G e o X p l o r e Documentation

0. What is *GeoXplore*?

GeoXplore is a program that contains 3 different systems of which each can be trained on input data to learn to classify data from a similar source. They are: i) ***Fuzzy Clustering*** of feature vectors into a number of classes; ii) a ***Radial Basis Functional Link Network*** (RBFLN) that is an efficient and accurate new type of neural network that trains on feature vectors for which the class is known; and iii) a ***Probabilistic Neural Network*** (PNN) that trains on feature vectors with known classes to provide the maximum standardized probability density function value of the correct class. After training, any of these systems can process new unknown data feature vectors from a similar source as the training vectors and recognize the class to which they belong.

1. Running *GeoXplore*

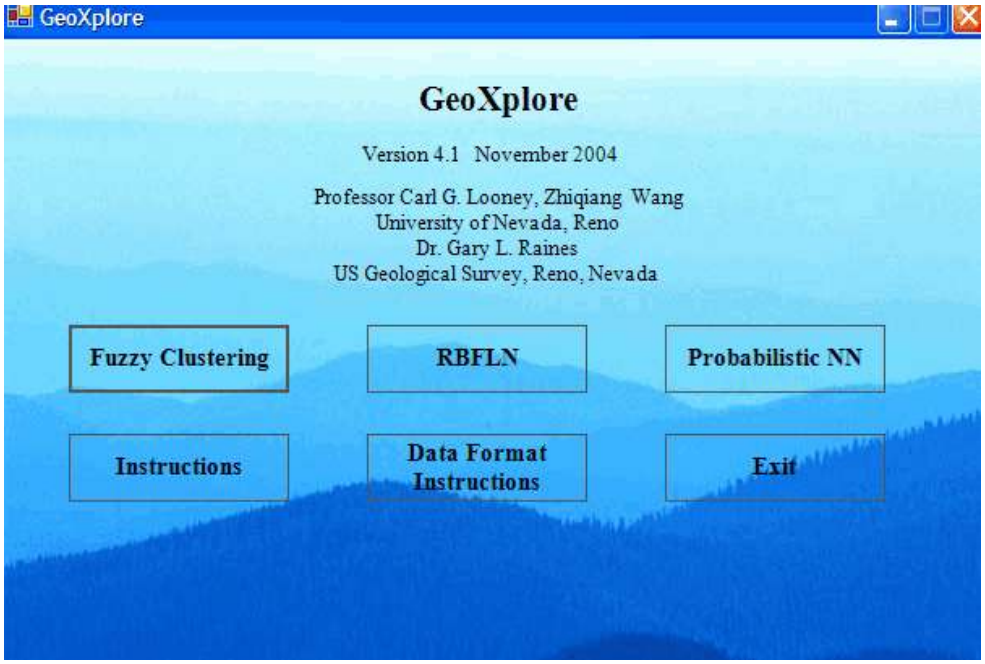
1.1 Starting *GeoXplore*. Here we assume *Geoxplore* is located in the directory *d:\path*. To start the program, either of the following mehods can be used.

Method 1. Bring up a command line window (also called a DOS window) and change to the directory *d:\path*. Then type the underlined part below to start the program.

d:\path> GeoXplore <ENTER>

Method 2. Click ***Start*** at the bottom left of the screen, then click ***Run***, then ***Browse***, and then go to the directory (designated by *d:\path* here) and click on ***GeoXplore.exe*** or just ***GeoXplore*** (sometimes the suffix ***.exe*** will not show).

Figure 1. The first and main window of *GeoXplore*.



1.2 Choosing a System for Training. The program is now running and the window shown in Figure 3 is visible on the computer screen. The 3 different systems are described below. Each one corresponds to a button in the top row of the figure. They all give approximately the same results, but one may do better or worse than the others depending on the data and the user given parameters during the training. They can be used to check each other and the best results used.

A. Fuzzy Clustering - this system clusters the input vectors of features (attributes) in a file of feature vectors into clusters. The classes need not be known for this training that decides the classes from the data. Each cluster represents a particular class. A center, or prototype vector is computed for each class by a new fuzzy clustering method that is immune to outliers. When an unknown feature vector is put into the trained system, it is classified as being in the class to whose prototype it is closest. Goodness of classification values are given at that time.

