COMP9318(17S1)

Data Warehousing and Data Mining

ASSIGNMENT 1

Name: CHI XU

Student ID: z5096261

Q1

1)

|  |  |  |  |
| --- | --- | --- | --- |
| Location | Time | Item | SUM(Quantity) |
| Sydney | 2005 | PS2 | 1400 |
| Sydney | 2005 | ALL | 1400 |
| Sydney | 2006 | PS2 | 1500 |
| Sydney | 2006 | Wii | 500 |
| Sydney | 2006 | ALL | 2000 |
| Melbourne | 2005 | Xbox 360 | 1700 |
| Melbourne | 2005 | ALL | 1700 |
| Sydney | ALL | PS2 | 2900 |
| Sydney | ALL | Wii | 500 |
| Sydney | ALL | ALL | 3400 |
| Melbourne | ALL | Xbox 360 | 1700 |
| Melbourne | ALL | ALL | 1700 |
| ALL | 2005 | PS2 | 1400 |
| ALL | 2005 | Xbox 360 | 1700 |
| ALL | 2005 | ALL | 3100 |
| ALL | 2006 | PS2 | 1500 |
| ALL | 2006 | Wii | 500 |
| ALL | 2006 | ALL | 2000 |
| ALL | ALL | PS2 | 2900 |
| ALL | ALL | Wii | 500 |
| ALL | ALL | Xbox 360 | 1700 |
| ALL | ALL | ALL | 5100 |

2)

Select Location, Time, Item, SUM(Quantity) from Sales GROUP BY Location, Time, Item UNION

Select Location, Time, ALL, SUM(Quantity) from Sales GROUP BY Location, Time UNION

Select Location, ALL, Item, SUM(Quantity) from Sales GROUP BY Location, Item UNION

Select ALL, Time, Item, SUM(Quantity) from Sales GROUP BY Time, Item UNION

Select ALL, ALL, Item, SUM(Quantity) from Sales GROUP BY Item UNION

Select ALL, ALL, ALL, SUM(Quantity) from Sales

3)

|  |  |  |  |
| --- | --- | --- | --- |
| Location | Time | Item | SUM(Quantity) |
| Sydney | 2006 | ALL | 2000 |
| Sydney | ALL | PS2 | 2900 |
| Sydney | ALL | ALL | 3400 |
| ALL | 2005 | ALL | 3100 |
| ALL | 2006 | ALL | 2000 |
| ALL | ALL | PS2 | 2900 |
| ALL | ALL | ALL | 5100 |

4)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Location | Time | Item | SUM(Quantity) | Offset |
| 1 | 1 | 1 | 1400 | 17 |
| 1 | 1 | 0 | 1400 | 16 |
| 1 | 2 | 1 | 1500 | 21 |
| 1 | 2 | 2 | 500 | 22 |
| 1 | 2 | 0 | 2000 | 20 |
| 2 | 1 | 3 | 1700 | 31 |
| 2 | 1 | 0 | 1700 | 28 |
| 1 | 0 | 1 | 2900 | 13 |
| 1 | 0 | 2 | 500 | 14 |
| 1 | 0 | 0 | 3400 | 12 |
| 2 | 0 | 3 | 1700 | 27 |
| 2 | 0 | 0 | 1700 | 24 |
| 0 | 1 | 1 | 1400 | 5 |
| 0 | 1 | 3 | 1700 | 7 |
| 0 | 1 | 0 | 3100 | 4 |
| 0 | 2 | 1 | 1500 | 9 |
| 0 | 2 | 2 | 500 | 10 |
| 0 | 2 | 0 | 2000 | 8 |
| 0 | 0 | 1 | 2900 | 1 |
| 0 | 0 | 2 | 500 | 2 |
| 0 | 0 | 3 | 1700 | 3 |
| 0 | 0 | 0 | 5100 | 0 |

Since we should choose a function which satisfied the variety changes of data, we are using the following function to give enough space for distinguishing different values.

