Homework #5

Yongjin Shin
20090488
Industrial Management Engineering, POSTECH
dreimer@postech.ac.kr

September 3, 2018

Problem 1. Show detailed derivation of the Naive Bayes Classifier (NBC) algorithm explained in class.

Solution Compared to logistic regression, which is one of discriminative models, naive bales classification is a generative model. Assume the model takes input data formed like $\{x_i, y_i : i \in (N)\}$ and y is in finite set of classes ($k \in K$). Also, let x_i is composed of D features with either 0 or 1 but not both ($x \in \{0,1\}^D$).

Not like logistic regression, which obtains parameter θ to decide where the decision boundary is, however, naive bayes classifier needs likelihood and prior to find posterior.

$$\begin{aligned} p(y_n|x_n,\theta,\pi) &= \frac{p(x_n|y_n,\theta)p(y_n|\pi)}{p(x_n|\theta,\pi)} \\ &= \frac{p(x_n|y_n,\theta)p(y_n|\pi)}{\Sigma_{k=1}^{K} p(x_n|y_n=k,\theta)p(y_n=k|\pi)} \end{aligned} \tag{1}$$

The term naive bayes is derived from the assumption such that when it comes to find likelihood of x_i, the model infers it as conditional independence. Therefore:

$$p(x_n|y_n=k,\theta) = \prod_{d=1}^{D} p(x_{d,n}|y_n=k,\theta) \tag{2}$$

In addition, the parameter π_k ($k \in K$) to deduce prior based on given training dataset needs figured out. Thus we can conclude the naive bayes classifier as follows:

$$\begin{aligned} p(y_n|x_n,\theta,\pi) &\propto p(x_n|y_n,\theta)p(y_n|\pi) \\ &= \prod_{k=1}^{K}[p(x_n|y_n=k,\theta)p(y_n=k|\pi)] \\ &= \prod_{k=1}^{K}[\prod_{d=1}^{D}(p(x_{d,n}|y_n=k,\theta))p(y_n=k|\pi)] \\ &= \prod_{k=1}^{K}[\prod_{d=1}^{D}(\theta_{d,k}^{\mathbb{I}[y_n=k]\mathbb{I}[x_{d,n}=1]}(1-\theta_{d,k})^{\mathbb{I}[y_n=k]\mathbb{I}[x_{d,n}=0]})p(y_n=k|\pi)] \\ &= \prod_{k=1}^{K}[\prod_{d=1}^{D}(\theta_{d,k}^{\mathbb{I}[y_n=k]\mathbb{I}[x_{d,n}=1]}(1-\theta_{d,k})^{\mathbb{I}[y_n=k]\mathbb{I}[x_{d,n}=0]})\pi_k^{\mathbb{I}[y_n=k]}] \end{aligned} \tag{3}$$

Therefore, the likelihood of $D = \{(x_n, y_n|n = 1, ..., N\}$ is given by:

$$
\begin{aligned}
p(D|\theta,\pi) &= \prod_{n=1}^{N}[p(x_n,y_n|\theta,\pi)] \\
&= \prod_{n=1}^{N}[p(x_n|y_n,\theta)p(y_n|\pi)] \\
&= \prod_{n=1}^{N}[\prod_{k=1}^{K}[\pi_k^{\mathbb{I}[y_n=k]}\prod_{d=1}^{D}(\theta_{d,k}^{\mathbb{I}[y_n=k]\mathbb{I}[x_{d,n}=1]}(1-\theta_{d,k})^{\mathbb{I}[y_n=k]\mathbb{I}[x_{d,n}=0]})]] \\
&= (\prod_{k=1}^{K}\pi_k^{N_k})(\prod_{k=1}^{K}\prod_{d=1}^{D}\theta_{d,k}^{N_{d,k}}(1-\theta_{d,k})^{N_k-N_{d,k}})
\end{aligned}
\tag{4}
$$

where

$$
N_{d,k} = \Sigma_{n=1}^{N}\mathbb{I}[y_n=k]\mathbb{I}[x_{d,n}=1], \quad N_k = \Sigma_{n=1}^{N}\mathbb{I}[y_n=k] \tag{5}
$$

If we find θ and π which represent given training data well, we will be able to recognize some probability aspects of each features in every classes.

