Contextual image-classification using k-medoid method

Because of the influence of the noise, a target image often carries some fluctuation in the gray values, which cannot be correctly recognized by spectral classification. An important decision in image-classification is to strike a balance between spectral and spatial-recognition. The weighted combination of contextual and non-contextual (spectral) data could provide the best image classification, particularly in the presence of noise.

Let us classify a point source pollution pattern. Suppose the pollution contours form "doughnut" shapes about some center point source, as shown in the classified image in Figure 1(c). We consider the difference in distance from the center point source to a pixel and to a potential representative pixel, $|d_i-d_j|$, as the contextual part of the model formulation, and the difference between gray values, $|f_i-f_j|$, as the non-contextual (spectral) part of the formulation. The combination of spectral and spatial data can be accomplished by applying weights to each set of data. These weights range from 0 to 1, and the applied weights sum to unity.

0	0	0	0	0	1
0	2	2	2	1	0
1	1	4	4	2	0
0	2	5	3	2	0
0	2	1	2	1	1
0	0	0	1	0	0

(a) Raw image

0	0	0	0	0	1
0	2	2	2	1	0
1	1	4	4	2	0
0	2	5	3	2	0
0	2	1	2	1	1
0	0	0	1	0	0

(b) Spectral pattern-recognition (w=1)

0	0	0	0	0	0
0	1	1	1	1	0
0	1	2	2	1	0
0	1	2	2	1	0
0	1	1	1	1	0
0	0	0	0	0	0

(c) Spectral and spatial pattern-recognition ($0 \le w \le 0.5$)

Figure 1 Raw data and trial solutions

As a result, the "cost" of assigning a node i to representative pixel j is: $w|f_i-f_j|+(1-w)|d_i-d_j|$, where $0 \le w \le 1$. This cost metric is to be minimized to discern similar pollution patterns. We can adjust the weight w to obtain the best classification result, which can almost eliminate the gray value fluctuations caused by noise.

In the following example, we set the weight to be 0.5, and the threshold to judge different layers to be 1. For the convenience of programming, we first set all the cost to -1, as shown in the lower triangle of the cost matrix shown as Figure 2(b). Since the cost matrix is a symmetric across the diagonal, we only set the right top triangle of the matrix with the real values. Final classification is dependent on the cost matrix, which gives us the correct information about the similarity of the members in each group, considering both the difference of the gray values and the difference of the distances from the source

of pollution. The result of the program can be shown by several layers derived from the raw image. Every layer is pre-set to a -1 matrix, thus we can see the pollution contours of each layer clearly.

In layer 1, the least polluted pattern is discerned. It can be seen that the outer fringe of the image is identified by the MATLAB program, which sets values other than -1 in the corresponding cells. This is shown in the first frame of Figure 2(c). The second layer, representing the next level of pollution, is shown in the second frame of Figure 2(c). It consists of the area inside the outer fringe. Finally, the third layer represents the most heavily pollution 2x2 region, at the center portion of the image close to the point source. The result of this contextual image-classification using k-medoid method is also shown schematically in Figure 1(c), with the weight set at w = 0.5 in this case. We can see that it reflects the correct pollution contours.

	1	2	3	4	5	6
1	0	0	0	0	0	1
2	0	2	2	2	1	0
3	1	1	4	4	2	0
4	0	2	5	3	2	0
5	0	2	1	2	1	1
6	0	0	0	1	0	0

