***PARM***

**There are 4 codes that run sequentially to give results:**

Parm1.java (java swings to handle images at input side🡪To convert images into BSQs)

Parm2.c(C code to convert BSQ into bSQs)

Parrm3.c(C code responsible for performing bit level data mining)

Parm4.java (java swings to convert textual results into images)

**Naming Conventions:**

**Input files:**

geo1…..geon.jpg/png/giff/jpeg=RSI images including 1 event and (n-1) contributing factors.

**Code files:**

RGB to RGB mapping(event with one Contributing factor)= parm11.java,parm22.c,parm33.c,parm44.java

RGB to Grey mapping(event with one Contributing factor)= parm1.java,parm2.c,parm3.c,parm4.java,parm5.java,parm6.java

RGB to RGB(event with multiple Contributing factors)= parm11.java,parm22.c,parm33.c,parm44.java

**Repositories:**

parm\_rsi.txt=It is the Band Sequential format.

b1b1.txt ……….. bnb4.txt=These are the bit sequential formats where n=Number\_of\_input\_images\*3.

rules.txt=To store the association rules.

**Output files:**

generated\_image1=predicted image with the lower 4bits of each byte set to zero.

generated\_image2=predicted image with the lower 4bits of each byte set to one.

**Items:**

The count of possible number of items=3(Number of bands)\*4(Number of bits per band)\*2(bit can be either 1 or 0)\*M(Number of images).

An item is represented as ABC which means it is representing the Bth bit of Ath band with the value being C.The (R,G,B) bands of first image are represented as (1,2,3) and that of 2nd image as (4,5,6) so on and so forth.

**Dimensions of files:**

**Input files:**

All the images are of the same dimensions say (NxN).

**Repositories:**

parm\_rsi.txt= Size of this file in ( Nx(Nx3x4xM))

Where

N is the image size

3 represents the bands(R,G,B) of each pixel.

4 represents the number of bits per band(only the higher order 4 bits are considered)

M represents the number of input images(If M=2 then there is an event and only one contributing factor)

b1b1…..bnb4=Size of all the bit sequential format files is NxN.

**Output files:**

generated\_image1…n=The output images are all of size NxN.

**How the process works?**

Let us consider the case of an event and one contributing factor.

**Event** : A 2x2 image where each pixel has 3 visual bands(R,G,B)

|  |  |
| --- | --- |
| 121,131,141 | 127,129,131 |
| 117,225,230 | 131,210,212 |

**Contributing Factor**: A 2x2 image where each pixel has 3 visual bands(R,G,B)

|  |  |
| --- | --- |
| 121,212,129 | 131,210,141 |
| 129,230,212 | 141,225,131 |

Once the inputs are received we create the **Band sequential format** which looks like this:

|  |
| --- |
| 121,131,141,121,212,129,127,129,131,131,210,141 |
| 117,225,230,129,230,212,131,210,212,141,225,131 |

Now these **integral values are converted into binary form** wherein only the higher order 4 bits are reserved for each byte.

|  |
| --- |
| 0111,1000,1000,0111,1101,1000,0111,1000,1000,1000,1101,1000 |
| 0111,1110,1110,1000,1110,1101,1000,1101,1101,1000,1110,1000 |

Now the **bit sequential formats** are created from the above file:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| B1b1 :   |  |  | | --- | --- | | 0 | 0 | | 0 | 1 | | B1b2 :   |  |  | | --- | --- | | 1 | 1 | | 1 | 0 | | B1b3 :   |  |  | | --- | --- | | 1 | 1 | | 1 | 0 | | B1b4 :   |  |  | | --- | --- | | 1 | 1 | | 1 | 0 | | B2b1 :   |  |  | | --- | --- | | 1 | 1 | | 1 | 1 | | B2b2:   |  |  | | --- | --- | | 0 | 0 | | 1 | 1 | | B2b3:   |  |  | | --- | --- | | 0 | 0 | | 1 | 0 | | B2b4 :   |  |  | | --- | --- | | 0 | 0 | | 0 | 1 | |
| B3b1 :   |  |  | | --- | --- | | 1 | 1 | | 1 | 1 | | B3b2 :   |  |  | | --- | --- | | 0 | 0 | | 1 | 1 | | B3b3 :   |  |  | | --- | --- | | 0 | 0 | | 1 | 0 | | B3b4 :   |  |  | | --- | --- | | 0 | 0 | | 0 | 1 | | B4b1 :   |  |  | | --- | --- | | 0 | 1 | | 1 | 1 | | B4b2 :   |  |  | | --- | --- | | 1 | 0 | | 0 | 0 | | B4b3 :   |  |  | | --- | --- | | 1 | 0 | | 0 | 0 | | B4b4 :   |  |  | | --- | --- | | 1 | 0 | | 0 | 0 | |
| B5b1 :   |  |  | | --- | --- | | 1 | 1 | | 1 | 1 | | B5b2 :   |  |  | | --- | --- | | 1 | 1 | | 1 | 1 | | B5b3 :   |  |  | | --- | --- | | 0 | 0 | | 1 | 1 | | B5b4 :   |  |  | | --- | --- | | 1 | 1 | | 0 | 0 | | B6b1 :   |  |  | | --- | --- | | 1 | 1 | | 1 | 1 | | B6b2 :   |  |  | | --- | --- | | 0 | 0 | | 1 | 0 | | B6b3 :   |  |  | | --- | --- | | 0 | 0 | | 0 | 0 | | B6b4 :   |  |  | | --- | --- | | 0 | 0 | | 1 | 0 | |

Once the bit sequential formats are created we try to find out the frequent item from each file.

