# In-class Activity 8: Implementation of Naive Bayes Classifier

Due by March 18th (Monday) 11:59 pm

March 14, 2019

1. **Background:** In machine learning, Naive Bayes Classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features. In essence, NBC computes the posterior (probability of observations given each potential class) and make classification based on the likelihood (and chooses the class label for which posterior is the largest). In plain English, using Bayesian probability terminology, we can describe the posterior in:

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}$$

Suppose we have $K$ classes and one of them can be represented as $C_k$, and we have an observation $\mathbf{x} = (x_1, x_2, \cdots, x_n)$ that has $n$ features. Then, the above expression can be written as:

$$p(C_k|\mathbf{x}) = \frac{p(C_k)p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$

where $p(C_k)$ is the prior probability of class $C_k$, $p(\mathbf{x}|C_k)$ is the likelihood of observe $\mathbf{x}$ given class $C_k$, and the evidence is $p(\mathbf{x})$ which is defined by $p(\mathbf{x}) = \sum_{k=1}^{K} p(C_k)p(\mathbf{x}|C_k)$.

To compute $p(\mathbf{x}|C_k)$, we can further make the "naive" assumption that all features are independent from each other. Therefore, we have:

$$P(\mathbf{x}|C_k) = \prod_{i=1}^{n} p(x_i|C_k)$$

where $x_i$ is the $i^{th}$ feature.

To plug in all everything, we have:

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\prod_{i=1}^{n} p(x_i|C_k)}{\sum_{k=1}^{K} p(C_k)\prod_{i=1}^{n} p(x_i|C_k)}$$

Since the denominator is essentially independent from the class $C_k$ and the product and be converted to summation if we are in log space, the posterior $p(C_k|\mathbf{x})$ with respect to class $C_k$ given $\mathbf{x}$ can be simplified into the log-likelihood $L(C_k|\mathbf{x})$:

$$L(C_k|\mathbf{x}) = log(p(C_k)) + \sum_{i=1}^{n} log(p(x_i|C_k))$$

If we don't have any prior information about which class is more probable to occur than others, we can assume they are equiprobable. Then, we can further simplify the log-likelihood into:

$$L(C_k|\mathbf{x}) = \sum_{i=1}^{n} log(p(x_i|C_k))$$

To really implement Naive Bayes classifier, you need to first estimate the likelihood $p(x_i|C_k)$ for all features $x_i$ and classes $C_k$. This process is called training which you take some observations that you know which classes they were originally from and count the frequency of each feature showing up in the data from each class. Then, you need to implement prediction which is given a query data that you don't know the class and you compute the posterior by plug in to the formula above and then take the class that has the highest likelihood score.

In the context of our course, the observation $\mathbf{x}$ is a DNA sequence that is from species $C_1$. Suppose we represent the DNA sequence using 2-mer frequency, e.g., DNA sequence $AATCGAAGGCT$ is represented by the following frequency table:

| 2-mer | AA | AC | AG | AT | CA | CC | CG | CT | GA | GC | GG | GT | TA | TC | TG | TT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 2 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 |

The number of features is $4^k$ where $k$ refers to k-mer. In this case, we are dealing with 2-mer, so the number of features is $4^2 = 16$. Then we can estimate the likelihood $p(x_i|C_1)$ for all 2-mers.

| 2-mer | AA | AC | AG | AT | CA | CC | CG | CT | GA | GC | GG | GT | TA | TC | TG | TT |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $p(x_i\|C_1)$ | $\frac{2}{10}$ | 0 | $\frac{1}{10}$ | $\frac{1}{10}$ | 0 | 0 | $\frac{1}{10}$ | $\frac{1}{10}$ | $\frac{1}{10}$ | $\frac{1}{10}$ | $\frac{1}{10}$ | 0 | 0 | $\frac{1}{10}$ | 0 | 0 |

Note that we want to compute the posterior in log space but there are some zeros that makes it impossible to take the logarithm. Therefore, we add 1 to each of the count then compute the probability. This process is called smoothing. You can add other values, e.g., 0.001, too. Here we just add 1 for simplicity.

| 2-mer | AA | AC | AG | AT | CA | $\cdots$ | GT | TA | TC | TG | TT |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $log(p(x_i\|C_1))$ | $log(\frac{3}{26})$ | $log(\frac{1}{26})$ | $log(\frac{2}{26})$ | $log(\frac{2}{26})$ | $log(\frac{1}{26})$ | $\cdots$ | $log(\frac{1}{26})$ | $log(\frac{1}{26})$ | $log(\frac{2}{26})$ | $log(\frac{1}{26})$ | $log(\frac{1}{26})$ |

If you have another DNA sequence from a different species $C_2$, you can apply the same process to estimate the log-probability for all the k-mers. Once you have done it for all species from $C_1$ to $C_K$. Your training process is done.

Now consider the prediction process. Suppose we have a query DNA sequence that we want to figure out which species it was originally from, e.g., $AAACGTTT$. I can just compute the log-likelihood score $L(C_k|\mathbf{x})$ for species $C_1$ by:

$$L(C_1|\mathbf{x}) = log(p(AA|C_1))+log(p(AA|C_1))+log(p(AC|C_1))+log(p(CG|C_1))+log(p(GT|C_1))+log(p(TT|C_1))+log(p(TT|C_1))$$

Note that in prediction process, you have to slide across all the k-mer and add the corresponding log-probability together. Plug in the numbers we estimated above, we have:

$$L(C_1|\mathbf{x}) = log(\frac{3}{26}) + log(\frac{3}{26}) + log(\frac{1}{26}) + log(\frac{2}{26}) + log(\frac{1}{26}) + log(\frac{1}{26}) + log(\frac{1}{26}) = -19.9163$$

We compute $L(C_k|\mathbf{x})$ for all $K$ classes, then we can assign the query to the class whose L is largest.

2. **Question:** Implement the Naive Bayes Classifier described above, train the classifier with three training DNA sequences and compute the log-likelihood score for a query DNA sequence. File **Limnohabitans.fasta**, **t_vulcanus_rbcl.fasta**, **s_thermotolerans.fasta** are training DNA sequences and **uncultured.fasta** is the query DNA sequence. Hint: you can modify the function you programmed for in-class activity 5 to count the k-mer frequency and estimate the probability. **Using 3-mers instead of 2-mers as your features to train and compute the posterior.**