A.1. Other Classification Techniques

A.1.1. Classification based on neural network

The applications of neural networks are almost limitless but they fall into several main categories like classification, modeling, forecasting and novelty detection. Some examples of successful applications include pattern recognition, handwritten character recognition, color recognition, automatic target recognition, and image compression and others. Many researchers have compared Artificial Neural Networks (ANNs) and Logistic Regression (LR) models. They have shown that neural networks are able to make a better generalization over the traditional statistical methods such as regression techniques.

Examples of neural network architectures are single layer perceptrons, multilayer perceptrons, time-delay neural networks, Kohonen feature maps, learning vector quantization, radial basis function and Hopfield neural networks.

Backpropagation network is able to deal with various types of data and also has the ability to model a complex decision system. However, some problem domains might involve a large amount of data. Backpropagation network with
four input units and two hidden units for example require certain epochs, to create a classification or prediction model. More input units or hidden units could increase the complexity of the model and also increase its computational complexity.

The first stage of multi-backpropagation training is to minimize the complexity of data used to train the network. Minimizing the complexity means reducing the complexity of each pattern by normalizing its attributes. Normalization referred to in this study is the same as applied in relational databases where attributes are grouped into several categories to minimize the relationship between attributes. In representing an object, not all of the attributes can be represented. Instead, only necessary attributes are presented to fulfill the user's information need (Kroenke, 1997). This technique could also reduce the redundancy of data. In the second stage, each category is represented in the specialized networks.

A.1.2. Classification with fuzzy sets

Fuzzy set theory the membership grade can be taken as a value intermediate between 0 and 1 although in the normal case of set theory membership the grade can be taken only as 0 or 1. The function of the membership grade is called its "membership function" in Fuzzy theory. The user in consideration of the fuzziness will define the membership function. In remote
sensing it is often not easy to delineate the boundary between two different classes. For example, there are transitive vegetation or mixed vegetation between forest and grassland. In such cases as unclearly defined class boundaries, Fuzzy set theory can be usefully applied, in a qualitative sense [SYSJ1999].

The input to any classification scheme is a set of observations. There can be many different valid descriptions derived from these observations, each corresponding to a different application. The classification process can be thought of as one of filtering data, since once the observations are collected into a massive database, various interpretative reports can be generated for different purposes that use only part of the full set of data.

Fuzzy set theory deals with these problems of ambiguity by avoiding the constraint of associating each subsystem with a unique category. For example, instead of having to identify each plot or region as rock or fish, it is possible to balance conflicting observations in such a way that a classification like 70% rock and 30% fish is possible.

A fuzzy c-mean algorithm allows providing a fuzzy partition of the image by classifying the pixels according to several attributes from the image. It gives for each pixel a membership grade to a particular region. The algorithm uses a set A={x1, ..., xn} where x, is a vector that has "d" attributes associated with the pixel and "n" is the size of the image. For a pre specified number of regions c, the
values of fuzzy memberships are grouped together in matrix form as $U = [u_{ik}]$ (1 <= $k$ <= n, 1 <= $i$ <= c), where each row includes the membership grades for pixels belonging to region $R_i$.

Using fuzzy logic, the element $r$ belonging to the universe $R$, may be assigned a grade of membership with the membership function $m_A( r )$ to a fuzzy set $A$ is defined as:

$$A = \{ (m_A(r), r) \}, \quad r \in R, \quad m_A( r ) \in [0,1]$$

Fuzzy notion and the concept of fuzzy sets offers special advantages over classical clustering by allowing algorithms to assign each pixel a partial or distributed membership to each of the $k$ clusters. In this way fuzzy clustering procedures may yield more accurate representations of real data structures.

A.1.3. General Ideas Behind Classification

Acoustic positioning system have always been seen as expensive, complicated positioning tools and as such have been mainly for the use of deeper water offshore applications and dedicated highly trained users [NMJR 1999].

The satellite data can be reproduced in digital data from by using a scanning device and personal computer. However, for printing reason some
undesired texture would be presented in the printing image. This kind of texture has a specified Fourier Spectrum, and using some kind of filter function can eliminate it. [CTSM1995].
APPENDIX B

B.2. Software Listing

The software developed for various classification techniques are presented here. The software is developed by Java programming language. The important procedures involved for classification are given in the list.

