COMP9318 (16S2) ASSIGNMENT 2 SAMPLE SOLUTION

Q1

(1). Consider entropy after splitting using A, B, and C as:

```
A | + -
=======
0 | 3 1
1 | 3 3
```

$E(A) = 0.9245$.

```
B | + -
=======
0 | 5 0
1 | 1 4
```

$E(B) = 0.3610$.

```
C | + -
=======
0 | 4 3
1 | 2 1
```

$E(C) = 0.9651$.

Therefore, we choose B to split first.

Consider the partition with $B = 0$, all records belong to the same class (+), so we can stop.

Consider the partition with $B = 1$, we have to further split it using either A or C.

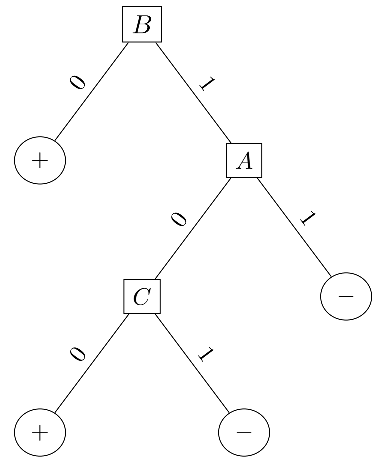
```
A | + -
=======
0 | 1 1
1 | 0 3
```

$E(A) = 0.4$.

```
C | + -
=======
0 | 1 3
1 | 0 1
```

$E(A) = 0.6490$.

Therefore, we choose A to split. While the partition for $A = 1$ clearly belong to the $-$ class, there is a tie for the other partition. It is obvious that a further test of C (the only test left) resolves the ambiguity. Hence, the final tree is:



(2). The precision of the tree on the training dataset is $\frac{10}{10} = 1.0$.
(3). The precision of the tree on the testing dataset is $\frac{2}{5} = 0.4$, as it errs on the 2nd, 3rd, and 4th testing records.
(4). The function $f(x) = x \log(x)$ is a convex function and hence the (generalized) Jensen's Inequality can be applied, i.e., for $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$, we have

$$\sum_i \lambda_i f(x_i) \geq f(\sum_i \lambda_i x_i)$$

Let the training data contain n instances. Now consider each class C_i separately. Assume that it has n_i instances. If we split on an attribute, all such n_i instances will be partitioned into multiple partitions, each with $n_i^{(j)}$ instances. In the meanwhile, the training data is split into multiple partitions, each with p^j instances. Let $\lambda^{(j)} \stackrel{\text{def}}{=} \frac{p^{(j)}}{n}$.

Now we consider their contributions to the (weighted) entropy:

$$\sum_j \lambda^{(j)} f(\frac{n_i^{(j)}}{p_j}) \geq f(\sum_j \lambda^{(j)} \frac{n_i}{p_i}) = f(\sum_j \frac{p^{(j)}}{n} \cdot \frac{n_i^{(j)}}{p^{(j)}}) = f(\sum_j \frac{n_i^{(j)}}{n}) = f(\frac{\sum_j n_i^{(j)}}{n}) = f(\frac{n_i}{n})$$

Note that the last term is exact the probability of class C_i in the unsplit training data. Therefore, by summing over all classes, and taking the negation on both side, we can easily derive the conclusion.
(5). Under the given $\boldsymbol{w}$, the predicted probability of each data is as follows:

D	A	B	C	class	prob
1	0.0	0.0	0.0	1.0	0.549834
1	0.0	0.0	1.0	1.0	0.645656
1	0.0	1.0	0.0	1.0	0.524979
1	0.0	1.0	1.0	0.0	0.622459
1	1.0	0.0	0.0	1.0	0.622459

1	1.0	0.0	0.0	1.0	0.622459
1	1.0	1.0	0.0	0.0	0.598688
1	1.0	0.0	1.0	1.0	0.710950
1	1.0	1.0	0.0	0.0	0.598688
1	1.0	1.0	0.0	0.0	0.598688

Where D is the offset term. Then the data log likelihood is

$$\sum_{i=1}^{10} y^{(i)} \log\left(p^{(i)}\right) + (1 - y^{(i)}) \log\left(1 - p^{(i)}\right) = -6.6825.$$

Q2

(1). See Algorithm 1.

Algorithm 1: k-means(D, k)

Data: D is a dataset of n d-dimensional points; k is the number of clusters.

