given their target classification, that is*

$$P(X_i = w_k|c) = P(X_j = w_k|c),\ \forall\, i, j, k, c.$$

Assumption 2 implies that we can estimate the *position-independent* probabilities $P(w_k|c)$ instead of the set of probabilities $P(X_1 = w_k|c), P(X_2 = w_k|c), \ldots$. Thus, the number of probabilities we need to estimate reduces to

$$|\mathcal{C}| + |\mathcal{C}| \times |\mathcal{V}|.$$

For example, the position-independent probabilities $P(w_k|c = spam)$ can be estimated by

$$P(w_k|c = spam) = \frac{\text{the number of times the word } w_k \text{ appears in the spam emails}}{\text{the total number of words in the spam emails}}.$$

To simplify notations, let n_c be the total number of words in training examples whose target value is c, that is

$$n_c = \sum_{\{i: y_i = c\}} |\mathbf{x}_i|,$$

and $n_{c,k}$ be the number of times the word w_k is found among these n_c word positions. Thus

$$P(w_k|c) = \frac{n_{c,k}}{n_c}.$$

However, the above estimation may be problematic when the word w_k does not appear in the training samples whose target value is c. This implies that the corresponding $P(w_k|c)$ is 0. As a result, as long as the word w_k appears in an email, its target value can not be c. To fix this problem, we apply the Laplace smoothing technique

$$P(w_k|c) = \frac{n_{c,k} + 1}{n_c + |\mathcal{V}|}. \tag{3}$$

3 Naive Bayes in Psedocode

We summarize the training and testing of naive bayes classifier as follows.

Algorithm 1 Training Naive Bayes Classifier

```
Input: The training set of emails with the corresponding labels $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$
 1: $\mathcal{V} \leftarrow$ the set of distinct words and other tokens found in $\mathcal{D}$;    % build the vocabulary
 2: for each target value $c$ in the labels set $\mathcal{C}$ do                  % estimate the probabilities
 3:     $\mathcal{D}_c \leftarrow$ the training samples whose labels are $c$
 4:     $P(c) \leftarrow \frac{|\mathcal{D}_c|}{|\mathcal{D}|} = \frac{\text{the number of samples in } \mathcal{D}_c}{\text{the number of samples in } \mathcal{D}}$
 5:     $T_c \leftarrow$ a single document by concatenating all training samples in $\mathcal{D}_c$
 6:     $n_c \leftarrow |T_c| =$ the number of word positions in $T_c$
 7:     for each word $w_k$ in the vocabulary $\mathcal{V}$ do
 8:         $n_{c,k} \leftarrow$ the number of times the word $w_k$ occurs in $T_c$
 9:         $P(w_k|c) = \frac{n_{c,k}+1}{n_c+|\mathcal{V}|}$
10:     end for
11: end for
```

Algorithm 2 Testing Naive Bayes Classifier

```
Input: A new email $\mathbf{x}$. Let $x_i$ be the $i^{th}$ token in $\mathbf{x}$. $\mathcal{I} = \emptyset$.
 1: for $i = 1, \ldots, |\mathbf{x}|$ do
 2:     if $\exists\, w_{k_i} \in \mathcal{V}$ such that $w_{k_i} = x_i$ then
 3:         $\mathcal{I} \leftarrow \mathcal{I} \cup k_i$
 4:     end if
 5: end for
 6: predict the label of $\mathbf{x}$ by
        $$\hat{y} = \mathop{\mathbf{argmax}}_{c \in \mathcal{C}} P(c) \prod_{i \in \mathcal{I}} P(w_{k_i}|c)$$
```

Algorithm 1 aims to complete two major tasks. The first one is to build a vocabulary, which contains all distinct words and tokens ***in the training samples***. The second one is to compute the class priors and the conditional probabilities. Notice that, as the vocabulary is built upon the training samples, we may find words or tokens in the testing samples that are out of the vocabulary. If this happens, Algorithm 2 will simply ignore these words or tokens that are out of the vocabulary.

Question 2. Notice that, for the spam email detection problem, emails have different number of words and tokens. This implies that, different data instances $\mathbf{x}_i$ may have different dimensions, i.e., $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,n_i})$. Is the naive Bayes classifier still applicable?

4 Measures of Classifiers' Performance

For a classification task, we may have multiple classifiers. How to select the best classifier? We need some criteria to measure the performance of these classifiers.

We consider the binary classification problems. A natural idea may lead to the so-called *accuracy*.

$$\text{Accuracy} = \frac{\text{no. of correct predictions}}{\text{no. of data instances}}.$$

However, accuracy can not properly measure the performance of classifiers for imbalanced data. For example, for a data set where only less than 1% data instances are positive samples, a classifier that always assigns negative labels to data instances has an accuracy larger than 99%.

To address this problem, we develop several other measures that are more useful. There are four commonly used measurements, that are, *true positives, true negatives, false positives, and false negatives.* The terms *positives and negatives* refer to the prediction by the classifier. The terms *true and false* refer to whether the predictions are consistent with the true labels. Commonly used measures of the classifiers' performance are as follows.

$$\begin{aligned}
\text{Precision} &= \frac{\text{tp}}{\text{tp} + \text{fp}}, \\
\text{Recall} &= \frac{\text{tp}}{\text{tp} + \text{fn}}, \\
\text{F} - \text{score} &= \frac{2}{\frac{1}{\text{Precision}} + \frac{1}{\text{Recall}}}.
\end{aligned}$$

References

[1] T. M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.