Q2

1)

Following is the Bayesian classifier:

Hence we can derive an inequality function as following:

Since x is equal to either 0 or 1, then we get:

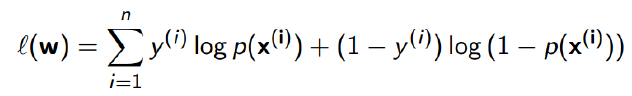
To express above inequality function in log-space, then the formula will change to be the following one:

And this is a linear equation format.

When it comes into a d+1-dimension space, the will be

2)

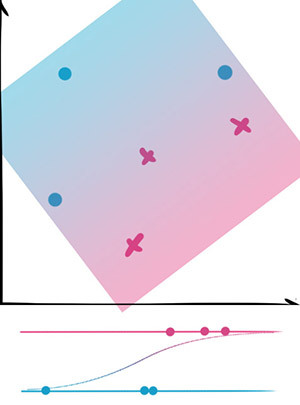
Due to Log\_likelihood of LR is shown below:



Compared with Log\_likelihood of Naïve Bayes:

When ∈ { 0, 1 }, it is more complex using LR to learn w than Naïve Bayes.

The core idea of logistic regression is finding most appropriate parameters to make predict values as close as values from sampling data. And as a picture shown below, it’s better to have value between 0 and 1 instead of being exactly 0 or 1.



Q3

1)

Initialize k centers C = [c1, c2, …, ck];

canStop ← false;

while canStop = false do

Initialize k empty clusters G = [g1, g2, …, gk];

for each data point p ∈ D do

cx ← NearestCenter(p, C);

gcx. append(p);

l(pi) ← argminDistance(pi, cj) j ∈{1,2, …, k};

for each group g ∈ G do

ci ← ComputeCenter(g);

repeat

for each ci ∈ C do

UpdateCenter(ci)

for each pi ∈ D do

minDist ← argminDistance(pi, cj) j ∈{1,2, …, k};

if minDist ≠ l(pi) then

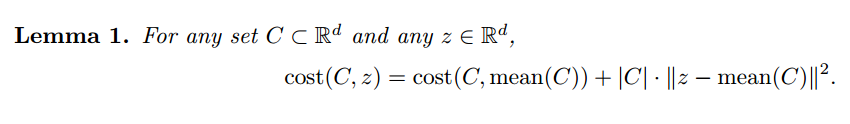
l(pi) ← minDist;

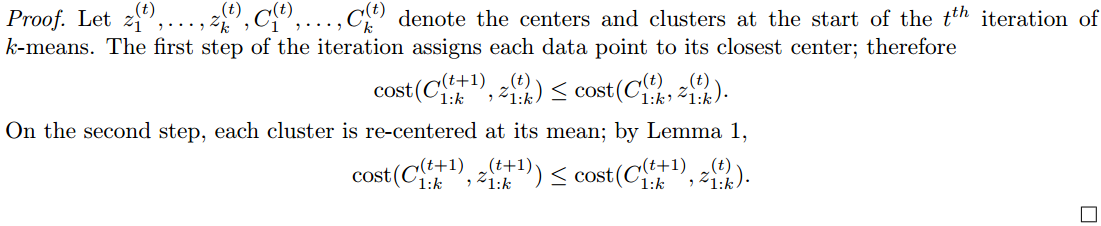
canStop ← true;

until canStop = true

return G;

2)





First, we get the mean, and the current cost is the variance.

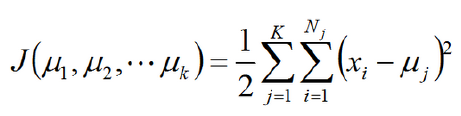
Then we calculate the Euclidean distance and for every object, if there exists another center close to this object’s current center, it will be combined to another center.

Finally, because it’s closer to another center, which cause the Euclidean distance smaller, that’s how the cost decrease.

Therefore, during the course of the k-means algorithm, the cost monotonically decreases.

3)

The final cost function would be:



The algorithms almost surely converge to a local minimum because the local variations of the loss function are conveniently bounded (semi-differentiability).