By MLE, we can optimize θ and π, separately. Thus, from equation (4), the log likelihood can be represented as follows:

$$
\begin{aligned}
\mathcal{L} &= \log p(D|\theta,\pi) \\
&= \Sigma_{k=1}^{K}N_k\log\pi_k + \Sigma_{k=1}^{K}\Sigma_{d=1}^{D}[N_{d,k}\log\theta_{d,k} + (N_k - N_{d,k})\log(1-\theta_{d,k})]
\end{aligned}
\tag{6}
$$

1) Likelihood(θ)

To solve $\frac{\partial\mathcal{L}(\theta)}{\partial\theta_{d,k}} = 0$ for $\theta_{d,k}$:

$$
\frac{\partial\mathcal{L}(\theta)}{\partial\theta_{d,k}} = \frac{N_{d,k}}{\theta_{d,k}} - \frac{(N_k - N_{d,k})}{(1-\theta_{d,k})} = 0 \tag{7}
$$

$$
\hat{\theta}_{d,k} = \frac{N_{d,k}}{N_k} \tag{8}
$$

2) The class prior(π)

Let $\mathcal{L}(\pi)$ be remained with only π_k terms so that it would be $\Sigma_{k=1}^{K}N_k\log\pi_k$. Meanwhile, as $\Sigma_{k=1}^{K}\pi_k = 1$, MLE comes to be a constraints optimization problem[2]. A new variable(λ) called a Lagrange multiplier is introduced to solve this problem:

$$
\mathcal{L}(\pi) = \Sigma_{k=1}^{K}N_k\log\pi_k + \lambda[1 - \Sigma_{k=1}^{K}\pi_k] \tag{9}
$$

Therefore, $\nabla_{\pi_k,\lambda}\mathcal{L}(\pi,\lambda) = 0$ needs to be calculated:

$$
\frac{\partial\mathcal{L}(\pi,\lambda)}{\partial\lambda} = 1 - \Sigma_{k=1}^{K}\pi_k = 0 \tag{10}
$$

$$
\frac{\partial\mathcal{L}(\pi,\lambda)}{\partial\pi_k} = \frac{N_k}{\pi_k} - \lambda = 0 \tag{11}
$$

From equation (10) and (11), we can solve for using the sum-to-one constraint:

$$
\begin{aligned}
\Sigma_{k=1}^{K}N_k &= \lambda\Sigma_{k=1}^{K}\pi_k \\
N &= \lambda
\end{aligned}
$$

Thus,

$$\hat{\pi}_k = \frac{N_k}{N}$$

Hence, with the result of the model training, we can find the possibility of being class k with the information about existences(either 0 or 1, exclusively) of some features without any decision boundary. The predicted class can be computed by given a new input x_* as follows(we assume that p(x—y) follows Bernoulli distribution):

$$\begin{aligned} p(y_* = k|x_*, \mathcal{D}) &\propto p(x_*|y_* = k, \mathcal{D})p(y_* = k|\mathcal{D}) \\ &= [\prod_{d=1}^{D} \int p(x_{d,*}|y_* = k, \theta_{d,k})p(\theta_{d,k}|\mathcal{D})d\theta_{d,k}][\int p(y_* = k|\pi)p(\pi|\mathcal{D})d\pi] \\ &= [\prod_{d=1}^{D} \mathbb{E}(p(x_{d,*}|y_* = k, \theta)][\mathbb{E}(p(y_* = k|\pi)] \\ &= \hat{\pi}_k \prod_{d=1}^{D} \hat{\theta}_{d,k}^{\mathbb{I}[x_{d,n}=1]}(1 - \hat{\theta}_{d,k})^{\mathbb{I}[x_{d,n}=0]} \end{aligned} \tag{12}$$

Do this computation for all classes $k \in K$ and then normalize them to be $\Sigma_{k=1}^{K} p(y_* = k|x_*, \mathcal{D}) = 1$
(Note: Two algorithms for the model training and prediction will be given in problem 2)

Problem 2. Implement your own NBC algorithm and evaluate the classification accuracy using the tweet dataset.