**Suppose support is 50%.**We start with the file B1b1 wherein there are three 0s and one 1.so the frequent item is 110 i.e 1st bit of 1st band is 0.

So **the 1-frequent itemsets** are : 110,121,131,141,211,220,221,230,240,311,321,320,330,340,411,420,430,440,511,521,530,531,541,540,611,620,630,640.

Now for generating **2-frequent itemsets** simply perform anding of bSQs.If we AND B1b2 and B1b4,we get

B1b2 AND B1b3 =

|  |  |
| --- | --- |
| 1 | 1 |
| 1 | 0 |

Here the count of 1s is 3 greater than support count. So {121,131} forms a 2-frequent itemset.

Similarly the entire process is repeated to get n-frequent itemsets.

**Example** of n-frequent itemset : {121,131,141}

So the rules are:

{121}🡪{131,141}

{131}🡪{121,141}

{141🡪{121,131}

{121,131}🡪{141}

{121,141}🡪{131}

{131,141}🡪{121}

Then the rules are pruned based on the confidence desired.

All these rules get stored into a repository.Then these are parsed to match the LHS of a rule with the bands of pixels of the image “Contributing\_factor” of some different geographical region.Once a match is found the corresponding pixels in the “Predicted\_event” are imparted colors.

**Example:**

Suppose the rule is {441,510,631}🡪{121,231,341}

Now if the 4th bit of Red band is 1 and 1st bit of Green band is 0 and 3rd bit of Blue band is 1 of the image “Contributing\_factor”/geo3.jpg then,

2nd bit of Red band is set to 1 and 3rd bit of Green band is set to 1 and 4th bit of Blue band is set to 1 of the image “Predicted\_event”/generated\_image.jpeg.

