Tsinghua-Berkeley Shenzhen Institute
Learning from Data
Fall 2019

Programming Assignment 2

Issued: Friday 11th October, 2019 **Due:** Friday 25th October, 2019

2.1. (4 points) *MLE.* Recall that the naive Bayes model tries to maximize the likelihood of the joint distribution $P(X,Y)$ as:

$$L(\phi_y, \phi_j(x|y)) = \prod_{i=1}^{m} p(x^{(i)}, y^{(i)})$$

where the $\phi_j(x|y) := p(X_j = x|Y = y)$ and $\phi_y := p(Y = y)$ are the parameters required to be estimated for the model. Here, the $x_j, \forall j \in \{1, .., d\}$ is assumed be binary-valued: $x \in \mathcal{X} := \{0, 1\}$, and the label $y \in \mathcal{Y} := \{1, ..., K\}$. Please derive the **maximum likelihood estimates** of the $\phi_j(x|y)$ and ϕ_y for the NB model.

Solution: The log-likelihood of the equation described above can be transformed as:

$$\begin{aligned} \log L(\phi_y, \phi_j(x|y)) &= \sum_{i=1}^{m} \log p(x^{(i)}, y^{(i)}) \\ &= \sum_{i=1}^{m} \log \left(p(y^{(i)}) \prod_{j=1}^{d} p(x_j^{(i)} \mid y^{(i)}) \right) \\ &= \underbrace{\sum_{i=1}^{m} \log p(y^{(i)})}_{\textbf{(I)}} + \underbrace{\sum_{i=1}^{m} \sum_{j=1}^{d} \log p(x_j^{(i)} \mid y^{(i)})}_{\textbf{(II)}} \end{aligned} \tag{1}$$

First, let's concentrate on the part (**I**) in the above Eq.(1):

$$\begin{aligned} \sum_{i=1}^{m} \log p(y^{(i)}) &= \sum_{y \in \mathcal{Y}} \sum_{i=1}^{m} \mathbb{1}\{y^{(i)} = y\} \log \phi_y \\ &= \sum_{y=1}^{K} \text{count}(y) \log \phi_y \end{aligned} \tag{2}$$

Our goal is to maximize the Eq.(2) subject to the constraints $\phi_y > 0$ and $\sum_{y \in \mathcal{Y}} \phi_y = 1$, which is easy to achieve by introducing a **Lagrangian** as:

$$g(\lambda, \phi_y) = \sum_{y=1}^{K} \text{count}(y) \log \phi_y - \lambda \left(\sum_{y=1}^{K} \phi_y - 1 \right) \tag{3}$$

Note that we do not include the part (**II**) above because it has nothing to do with the ϕ_y. Now differentiating with respect to ϕ_y gives

$$\frac{\partial g(\lambda, \phi)}{\partial \phi_y} = \frac{\text{count}(y)}{\phi_y} - \lambda \tag{4}$$

and setting the derivatives to zero gives

$$\phi_y = \frac{\text{count}(y)}{\lambda} \tag{5}$$

Taking the summation of Eq.(5) over y gives

$$\sum_y \phi_y = \frac{1}{\lambda} \sum_y \text{count}(y) = 1 \tag{6}$$

hence we know that $\lambda = \sum_y \text{count}(y)$, and we get the MLE of the ϕ_y as:

$$\phi_y = \frac{\text{count}(y)}{\sum_{y=1}^{K} \text{count}(y)} = \frac{\text{count}(y)}{m} \tag{7}$$

Similarly, the part (**II**) can be transformed like:

$$\begin{aligned}(\text{II}) &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \sum_{j=1}^{d} \sum_{i=1}^{m} \mathbb{1}\{x_j^{(i)} = x \wedge y^{(i)} = y\} \phi_j(x \mid y) \\ &= \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \text{count}_j(x \mid y) \phi_j(x \mid y)\end{aligned} \tag{8}$$

Our goal now is to maximize the above equation with the constraints that $\phi_j(x \mid y) > 0$ and

$$\sum_{x \in \mathcal{X}} \phi_j(x \mid y) = 1 \tag{9}$$

Building a Lagrangian as

$$h(\lambda, \phi_j(x \mid y)) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} \text{count}_j(x \mid y) \phi_j(x \mid y) - \lambda \left(\sum_{x \in \mathcal{X}} \phi_j(x \mid y) - 1 \right) \tag{10}$$

Same, let's take the derivatives $\frac{\partial h(\lambda, \phi_j(x|y))}{\partial \phi_j(x|y)}$ and make it zero, we will obtain

$$\phi_j(x \mid y) = \frac{\text{count}_j(x \mid y)}{\lambda} \tag{11}$$

Combining the Eq.(9) and the Eq.(11), the MLE of $\phi_j(x \mid y)$ is

$$\phi_j(x \mid y) = \frac{\text{count}_j(x \mid y)}{\sum_{x \in \mathcal{X}} \text{count}_j(x \mid y)} = \frac{\text{count}_j(x \mid y)}{\text{count}(y)} \tag{12}$$

2.2. (6 points) *Naive Bayes.* We have prepared a binary classification dataset drawn from the UCI Adult which contains 1,605 data with 123 features in total. The task here is to build a **Bernoulli naive Bayes** classifier that does inference as:

$$p(y=k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=k)p(y=k)}{p(\boldsymbol{x})} = \frac{p(y=k)\prod_{i=1}^{n} p(x_i|y=k)}{p(\boldsymbol{x})}$$

The Bernoulli naive Bayes implements the naive Bayes training and classification algorithms for data that is distributed according to **multivariate Bernoulli distributions**; i.e., there may be multiple features but each one is assumed to be a binary-valued (Bernoulli, boolean) variable. Therefore, this class requires samples to be represented as binary-valued feature vectors. See details in the **naive_bayes.py**.

Notice:

1. Again, use matrix operations other than loops for efficiency. If the running time exceeds 5 minutes, you will get point deductions.
2. You are ought to acquire at least 75% test accuracy, and the correctness of your implementation will also be checked.
3. **Submit your solution of question 2.1 in pdf and the naive_bayes.py together!**

Tips:

1. Do not forget the Laplace smoothing.
2. Use $\log \prod(\cdot) = \sum \log(\cdot)$ to avoid operating on products.
3. Note that the $P(x_i|y) = P(i|y)x_i + (1 - P(i|y))(1 - x_i)$ is used in Bernoulli naive Bayes's decision rule. You can compare your results with the BernoulliNB in **scikit-learn**, if your implementation is right, the results test accuracy shall be the same (or very close).