Solution

1. Method

Algorithm 1: NBC Model Training

input : Training Dataset $\mathcal{D} = \{(x_n, y_n)|n = 1, ..., N\}$
output: $\hat{\pi}_k, \quad \hat{\theta}_{d,k}$

Initialize $N_k = 0$, and $N_{d,k} = 0$
for *n = 1,2,3, ..., N* **do**
  $k = y_n$
  $N_k = N_k + 1$
  for *d = 1,...,D* **do**
    if $x_{d,n} = 1$ **then**
      $N_{d,k} = N_{d,k} + 1$
    end
  end
end
Return $\hat{\pi}_k = \frac{N_k}{N}$ and $\hat{\theta}_{d,k} = \frac{N_{d,k}}{N_k}$

For the prediction, we need to compute posterior

$$p(y = k|x) = \frac{p(x|y = k)p(y = k)}{\sum_{j=1}^{K} p(x|y = j)p(y = j)} \tag{13}$$

Due to $\sum p(x|y) = 1$, if x_n has large dimension, $\sum p(x|y = k)$ can easily be very Small number so that there could be numerical underflow in the middle of computation. Therefore we need log-sum trick; Logarithms has a property such that we can factor out the term and represent the remaining term with relative to the factored out term. Log-sum trick can be represented as follows:

$$\log p(y = k|x) = a_k - \log \Big[\sum_{j=1}^{K} \exp a_j \Big] \tag{14}$$

Let $a_{max} = max_k a_k$. Then the exponential term can be transformed as follows:

$$\begin{aligned} \log [\sum_{j=1}^{K} \exp a_j] &= \log \Big[\exp a_{max} \Big(\sum_{k=1}^{K} \exp (a_k - a_{max}) \Big) \Big] \\ &= a_{max} + \log \sum_{k=1}^{K} \exp (a_k - a_{max}) \end{aligned} \tag{15}$$

Algorithm 2: Prediction with NBC

input : $\hat{\pi}_k$, $\hat{\theta}_{d,k}$ and inputs x_n
output: Predicted class $\hat{y}_n$

Initialize $N_k = 0$, and $N_{d,k} = 0$
for n = *1,2,3, ..., N* **do**
  for k = *1,2,3, ..., K* **do**
    $l_{k,n} = \log \hat{\pi}_k$
    for d = *1,2,3,...,D* **do**
      if $x_{d,n} = 1$ **then**
        $l_{k,n} = l_{k,n} + \log \hat{\theta}_{d,n}$
      else
        $l_{k,n} = l_{k,n} + \log (1 - \hat{\theta}_{d,n})$
      end
    end
  end
end
$p_{k,n} = \exp(l_{k,n} - logsumexp(l_{:,n}))$
Return $\hat{y}_n = argmax_K p_{k,n}$

1.1 Dataset

Training data set is composed of over 47,000 tweet data with labels of 1 and 0. Test data set has over 5,800 tweet data.

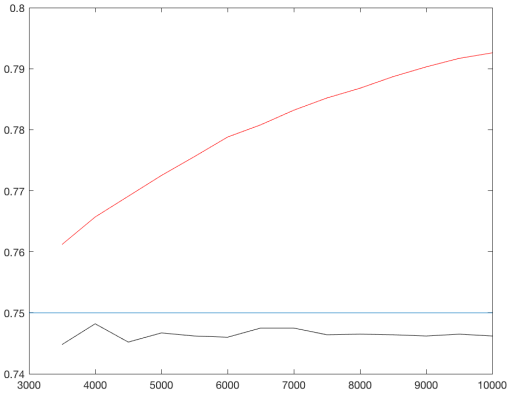
1.2 Preprocessing

Since data set is a bunch of sentences, each sentence needs to be separated into single word. In the end, separated words consists of word vector used as a feature. Training data set has 34,182 unique words. Among them, there are 155 *stop words*, such as *i, my, a, the , you and etc.*, which are commonly used. Therefore, 34,027 words was used as the bag of words and top K frequently used words were picked. Hence $(K \times N)$ size of training data set was made where K is the number of features and N is that of data.

