EE5907/EE5027 Week 3: Univariate Gaussian + Naive Bayes

Q1: Mixed Observations Naive Bayes

(i)

$$\begin{aligned}
p(y|x_1 = 0) &= \frac{p(y)p(x_1 = 0|y)}{p(x_1 = 0)} \\
&\propto p(y)p(x_1 = 0|y) \\
&\propto p(y) \qquad \text{because feature is uninformative.} \\
&= [0.5 \;\; 0.25 \;\; 0.25]
\end{aligned}$$

x_1 is uninformative because $\theta = [0.5 \;\; 0.5 \;\; 0.5]$, i.e., the probability of getting a head is the same for all classes. Note that x_1 will also be uninformative if $\theta = [0.4 \;\; 0.4 \;\; 0.4]$.

(ii)

$$\begin{aligned}
p(y|x_2 = 0) &= \frac{p(y)p(x_2 = 0|y)}{p(x_2 = 0)} \\
&\propto p(y)p(x_2 = 0|y) \\
&= \pi_y \frac{1}{\sqrt{2\pi\sigma_y^2}} e^{-\frac{1}{2\sigma_y^2}(0-\mu_y)^2} \\
&\propto \pi_y e^{-0.5\mu_y^2} \\
&= [0.5e^{-0.5} \;\; 0.25e^{0} \;\; 0.25e^{-0.5}] \\
&= [0.3033 \;\; 0.25 \;\; 0.1516]
\end{aligned}$$

Therefore

$$\begin{aligned}
p(y|x_2 = 0) &= [0.3033 \;\; 0.25 \;\; 0.1516]/(0.3033 + 0.25 + 0.1516) \\
&= [0.4302 \;\; 0.3547 \;\; 0.2151]
\end{aligned}$$

(iii)

$$\begin{aligned}
p(y|x_1 = 0, x_2 = 0) &= p(y|x_2 = 0) \qquad \text{because } x_1 \text{ is uninformative.} \\
&= [0.4302 \;\; 0.3547 \;\; 0.2151]
\end{aligned}$$

Exercise 3.20

(a) Since vector x is a D bit vector, and the value of each bit is binary, the vector x can take on one of 2^D possible configurations. For the full model, each of the 2^D configuration is free to take on any values, except for the one constraint that the sum of probabilities need to sum to 1. Therefore, the full model of $p(x|y=c)$ has $2^D - 1$ parameters.

(b) The naive bayes is likely to give a lower test error. With few training samples N, the naive bayes is less likely to overfit.

(c) The full model will perform better. The full model is a more accurate model. With large N, we can reliably estimate all the parameters without overfitting.

(d) **Fitting full model**
For each class c, there is a $2 \times \cdots \times 2$ matrix (D-dimensional matrix) of probability M_c we are trying to estimate. We start by initializing each M_c to 0. For each datapoint, we then add the count to the appropriate M_c (based on the class label of the datapoint). We finally normalize each M_c so that the resulting matrix sums to 1. See Algorithm 1.

Algorithm 1 Fitting Full Model

```
%Initialization
for each c ∈ C do
    N_c ← 0
    M_c ← 0
end for

%Count
for n = 1 : N do
    index ← ComputeIndexIntoM(x_n)
    M_{y_n}(index) ← M_{y_n}(index) +1
    N_{y_n} ← N_{y_n} + 1
end for

%Normalization
for each c ∈ C do
    M_c ← M_c/N_c
    N_c ← N_c/N
end for
```

The initialization requires $O(C2^D)$ operations, the counting requires $O(ND)$ operations and the normalization requires $O(C2^D)$ operations. Therefore the computational complexity of the full model is $O(ND) + O(2^D C)$.

Fitting naive Bayes Model
The process of fitting naive Bayes Model is the same as full model, but the computation complexities for "initialization", "counting" and "normalization" are $O(CD)$, $O(ND)$, and $O(CD)$ respectively. So, the computational complexity of the Naive Bayes Model is $O\left(ND\right) + O\left(DC\right)$.

(e) **Full model**

$$\underset{y}{\operatorname{argmax}}\, p(y|x_1, \cdots, x_D, \theta) = \underset{y}{\operatorname{argmax}}\, p\left(x_1, \cdots, x_D \middle| y, \theta\right) p(y|\theta)$$

For a given x, we can compute the index in $O(D)$ time. For each class c, we can then grab the corresponding entry in each M_c, multiply with $p(c)$ to obtain the posterior probability for class c (this takes O(C) time). We can then compare the posterior probability among the different classes and return the class with the highest posterior posterior probability (this takes O(C) time). Therefore the runtime is $O(D+C)$.

Naive Bayes Model

$$
\begin{aligned}
&\operatorname*{argmax}_y p\left(y|\, x, \theta\right) \\
=&\operatorname*{argmax}_y p\left(x|\, y, \theta\right) P\left(y|\, \theta\right) \\
=&\operatorname*{argmax}_y p\left(x_1|\, y, \theta\right) p\left(x_2|\, y, \theta\right) \cdots p\left(x_D|\, y, \theta\right) p\left(y|\, \theta\right)
\end{aligned}
$$

For a given x and given class c, we have to multiply D likelihood values with $p(c)$ to obtain the posterior probability (this takes O(CD) time). We can then compare the posterior probability among the different classes and return the class with the highest posterior posterior probability (this takes O(C) time). Therefore the runtime is $O(CD)$ time.

