Assignment 1

COMP9318 19T1

Chris Joy (z5113243)

Q1

Consider the following cuboid with built-in support for the value **ALL**.

Sales			
Location	**Time**	**Item**	**Quantity**
Sydney	2005	PS2	1400
Sydney	2006	PS2	1500
Sydney	2006	Wii	500
Melbourne	2005	XBox 360	1700

(1)

Location	**Time**	**Item**	**Quantity**
ALL	ALL	ALL	5100
ALL	2005	ALL	3100
ALL	2006	ALL	2000
ALL	ALL	PS2	2900
ALL	ALL	Wii	500
ALL	ALL	XBox 360	1700
ALL	2005	PS2	1400
ALL	2005	XBox 360	1700
ALL	2006	PS2	1500
ALL	2006	Wii	500
Sydney	ALL	ALL	3400
Sydney	2005	ALL	1400
Sydney	2006	ALL	1500
Sydney	ALL	PS2	2900
Sydney	ALL	Wii	500
Sydney	2005	PS2	1400
Sydney	2006	PS2	1500
Sydney	2006	Wii	500
Melbourne	ALL	ALL	1700
Melbourne	2005	ALL	1700
Melbourne	ALL	XBox 360	1700
Melbourne	2005	XBox 360	1700

(2)

```
SELECT
    COALESCE(Location, "ALL") Location,
    COALESCE(Time, "ALL") Time,
    COALESCE(Item, "ALL") Item,
    SUM(Quantity) AS Quantity
FROM (
    SELECT *
    FROM Location
    CROSS JOIN Time
    CROSS JOIN Item
)
WHERE Quantity > 0
ORDER BY Location, Time, Item
```

Note COALESCE was used to mask *null* values from the table. We're also sorting the final results alphabetically, in the priority of location first then time and finally by item.

(3)

Location	Time	Item	Quantity
ALL	ALL	ALL	2900
ALL	2005	ALL	1700
ALL	2006	ALL	1500
ALL	ALL	PS2	2900
ALL	2005	PS2	1400
ALL	2006	PS2	1500
Sydney	ALL	ALL	2900
Sydney	2005	ALL	1400
Sydney	2006	ALL	2000
Sydney	ALL	PS2	2900

(4)

Given the following mapping functions described in the specs (ie. $f_{Location}(x)$ $f_{Time}(x)$ $f_{Item}(x)$), we map each dimension column using their respective mapping:

Sales			
Location	**Time**	**Item**	**Quantity**
1	1	1	1400
1	2	1	1500
1	2	3	500
2	1	2	1700

We can use this function to map a multi-dimensional point to a one-dimensional point in the array below:

$$f_{Lookup}(location, time, item) = f_{Location}(location) + 3 \cdot f_{Time}(time) + 9 \cdot f_{Item}(item)$$

Converting R into tabular form with an index and its corresponding value gives us:

ArrayIndex	**Value**
13	1400
16	1500
23	1700
34	500

$f_{Lookup}(location, time, item)$ is an injective function, meaning that there's no combination of $location$, $time$ and $item$ that would result in the same $ArrayIndex$. The range of indexes produced by this function $[0, 35]$. The maximum number of combinations is 36 (ie. $3 \times 3 \times 4$). Since the function's range equals to the maximum number of combinations, we can say that the mapping function is efficient (ie. the resultant array is dense, not sparse).

Q2

(1)

Suppose we are given a binary feature vector $\vec{x}$ of d dimensions and a class attribute y, where $y \in \{0, 1\}$.