2. Result

Top K	Training	Test
3500	0.7612	0.7448
4000	0.7657	**0.7482**
4500	0.7691	0.7452
5000	0.7725	0.7467
5500	0.7756	0.7462
6000	0.7788	0.746
6500	0.7808	0.7475
7000	0.7832	0.7475
7500	0.7852	0.7464
8000	0.7868	0.7465
8500	0.7887	0.7464
9000	0.7903	0.7462
9500	0.7917	0.7465
10000	**0.7926**	0.7462

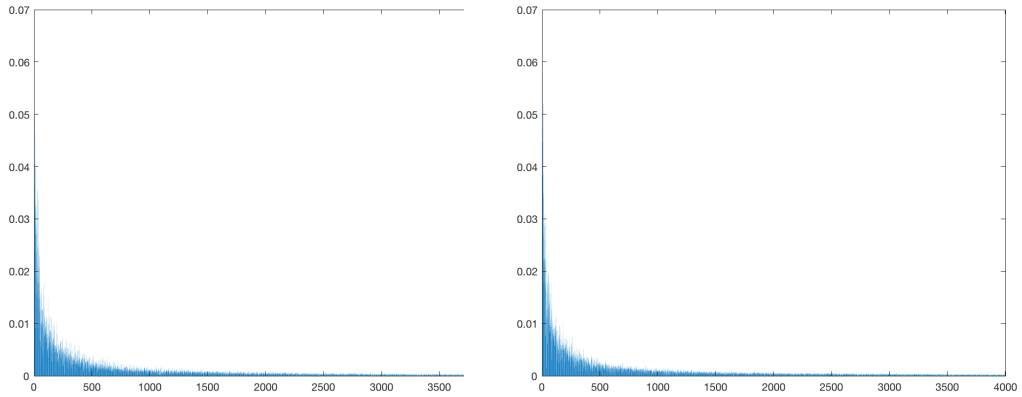
(a) Accuracy Table



(b) Accuracy Plotting

Figure 1: Classification Accuracy

From (a) of figure 1, the highest test accuracy is 74.82% when K is 4000. On the other hand, the highest training accuracy is 79.26% with 10000 K. In (b), the red color plot is of training accuracy, and the black is of test accuracy. The red one is monotonically increasing within the interval, however though test accuracy



(a) $p(x = 1|y = 0)$ (b) $p(x = 1|y = 1)$

Figure 2: Class conditional densities

is peak at 4000 overall accuracy is around 74.6%.

Figure 2 shows $\theta_{d,k}$ and obviously bag of words was sorted by highly used words so that the most of first 1000 words look have visually bigger values than others.

The average accuracy of validation set is 74.65% which is higher than the result from logistic regression. Compared to the result of training set, both models show similar accuracy results(around 79%), however, logistic showed better when it used smaller K (when K is 4000, logistic has 78.1% accuracy but NBC has only 76.57%).

3. Discussion

Even though the validation accuracy doesn't change much when top K is getting bigger, since Naive bayes classification takes much faster than logistic regression, the experiment could be preceded with top 10,000 unique words. This is due to NBC algorithm's some simple calculations, on the contrary, logistic regression has to inverse huge matrix to find parameters, which takes much longer time. As [3] mentioned that NBC reaches its asymptotic faster(O(logn)) than Logistic regression(O(n)), this implies that NBC has bigger scale-ability than logistic regression.

Another worth to mention is that NBC assumes the features conditionally independent, so that it is robust toward the data quality. Even if there are some missing or omitted information in real data set, it will just skip and handle it well.Therefore NBC has a higher bias but lower variance than logistic regression. Thus, if the data set follows the bias, then NBC will be better than logistic regression.However tweet dataset is not proper one to check this advantage of NBC.

References

[1] Kevin Murpy. *Machine Learning: A probabilistic Perspective.*

[2] Wikipedia *https://en.wikipedia.org/wiki/Lagrange_multiplier.*

[3] Andrew Y. Ng. Michael I. Jordan. *On Discriminate vs Generative Classifiers: A comparison of logistic regression and naive bayes.*