Initialize k centers $C = [c_1, c_2, \ldots, c_k]$;
canStop $\leftarrow$ **false**;
while *canStop* $=$ **false do**
    Initialize k empty clusters $G = [g_1, g_2, \ldots, g_k]$;
    for each data point $p \in D$ **do**
        $c_x \leftarrow$ NearestCenter(p, C);
        g_{c_x}.append(p);
    canStop $\leftarrow$ **true**;
    for each group $g \in G$ **do**
        $old\_c_i \leftarrow c_i$;
        $c_i \leftarrow$ ComputeCenter(g);
        if $old\_c_i \neq c_i$ **then**
            canStop $\leftarrow$ **false**;
return G;

(2). It is obvious that we only need to prove the same conclusion for a cost function y that sums up square of the $dist$, i.e.,

$$\begin{aligned} f = cost'(g_1, \ldots, g_k) &= \sum_{i=1}^{k} cost'(g_i) \\ &= \sum_{i=1}^{k} \left(\sum_{p \in g_i} (dist(p, c_i))^2 \right) \end{aligned}$$

Within each iteration, we first assign each p to its nearest center, and then update the centers. It is easy to see first step always reduces f. For the second step, we study the extreme value of f. It is obvious that we can study $cost'(g_i)$ individually. Assume in two dimensional space, c_i is represented as (x, y), then we take partial derivatives

$$\begin{aligned}\frac{\partial cost'(g_i)}{\partial x} &= \frac{\sum_{p \in g_i} (p.x - x)^2 + (p.y - y)^2}{\partial x} \\ &= -\sum_{p \in g_i} 2(p.x - x)\end{aligned}$$

We can obtain similar results on y.

Hence $x = \frac{\sum_{p \in g_i} p.x}{|g_i|}$ (similar for y) achieve the minimum value, which is exactly the new centers chosen at the end of each iteration.

(3). Assume the conclusion in (2). This says the cost f at the end of each iteration is monotonically decreasing. Obviously f has a lower bound of 0. Therefore, according to the "monotone convergence theorem", the cost will converge.

Q3

(1). We define the *logOdds*, and if it is larger than 0, then the prediction is positive class; otherwise, the classification is the negative class.

$$\begin{aligned}logOdds &\stackrel{\text{def}}{=} \log\left(\frac{\mathbf{Pr}[C_+|\boldsymbol{u}]}{\mathbf{Pr}[C_-|\boldsymbol{u}]}\right) \\ &= \log\left(\mathbf{Pr}[\boldsymbol{u}|C_+] \cdot \mathbf{Pr}[C_+]\right) - \log\left(\mathbf{Pr}[\boldsymbol{u}|C_-] \cdot \mathbf{Pr}[C_-]\right) \\ &= \log\left(\prod_{i=1}^{d} \mathbf{Pr}[x_i = u_i|C_+] + \log\left(\mathbf{Pr}[C_+]\right)\right) - \log\left(\prod_{i=1}^{d} \mathbf{Pr}[x_i = u_i|C_-] + \log\left(\mathbf{Pr}[C_-]\right)\right) \\ &= \left(\sum_{i=1}^{d} \log\left(\mathbf{Pr}[x_i = u_i|C_+]\right) + \log\left(\mathbf{Pr}[C_+]\right)\right) - \left(\sum_{i=1}^{d} \log\left(\mathbf{Pr}[x_i = u_i|C_-]\right) + \log\left(\mathbf{Pr}[C_-]\right)\right) \\ &= \left(\sum_{i=1}^{d} (\log\left(\mathbf{Pr}[x_i = u_i|C_+]\right) - \log\left(\mathbf{Pr}[x_i = u_i|C_-]\right))\right) + \left(\log\left(\mathbf{Pr}[C_+]\right) - \log\left(\mathbf{Pr}[C_-]\right)\right) \\ &= \sum_{i=1}^{d} \log\left(\frac{\mathbf{Pr}[x_i = u_i|C_+]}{\mathbf{Pr}[x_i = u_i|C_-]}\right) + \log\left(\frac{\mathbf{Pr}[C_+]}{\mathbf{Pr}[C_-]}\right)\end{aligned}$$

We define the following symbols to simply the above equation:

$$\begin{aligned}\alpha(i, u_i) &\stackrel{\text{def}}{=} \log\left(\frac{\mathbf{Pr}[x_i = u_i|C_+]}{\mathbf{Pr}[x_i = u_i|C_-]}\right) \\ \beta &\stackrel{\text{def}}{=} \log\left(\frac{\mathbf{Pr}[C_+]}{\mathbf{Pr}[C_-]}\right)\end{aligned}$$

Then

$$\begin{aligned}
logOdds &= \beta + \sum_{i=1}^{d} \alpha(i, u_i) \\
&= \beta + \sum_{i=1}^{d} (\alpha(i, 1) \cdot u_i + \alpha(i, 0) \cdot (1 - u_i)) \qquad (\because u_i \text{ can only take 0 or 1 value}) \\
&= \beta + \sum_{i=1}^{d} \alpha(i, 0) + (\alpha(i, 1) - \alpha(i, 0)) \cdot u_i \\
&= \gamma + \sum_{i=1}^{d} \delta_i \cdot u_i
\end{aligned}$$

In the last step, we let $\gamma = \beta + \sum_{i=1}^{d} \alpha(i, 0)$, and $\delta_i = \alpha(i, 1) - \alpha(i, 0)$

Therefore, it is a linear classifer for an extended feature vector $[1, \boldsymbol{x}]$, and the parameter

$$\boldsymbol{w}^\top = [\gamma, \delta_1, \delta_2, \ldots, \delta_d]$$

(2). The main reason is that all the $\boldsymbol{w}_i$s in NB can be learned independently of each other (thanks to the conditional independence assumption), while $\boldsymbol{w}_i$s in LR have to be learned *jointly*.