B. Radial Basis Functional Link Net (RBFLN) - this special new type of neural network trains a set of parameters in a network to yield the correct class number as an output value when a feature vector is input. The training is done on any of the files of input vectors on which the fuzzy clustering system trains, but the classes for the training data must be known. This system is like a radial basis function neural network except it has extra lines from the input nodes to the output nodes to model the linear parts of a maping with a linear part rather than modeling both linear and nonlinear parts with a nonlinear mapping. Thus it is faster and more accurate.

C. Probabilistic Neural Network (PNN) - This is a modification of the standard probabilistic neural network that puts a Gaussian probability density function over each training feature vector and sums these with proper standardization for form a general probability density function for each cluster. Our modification reduces this set of Gaussians by eliminating feature vectors as Gaussian centers if they are very close to another feature vector. Thus it is faster. This also reduces the extraneous error that builds up when lots of feature vectors in the same class are close together (too many Gaussians).

1.3 The Training (Input) Data File Format. The format for all input data files is described below with comments that do not appear in the data files. Note that the data are in comma delimited rows with no spaces allowed anywhere. There is a carriage return at the end of each row (be sure there is no space before the carriage return).

Training File Row Data	**Comment on Training Data**
N	#no. feature components in input vector (integer)
M	#dummy number, but necessary (integer)
J	#always set to 1 (integer)
Q	#no. vectors in this file (integer)
1,c1,d1,x11,x21,...,xN1,D1	#first column is vector number (integer)
2,c2,d2,x12,x22,...,xN2,D2	#second column is class number (integer)
: : :	#third column d values are dummies, but necessary
Q,cQ,dQ,x1Q,x2Q,...,xNQ,DQ	#fourth - fourth+N columns are N features (float)
	#last column is a dummy column (necessary)

As examples, consider the training data in the two files below.

Training File 1

```
2
18
1
20
1,1,1.1,1,1,0
2,1,1.1,2,1,0
3,2,1.1,2,5,0
4,2,1.1,3,5,0
5,3,1.1,5,2,0
6,3,1.1,6,2,0
7,4,1.1,5,8,0
8,4,1.1,6,8,0
9,5,1.1,8,5,0
10,5,1.1,9,5,0
11,1,1.1,1,2,0
12,1,1.1,2,2,0
13,2,1.1,2,6,0
14,2,1.1,3,6,0
15,3,1.1,5,3,0
16,3,1.1,6,3,0
17,4,1.1,5,9,0
18,4,1.1,6,9,0
19,5,1.1,8,5,0
20,5,1.1,9,5,0
```

Training File 2

```
2
20
1
32
1,1,2,0.244898,0.902655,0.100000
2,0,4,0.346939,0.902655,0.100000
3,0,8,0.244898,0.893805,0.100000
4,0,9,0.183673,0.893805,0.100000
5,1,11,0.183673,0.920354,0.900000
6,1,19,0.530612,0.893805,0.900000
7,0,22,0.244898,0.884956,0.100000
8,0,24,0.387755,0.893805,0.100000
9,1,26,0.0,0.902655,0.900000
10,1,28,0.755102,0.902655,0.900000
11,1,29,0.755102,0.893805,0.900000
12,0,35,0.714286,0.893805,0.100000
13,1,36,0.265306,0.911504,0.900000
14,0,37,0.346939,0.893805,0.100000
15,1,43,0.265306,0.902655,0.900000
```

16,1,55,0.755102,0.920354,0.900000
17,1,60,0.755102,0.911504,0.900000
18,1,61,0.755102,0.982301,0.900000
19,1,71,0.755102,0.938053,0.900000
20,1,72,0.755102,0.964602,0.900000
21,1,74,0.755102,1.0,0.900000
22,0,83,0.530612,0.884956,0.100000
23,0,87,0.163265,0.902655,0.100000
24,1,90,0.530612,0.955752,0.900000
25,1,92,0.755102,0.955752,0.900000
26,1,108,0.571429,0.920354,0.900000
27,0,118,0.571429,0.884956,0.100000
28,0,128,0.77551,0.893805,0.100000
29,1,137,0.897959,0.893805,1.000000
30,0,154,0.040816,0.893805,0.100000
31,0,159,0.183673,1.0,0.100000
32,0,179,0.040816,0.884956,0.100000

1.4 The Classification (Input) Data File Format. The format of the classification input data files (data to be classified) is similar, but the second column here has no meaning because the classes are not known and thus no training can be done on these files (fuzzy clustering could be done because it does not use the given classes and thus is a check on the accuracy of the given class numbers).