B.2.1. Code for parallelepiped classification

```java
public static void main(String args[]) throws IOException
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Parallelepiped Classification");
    System.out.print("Enter the class number for classification 1/2/3 : ");
    String s = br.readLine();
    int cls = Integer.parseInt(s);
    Frame f = new ppcl(cls);
}

void classifyppcl(int pixels[], int xres, int yres)
{
```
long d;
long dist, MINDIST, sum;
int p, npix, k, b, c, nclass, nband, tband, tmclass;
npix = (xres*(yres+1));
nband = 3;
nclass = 3;
System.out.println("Total number of pixels : "+npix);
int mclass[][] = new int[nclass+1][nband+1]; //minclass(class,band)
int nofpix[] = new int[nclass+1]; //nofpix(class)
int band[][] = new int[npix+10][nband+1];
int modipix[] = new int[xres*(yres+1)];
int rl, rh, gl, gh, bl, bh, tk;
int rcb, gcb, bcb; //cluster boundary
int difr, difg, difb, tc;
for (c = 1; c <= nclass; c++)
    nofpix[c] = 0;

// mean value
mclass[1][1] = -16796281;
mclass[1][2] = -16769045;
mclass[1][3] = -16796281;
mclass[2][1] = -12991333;
mclass[2][2] = -12961011;
mclass[2][3] = -12991333;
mclass[3][1] = -3128525;
mclass[3][2] = -3117015;
mclass[3][3] = -3128525;
rcb = 99999999;
gcb = 99999999;
bcb = 99999999;
for(c=1;c<=nclass-1;c++)
    for(tc=c+1;tc<=nclass;tc++)
    {
        dif'fr = Math.abs(mclass[c][1]) - Math.abs(mclass[tc][1]);
        dif'fg = Math.abs(mclass[c][2]) - Math.abs(mclass[tc][2]);
        dif'fb = Math.abs(mclass[c][3]) - Math.abs(mclass[tc][3]);
        if( dif'fr < rcb) rcb = dif'fr;
        if( dif'fg < gcb) gcb = dif'fg;
        if( dif'fb < bcb) bcb = dif'fb;
    }
rcb /=2;
gcb /=2;
bcb /=2;
System.out.println("class selected is "+clas);
rl = (mclass[clas][1]) -rcb;
\[ rh = (mclass[clas][1]) + rcb; \]
\[ gl = (mclass[clas][2]) - gcb; \]
\[ gh = (mclass[clas][2]) + gcb; \]
\[ bl = (mclass[clas][3]) - bcb; \]
\[ bh = (mclass[clas][3]) + bcb; \]
// gpd = bpd;
// gbi = bbi;
k = 0;
for (p = 0; p < npix - 1; p += 3)
{
    band[k][1] = pixels[p];
    band[k][2] = pixels[p + 1];
    band[k][3] = pixels[p + 2];
    k = k + 1;
}
for (p = 0; p < npix - 1; p++)
{
    modipix[p] = pixels[p];
}
for (tk = 0; tk <= k; tk++)
{
    if (((band[tk][1] > rl) && (band[tk][1] < rh)) &&
        ((band[tk][2] > gl) && (band[tk][2] < gh)) &&
        ((band[tk][3] > gh) && (band[tk][3] < rh)))
        // do something
    }
((band[tk][3] > bl) && (band[tk][3] < bh)))
{
    modipix[tk*3] = -255;
    modipix[tk*3+1] = -255;
    modipix[tk*3+2] = -255;
    nofpix[clas]++;
}

System.out.print("pixels under class "+clas);
System.out.println(" : "+nofpix[clas]);
image=createImage(new MemoryImageSource(xres,yres, modipix,0,xres));
}

B.2.2. Code for minimum distance classification

void classifymindis(int pixels[], int xres, int yres)
{
    image=createImage(new MemoryImageSource(xres,yres, pixels,0,xres));
    long d;
    long dist, MINDIST,sum ;
    int p,npix,k,b, c, nclass,nband,tband,tmclass;
    int clas=0;
    npix = (xres*(yres+1)) ;
nband = 3;
nclass = 3;

int mclass[] = new int[nclass+1][nband+1]; //mnclass(class,band)
int nofpix[] = new int[nclass+1]; //nofpix(class)
int band[][] = new int[npix+10][nband+1];
int modipix[] = new int[xres*(yres+1)];

for (c = 1;c<=nclass;c++)
    nofpix[c] = 0;