(f)

$$p(y|x_v, \theta) = \frac{p(x_v|y, \theta)p(y|\theta)}{p(x_v|\theta)} = \frac{p(y|\theta)\sum_{x_h} p(x_v, x_h|y, \theta)}{p(x_v|\theta)} \tag{1}$$

the optimization is only based on the numerator $p(y|\theta)\sum_{x_h} p(x_v, x_h|y, \theta)$

Full model $\sum_{x_h} p(x_v, x_h|y, \theta)$ cannot be simplified further, so we have to compute $p(x_v, x_h|y, \theta)$ where x_v corresponds to the test case, and x_h corresponds to a particular configuration (with 2^h possible configurations) and sum them all together. So the computational complexity is $O(2^h(C+D))$ (assuming for each configuration x_v, x_h, we compute the index and use it for all classes c).

Naive Bayes model
$\sum_{x_h} p(x_v, x_h|y, \theta) = p(x_v|y, \theta)$, so we can simply ignore the hidden variables. So the computational complexity is $O\left(CV\right)$, where V is the number of visible variables.

Q3: Posterior Predictive Distribution for Exponential Distribution

(a)
- (i)
  -

$$
\begin{aligned}
\lambda_{ML} &= \operatorname*{argmax}_\lambda p(\{x_1, \cdots, x_N\}|\lambda) \\
&= \operatorname*{argmax}_\lambda \prod_{n=1}^{N} p(x_n|\lambda) \\
&= \operatorname*{argmax}_\lambda \prod_{n=1}^{N} \lambda e^{-\lambda x_n} \\
&= \operatorname*{argmax}_\lambda N \log\lambda - \lambda \sum_{n=1}^{N} x_n
\end{aligned}
$$

Differentiating with respect to λ, we get

$$\frac{N}{\lambda} - \sum_{n=1}^{N} x_n$$

Setting it to zero, we get

$$\frac{N}{\lambda} - \sum_{n=1}^{N} x_n = 0$$

$$\lambda = \frac{N}{\sum_{n=1}^{N} x_n}$$

- (ii)
  - The problem with this approach is that using the plug-in estimator to predict new data x_{N+1} will be overly confident.
  - One could place a prior on λ and then compute the posterior predictive distribution of new data x_{N+1}

(b) (i)

$$\begin{aligned}
p(\lambda|D) &= \frac{p(D|\lambda)p(\lambda)}{p(D)} \\
&\propto p(D|\lambda)p(\lambda) \\
&= \lambda^N e^{-\lambda \sum_{n=1}^{N} x_n} \frac{\beta^\alpha}{\Gamma(\alpha)} \lambda^{\alpha-1} e^{-\lambda\beta} \\
&\propto \lambda^{N+\alpha-1} e^{-\lambda\left[\beta + \sum_{n=1}^{N} x_n\right]} \\
&= \text{Gamma}(\lambda; \alpha', \beta'),
\end{aligned}$$

where $\alpha' = N + \alpha$ and $\beta' = \beta + \sum_{n=1}^{N} x_n$

(ii) A direct approach is to compute

$$p(x_{n+1}|D) = \int p(x_{n+1}|\lambda)p(\lambda|D)d\lambda,$$

by exploiting conjugate properties of the Poisson and Gamma distributions. An easier approach is to exploit Bayes' rule to compute the evidence:

$$\begin{aligned}
p(x_{N+1}|D) &= \frac{p(x_{N+1}|\lambda, D)p(\lambda|D)}{p(\lambda|x_{N+1}, D)} \\
&= \frac{p(x_{N+1}|\lambda)p(\lambda|D)}{p(\lambda|x_{N+1}, D)} \\
&= \frac{\text{exponential}(x_{N+1}; \lambda)\text{Gamma}(\lambda; \alpha', \beta')}{\text{Gamma}(\lambda; \alpha'', \beta'')}
\end{aligned}$$

To evaluate above, observe that $\alpha'' = \alpha + N + 1$ and $\beta'' = \beta + \sum_{n=1}^{N+1} x_n$

$$\begin{aligned}
p(x_{n+1}|D) &= \frac{\lambda e^{-\lambda x_{N+1}} \frac{(\beta+\sum_{n=1}^{N} x_n)^{\alpha+N}}{\Gamma(\alpha+N)} \lambda^{\alpha+N-1} e^{-\lambda(\beta+\sum_{n=1}^{N} x_n)}}{\frac{(\beta+\sum_{n=1}^{N+1} x_n)^{\alpha+N+1}}{\Gamma(\alpha+N+1)} \lambda^{\alpha+N} e^{-\lambda(\beta+\sum_{n=1}^{N+1} x_n)}} \\
&= \frac{(\beta + \sum_{n=1}^{N} x_n)^{\alpha+N}}{\Gamma(\alpha+N)} \times \frac{\Gamma(\alpha+N+1)}{(\beta+\sum_{n=1}^{N+1} x_n)^{\alpha+N+1}} \\
&= (\alpha+N) \frac{(\beta+\sum_{n=1}^{N} x_n)^{\alpha+N}}{(\beta+\sum_{n=1}^{N+1} x_n)^{\alpha+N+1}}
\end{aligned}$$