Consider the Naive Bayes Classifier:

$$\hat{y} = \underset{y \in \{0,1\}}{argmax} \quad P(y) \prod_{i=1}^{d} P(x_i|y)$$

And a linear classifier with the model parameter $\mathbf{w}$:

$$y = \begin{cases} 1, & if \quad \mathbf{w}^T\mathbf{x} > 0 \\ 0, & otherwise. \end{cases}$$

From the decision rule, we know that our classifier will predict 1 if $P(y = 1|\vec{x}) > (P(y = 0|\vec{x})$ and 0 otherwise. Rearranging this into odds form and replacing it with $\mathbf{w}^T\mathbf{x}$, we get:

$$y = \begin{cases} 1, & if \quad \frac{P(y=1|\vec{x})}{P(y=0|\vec{x})} - 1 > 0 \\ 0, & otherwise. \end{cases}$$

Going forth, we just need to change $\frac{P(y=1|\vec{x})}{P(y=0|\vec{x})} - 1$ into a form where we can find the model parameter w. We start off by re-writing the condition, using a product, in order to deal with the feature vector $\mathbf{x}$.

$$\frac{P(y=1|\vec{x})}{P(y=0|\vec{x})} = \frac{P(y=1)\prod_{i=1}^{d} P(x_i|y=1)}{P(y=0)\prod_{i=1}^{d} P(x_i|y=0)}$$

Using the rule of subtraction and collecting terms, we get the following:

$$\frac{P(y=1)}{1-P(y=1)} \cdot \prod_{i=1}^{d} \frac{P(x_i|y=1)}{P(x_i|y=0)}$$

In order to change into some linear function of x, we can exploit the properties of log. By applying log to the whole condition, we get the following:

$$\log \frac{P(y=1)}{1-P(y=1)} + \sum_{i=1}^{d} (\log P(x_i|y=1)) - \log P(x_i|y=0))$$

By using the Bernoulli Naive Bayes assumption: $P(y) = \theta(1-\theta)^{1-y}$, we can restate the condition as the following,

$$log\frac{\theta}{1-\theta} + \sum_{i=1}^{d}(1-\theta_{i|y=1}) - \sum_{i=1}^{d}(1-\theta_{i|y=0}) + \sum_{i=1}^{d}(x_i \cdot [\log \frac{\theta_{i|y=1}}{1-\theta_{i|y=1}} - log\frac{\theta_{i|y=0}}{1-\theta_{i|y=0}}])$$

This is now as a linear function of x, where $w_i = \log \frac{\theta_{i|y=1}}{1-\theta_{i|y=1}} - log\frac{\theta_{i|y=0}}{1-\theta_{i|y=0}}$ and the bias b $= log\frac{\theta}{1-\theta} + \sum_{i=1}^{d}(1-\theta_{i|y=1})$.

(2)

With Naive Bayes (a generative model), in order to calculate the posterior probability ($P(c|x)$), we'd need to essentially learn the class and prior probability (ie $P(c)$ $P(x)$, respectively). Afterwards we'd need to compute the likelihood for each feature given a class ($P(x|c)$). This means we've already learned the weights (W_{NB}) by computing the likelihood for each feature.

However with Logistic Regression (a discriminative model), the process of learning the weights (W_{LR}) involve trying to maximize the log-likelihood function, using an algorithm such as gradient ascent.

Both Naive bayes and logistic regression are linear classifiers. Since with Naive Bayes, we just need to compute the likelihood for each feature once (ie. $P(x_i|C)$), it's much faster in terms of learning its weights, compared to logistic regression.

Q3

Suppose we have a mixture M which contains an unknown percentage of two chemical compounds S_1 and S_2. These unknown percentages of chemical compounds are q_1 and q_2, respectively. The composition of the mixture would look something like this:

$$M = (q_1 \cdot S_1) + (q_2 \cdot S_2)$$

Each chemical compound (S_1 and S_2) is made up of $m = 3$ components O_1, O_2 and O_3. The percentages of these components are different for each chemical compound. ie.

$$M = [q_1 \cdot (p_{1,1} + p_{1,2} + p_{1,3})] + [q_2 \cdot (p_{2,1} + p_{2,2} + p_{2,3})]$$

The percentages of the three components of each chemical compound is known, ie.

$$M = [q_1 \cdot (0.1 + 0.2 + 0.7)] + [q_2 \cdot (0.4 + 0.5 + 0.1)]$$

The unknown percentages of each of these chemical compounds are u_1, u_2 and u_3. In essence, our final mix's composition is:

$$M = u_1O_1 + u_2O_2 + u_3O_3$$

(1)