// mean value
mclass[1][1] = -16796281;
mclass[1][2] = -16769045;
mclass[1][3] = -16796281;
mclass[2][1] = -12991333;
mclass[2][2] = -12961011;
mclass[2][3] = -12991333;
mclass[3][1] = -3128525;
mclass[3][2] = -3117015;
mclass[3][3] = -3128525;

k = 0;
for (p = 0 ;p< npix-1 ;p+=3)
{
    band[k][1] = pixels[p];
    band[k][2] = pixels[p + 1];
band[k][3] = pixels[p + 2];
k = k + 1;
}

for( p = 0;p< npix-1;p++)
{
    MINDIST = 999999999;
    for (c = 1;c<= nclass;c++)
    {
        sum = 0;
        for( b = 1;b<= nband;b++)
        {
            tband = Math.abs(band[p][b]);
            tmclass = Math.abs(mclass[c][b]);
            if (tband > tmclass)
                d = tband - tmclass;
            else
                d = tmclass - tband;
            sum = sum + d * d;
            /*if (abs(band[p][b]) > mclass[c][b])
                d = band[p][b] - mclass[c][b];
            else*/
                d = mclass[c][b] - band[p][b];
        }
    }
}
d 1st (long) Math.sqrt(Math.abs(sum));

if (dist < MINDIST)
{
    MINDIST = dist;
    clas = c;
}

nofpix[clas] = nofpix[clas] + 1;

System.out.println( "PIXELS FALL UNDER VARIOUS CLASSES" );
System.out.println("Total number of pixels "+npix);
System.out.println("CLASS 1 "+nofpix[1]);
System.out.println("CLASS 2 "+nofpix[2]);
System.out.println("CLASS 3 "+nofpix[3]);

B.2.3. Code for matching by difference based on location wise classification

public static void main(String args[]) throws IOException
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
System.out.println("Location wise classification using matching by difference ");

System.out.print("Enter the row value used in classification : ");

String s = br.readLine();

int row=Integer.parseInt(s);

System.out.print("Enter the column value used in classification : ");

s = br.readLine();

int col=Integer.parseInt(s);

Frame f=new match(row, col);
}

void classifymatch(int pixels[], int xres, int yres, int rowval, int colval)
{
    long d;
    long dist, MINDIST, sum;
    int p, k, b, npix, c, nclass, nband, tclass, tclass;
    int rowbegin, rowend, colbegin, colend;

    npix = (xres*(yres+1));

    nband = 3;

    nclass = 3;

    System.out.println("Total number of pixels : "+npix);

    int mclass[][] = new int[nclass+1][nband+1]; //mclass(class, band)
    int nofpix[] = new int[nclass+1]; //nofpix(class)
int band[][] = new int[npix+1][nband+1];
int modipix[] = new int [xres*(yres+1)];
int rl, rh, gl, gh, bl, bh, tk;
int rcb, gcb, bcb; //cluster boundary
int difr, difg, difb, tc;
int pidx, i, j;
for (c=1; c<=nclass; c++)
    nofpix[c] = 0;

// mean value
mclass[1][1] = -16796281;
mclass[1][2] = -16769045;
mclass[1][3] = -16796281;
mclass[2][1] = -12991333;
mclass[2][2] = -12961011;
mclass[2][3] = -12991333;
mclass[3][1] = -3128525;
mclass[3][2] = -3117015;
mclass[3][3] = -3128525;
rowbegin = rowval-25;
rowend = rowval+25;
colbegin = colval-25;
colend = colval+25;
if (rowbegin <= 0) rowbegin = 0;
if (rowend >= yres) rowend = yres-1;
if (colbegin <= 0) colbegin = 0;
if (colend >= xres) colend = xres-1;
k = 0;
for (i = rowbegin; i <= rowend; i++)
for (j = colbegin; j <= colend; j++)
{
    pidx = (j + xres*i);
    band[k][1] = pixels[pidx];
    band[k][2] = pixels[pidx + 1];
    band[k][3] = pixels[pidx + 2];
    k = k + 1;
}
for (p = 0; p < npix - 1; p++)
{
    modipix[p] = pixels[p];
}
double sdif, sumdif, sqrtsumdif;
int cls = 0;
sdif = 999999999;
for (c = 1; c <= nclass; c++)
{
    sumdif = 0;
for(i = rowbegin; i<=rowend;i++)

for(j = colbegin; j<=colend;j++)
{

    pidx = (j+ xres*i);

    difr = Math.abs(Math.abs(mclass[c][1]) - Math.abs(pixels[pidx]));
    difg = Math.abs(Math.abs(mclass[c][2]) - Math.abs(pixels[pidx+1]));
    difb = Math.abs(Math.abs(mclass[c][3]) - Math.abs(pixels[pidx+2]));
    sumdif += (difr+difg+difb);

}

sqrtsumdif = Math.sqrt(sumdif);

if (sqrtsumdif < sdif)
{
    
    sdif = sqrtsumdif;
    cls = c;

}

System.out.println("Class 1, Class 2 and Class 3 are the Resource names");
System.out.println("Resources abundant in this area is class ":+cls);
image=createImage(new MemoryImageSource(xres,yres, modipix.0,xres));
}