Classification File Data	**Comment on Classification Data**
N	#no. feature components in input vector (integer)
M	#dummy number, but necessary (integer)
J	#always set to 1 (integer)
Q	#no. vectors in this file (integer)
1,d1,dd1,x11,x21,...,xN1,D1	#first column is vector number (integer)
2,d2,dd2,x12,x22,...,xN2,D2	#second and third columns are dummies
: : :	#fourth - fourth+N columns are N features (float)
Q,dQ,ddQ,x1Q,x2Q,...,xNQ,DQ	#last column is a dummy column (for future use)

An example of a training file is given next.

2
22
1
40
1,0,1722.625,0.244898,0.911504,0
2,1,4736.0625,0.244898,0.902655,0
3,0,681.375,0.183673,0.902655,0
4,0,254.0,0.346939,0.902655,0
5,0,114.25,0.387755,0.902655,0
6,0,227.4375,0.020408,0.902655,0
7,0,89.125,0.020408,0.911504,0
8,0,14485.6875,0.244898,0.893805,0
9,0,1926.375,0.183673,0.893805,0

10,0,330.4375,0.183673,0.911504,0
11,1,179.6875,0.183673,0.920354,0
12,0,119.1875,0.183673,0.929204,0
13,0,58.25,0.183673,0.938053,0
14,0,47.5625,0.183673,0.946903,0
15,0,119.1875,0.346939,0.911504,0
16,0,22.125,0.183673,0.955752,0
17,0,648.75,0.244898,0.920354,0
18,0,219.1875,0.530612,0.902655,0
19,2,506.1875,0.530612,0.893805,0
20,0,309.0,0.244898,0.929204,0
21,0,176.0,0.244898,0.938053,0
22,0,1981.0,0.244898,0.884956,0
23,0,207.625,0.387755,0.884956,0
24,0,1185.0625,0.387755,0.893805,0
25,0,1143.75,0.0,0.893805,0
26,1,323.0625,0.0,0.902655,0
27,0,144.4375,0.0,0.911504,0
28,3,1121.3125,0.755102,0.902655,0
29,1,2610.8125,0.755102,0.893805,0
30,0,0.25,0.102041,0.911504,0
31,0,69.0,0.346939,0.920354,0
32,0,17.75,0.183673,0.964602,0
33,0,16.125,0.183673,0.973451,0
34,0,17.1875,0.387755,0.911504,0
35,0,2084.5,0.714286,0.893805,0
36,3,138.5,0.265306,0.911504,0
37,0,1241.8125,0.346939,0.893805,0
38,0,31.875,0.346939,0.929204,0
39,0,5.375,0.183673,0.99115,0
40,0,126.0,0.244898,0.946903,0

1.5. The Classification (Output) Results Data Files. These are the results of classification of unknown feature data with any one of the three trained systems. This data is written to an output file.

Data	**Comment on Results Data**
1,c1,g1	#first column is vector number
2,c2,g2	#second column is the class number
: :	#third column is the belief, or goodness, of the classification
Q,cQ,gQ	# into the given class

2. Using the Fuzzy Clustering System

2.1 Training the Fuzzy Clustering System. Here we click the Fuzzy Clustering button in the window of Figure 1. The new window shown below in Figure 2 comes up on the click. The top button gives brief instructions. The first thing to do is to train the fuzzy

clustering system on a training (input) data file. We will go through the steps here. Click the ***Train Fuzzy Clustering*** button now to bring up the window of Figure 3.

Figure 2. The first *Fuzzy Clustering* window.

We now click the ***Load File*** button and select the path and filename of the training file to use to train this system. A window will come up and allow the selection of a data file for training. Its name has the format **.dta* . Either click twice on the file name or click once on it and then click ***Open*** on the lower right. That file will be read in and closed.

Next, click the ***Save As*** button to provide a filename for the file that will hold the training results, that is, the parameters that were learned during training. In this case, this file holds the fuzzy centers of the clusters (classes) and the sigma values for the Gaussians. These files have the name format **.cen* to indicate that the centers of the clusters (and other information) are stored here.

We are now ready to train, so click the ***Run*** button at the bottom of the window. The window of Figure 4 will now come up and show the results of preliminary training with a large number of initial seeds as preliminary cluster centers, of which some clusters will be empty and others need to be eliminated if they are too close to other seeds. The number of clusters may still be too large, so we may neet to eliminate more of the current cluster centers.

Figure 3. The *Fuzzy Clustering Training* window.