Figure 2 (a) Raw image data matrix

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	-1	0.31003	0.99301	0.49301	0.31003	0	0.31003	1.7071	1.4772	1.9772	1.7071	0.31003	0.49301	1.9772
2	-1	-1	0.68298	0.18298	0	0.31003	0	1.3971	1.1672	1.6672	1.3971	0	0.18298	1.6672
3	-1	-1	-1	0.5	0.68298	0.99301	0.68298	0.71409	0.48419	0.98419	0.71409	0.68298	0.5	0.98419
4	-1	-1	-1	-1	0.18298	0.49301	0.18298	1.2141	0.98419	1.4842	1.2141	0.18298	0	1.4842
5	-1	-1	-1	-1	-1	0.31003	0	1.3971	1.1672	1.6672	1.3971	0	0.18298	1.6672
6	-1	-1	-1	-1	-1	-1	0.31003	1.7071	1.4772	1.9772	1.7071	0.31003	0.49301	1.9772
7	-1	-1	-1	-1	-1	-1	-1	1.3971	1.1672	1.6672	1.3971	0	0.18298	1.6672
8	-1	-1	-1	-1	-1	-1	-1	-1	0.77009	0.27009	0	1.3971	1.2141	0.27009
9	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.5	0.77009	1.1672	0.98419	0.5
10	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.27009	1.6672	1.4842	0
11	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1.3971	1.2141	0.27009
12	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	0.18298	1.6672
13	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	1.4842
14	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
15	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
16	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

Figure 2 (b) Cost matrix

	1	2	3	4	5	6
1	0	0	0	0	0	1
2	0	-1	-1	-1	-1	0
3	1	-1	-1	-1	-1	0
4	0	-1	-1	-1	-1	0
5	0	-1	-1	-1	-1	1
6	0	0	0	1	0	0

First layer (less pollution)

	1	2	3	4	5	6
1	-1	-1	-1	-1	-1	-1
2	-1	2	2	2	1	-1
3	-1	1	-1	-1	2	-1
4	-1	2	-1	-1	2	-1
5	-1	2	1	2	1	-1
6	-1	-1	-1	-1	-1	-1

Second layer

	1	2	3	4	5	6
1	-1	-1	-1	-1	-1	-1
2	-1	-1	-1	-1	-1	-1
3	-1	-1	4	4	-1	-1
4	-1	-1	5	3	-1	-1
5	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1

Third layer (heavy pollution)

Figure 2(c) Classification results

Example MATLAB Code

```
%%MATLAB program list
clear;
data=[  0 0 0 0 0 1
        0 2 2 2 1 0
```

```
        1 1 4 4 2 0
        0 2 5 3 2 0
        0 2 1 2 1 1
        0 0 0 1 0 0    ];%%raw image data
w=0.5;%%weight
height=size(data,1);
width=size(data,2);
cost=zeros(width*height,width*height)-1;%%cost matrix
sx=(width+1)/2;%%x coordinate of source
sy=(height+1)/2;%%y coordinate of source
for i=1:width*height
   for j=i+1:width*height
      xi=mod((i-1),width)+1;
      yi=floor((i-1)/width)+1;
      xj=mod((j-1),width)+1;
      yj=floor((j-1)/width)+1;
      fij=abs(data(xi,yi)-data(xj,yj));%%gray value difference between pixels i and j: |fi-fj|
      dij=abs(sqrt((sx-xi)*(sx-xi)+(sy-yi)*(sy-yi))-sqrt((sx-xj)*(sx-xj)+(sy-yj)*(sy-
yj)));%%distance difference between pixels i and j: |di-dj|
      cost(i,j)=w*fij+(1-w)*dij;%%cost of assigning node i to representive pixel j: w*|fi-
fj|+(1-w)*|di-dj|
   end
end

TH=1;%%threshold for segmentation
k=0;
l=0;
for i=1:width*height-1
   if cost(i,i+1)~=-1
      k=k+1;
      l=1;
      xi=mod((i-1),width)+1;
      yi=floor((i-1)/width)+1;
      classx(k,l)=xi;
      classy(k,l)=yi;
      for j=i+1:width*height
         xj=mod((j-1),width)+1;
```

```
        yj=floor((j-1)/width)+1;
        if cost(i,j)~=-1&& cost(i,j)<=TH
           l=l+1;
           classx(k,l)=xj;
           classy(k,l)=yj;
           cost(:,j)=-1;
           cost(j,:)=-1;
        end
      end
   end
end
for i=1:k
   class(:,:,i)=zeros(height,width)-1;
end
for i=1:k
   for j=1:size(classx,2)
      if classx(i,j)~=0 || classy(i,j)~=0
         class(classx(i,j),classy(i,j),i)=data(classx(i,j),classy(i,j));%%image segmentated to
i group: class(:,:,i);
      end
   end
end
```