We can essentially use a binomial probability function to model such machine that finds the unknown percentages u_j of each component $\{O_i\}_{j=1}^{m}$. ie.

$$P(p_{i,j}|u_j) = \binom{n}{n \cdot p_{i,j}} \cdot (u_j)^{n \cdot p_{i,j}} \cdot (1 - u_j)^{n \cdot (1 - p_{i,j})}$$

We'll take $n = 10$ with increments of 0.1.

$$P(p_{i,j}|u_j) = \binom{10}{10p_{i,j}} \cdot (u_j)^{10p_{i,j}} \cdot (1 - u_j)^{10(1 - p_{i,j})}$$

Our log-likelihood function will essentially be $L(p_{i,j}|u_j) = logP(p_{i,j}|u_j)$,

$$L(p_{i,j}|u_j) = \log\left[\binom{10}{10p_{i,j}} \cdot (u_j)^{10p_{i,j}} \cdot (1 - u_j)^{10(1 - p_{i,j})}\right]$$

We can simplify $L(p_{i,j}|u_j)$ further.

$$L(p_{i,j}|u_j) = \log\left[\binom{10}{10p_{i,j}} \cdot (u_j^{10p_{i,j}} - u_j^{10})\right]$$

(2)

Given $u_1 = 0.3$, $u_2 = 0.2$ and $u_3 = 0.5$ and the log-likelihood function $L(p_{i,j}|u_j)$, we need to find the $argmax_{u_j}$ in order to obtain the MLE of q_1 and q_2.

We can find MLE_{q_1}, by summing the training instances using the log likelihood at $i = 1$.

$$\begin{aligned}
MLE_{q_1} &= \sum_{j=1}^{m} L(p_{i,j}|u_j) \\
&= \sum_{j=1}^{m} \log\left[\binom{10}{10p_{i,j}} \cdot (u_j^{10p_{i,j}} - u_j^{10})\right] \\
&= \log\left[\binom{10}{10 \times 0.1} \cdot (0.3^{10\times 0.1} - 0.3^{10})\right] \\
&+ \log\left[\binom{10}{10 \times 0.2} \cdot (0.2^{10\times 0.2} - 0.2^{10})\right] \\
&+ \log\left[\binom{10}{10 \times 0.7} \cdot (0.5^{10\times 0.7} - 0.5^{10})\right] \\
&\approx 0.537
\end{aligned}$$

Similarily we find MLE_{q_2}, by doing the same above, but with $i = 2$.

$$\begin{aligned}
MLE_{q_2} &= \sum_{j=1}^{m} L(p_{i,j}|u_j) \\
&= \sum_{j=1}^{m} \log\left[\binom{10}{10p_{i,j}} \cdot (u_j^{10p_{i,j}} - u_j^{10})\right] \\
&= \log\left[\binom{10}{10 \times 0.4} \cdot (0.3^{10\times 0.4} - 0.3^{10})\right] \\
&+ \log\left[\binom{10}{10 \times 0.5} \cdot (0.2^{10\times 0.5} - 0.2^{10})\right] \\
&+ \log\left[\binom{10}{10 \times 0.1} \cdot (0.5^{10\times 0.1} - 0.5^{10})\right] \\
&\approx 0.447
\end{aligned}$$

If we substitute $q_1 = 0.537$ and $q_2 = 0.447$ into $M = [q_1 \cdot (0.1+0.2+0.7)]+[q_2 \cdot (0.4+0.5+0.1)] \approx 1$. Which tells us that our MLEs of q_1 and q_2 are very close to the actual value.

Finally, we can approximate the percentages of each component O_1, O_2 and O_3.

$$O_1 = (0.537 \times 0.1) + (0.447 \times 0.4) \approx 24.25\%$$

$$O_2 = (0.537 \times 0.2) + (0.447 \times 0.5) \approx 33.09\%$$

$$O_3 = (0.537 \times 0.7) + (0.447 \times 0.1) \approx 42.06\%$$