**Block diagram depicting code and repository interactions for RGB to RGB Mapping:**

Contributing factor-1

Event-2

**Event-1**

Parm11.java

Parm\_rsi.txt

Parm22.c

B1b1.txt

B2b4.txt

B2b3.txt

B2b2.txt

B2b1.txt

B1b4.txt

B1b3.txt

B1b2.txt

B3b3.txt

B3b2.txt

B3b1.txt

B4b4.txt

B4b3.txt

B4b2.txt

B4b1.txt

B3b4.txt

B5b1.txt

B5b2.txt

B5b3.txt

B5b4.txt

B6b1.txt

B6b2.txt

B6b3.txt

B6b4.txt

Parm33.c

Rules.txt

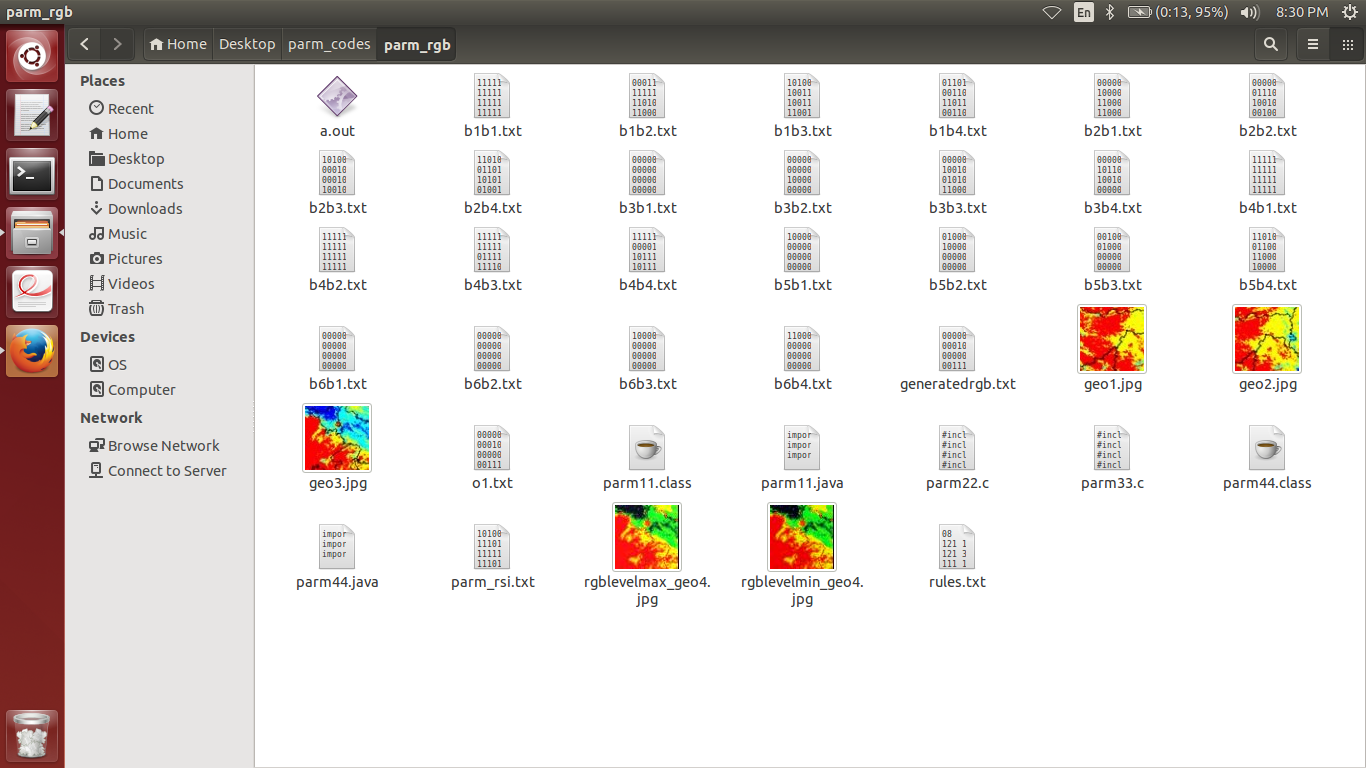
Parm44.java

Generated\_iamge2.jpg

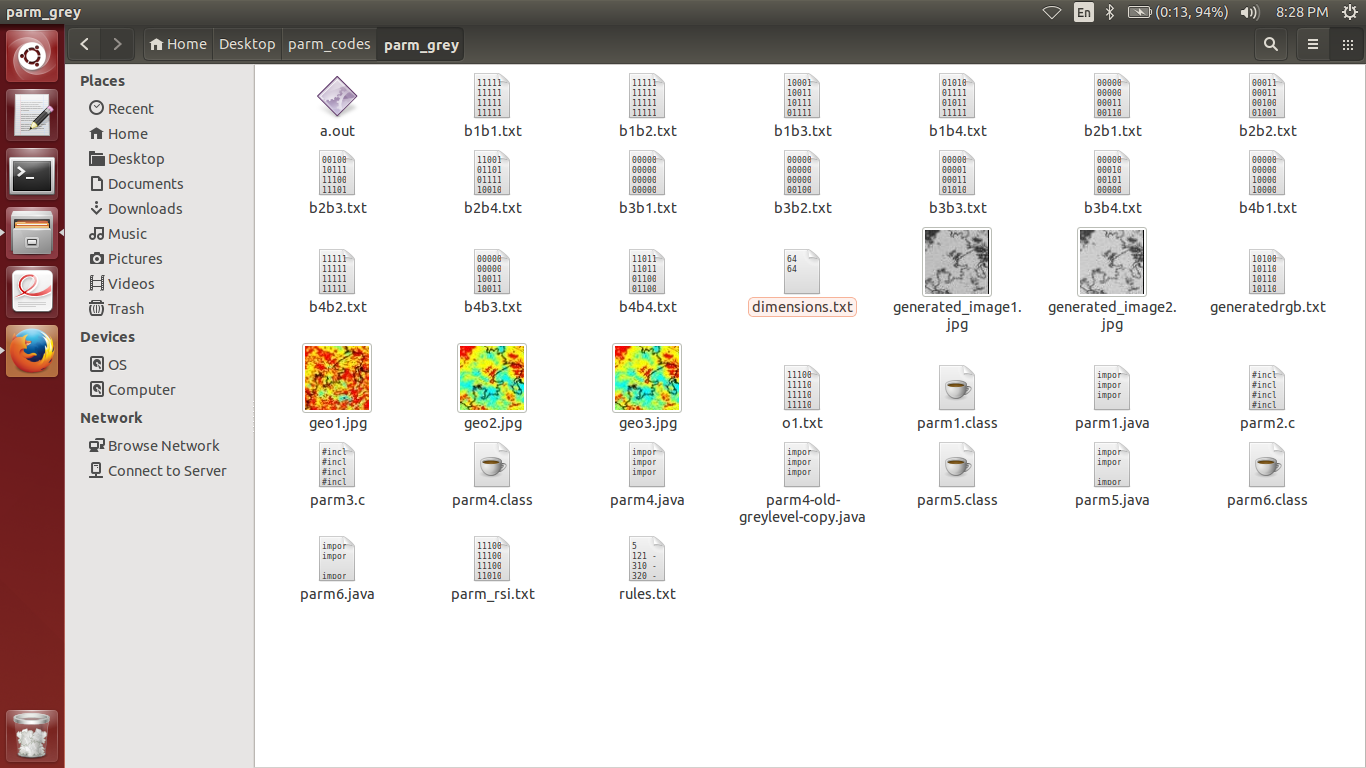
Generated\_image1.jpg

**Snapshots of file systems:**

* **RGB to RGB Mapping**

****

* **RGB to Grey Mapping**

****

**( let me know if there are any queries).**

**-Amitabh Tiwari**