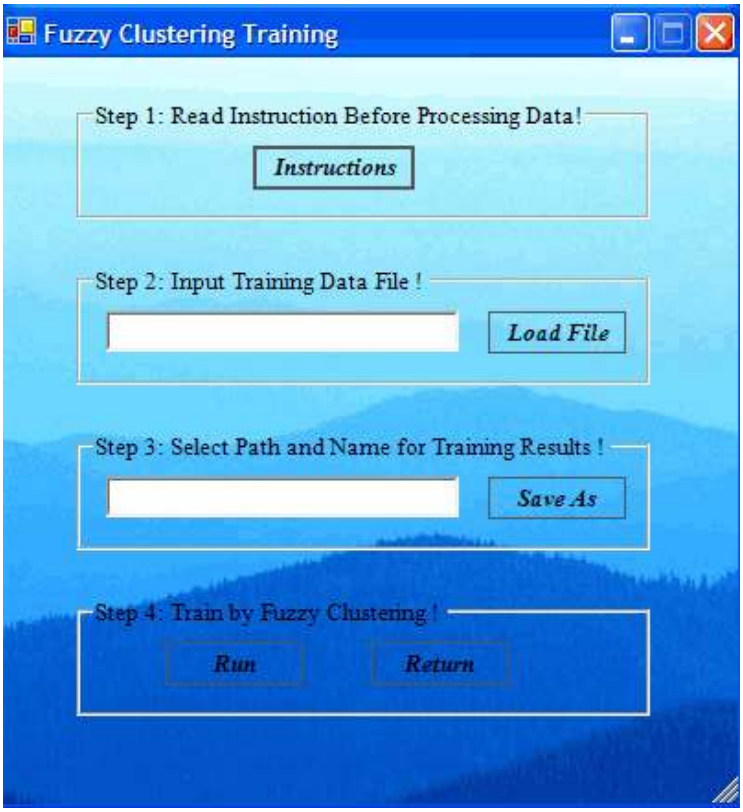
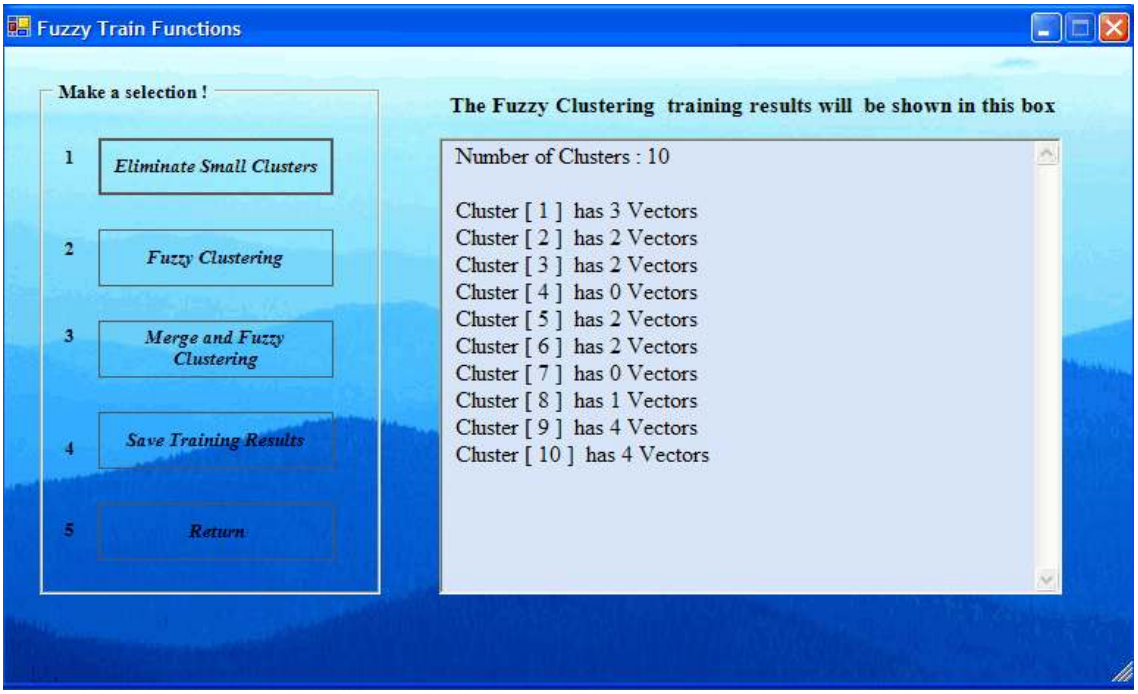
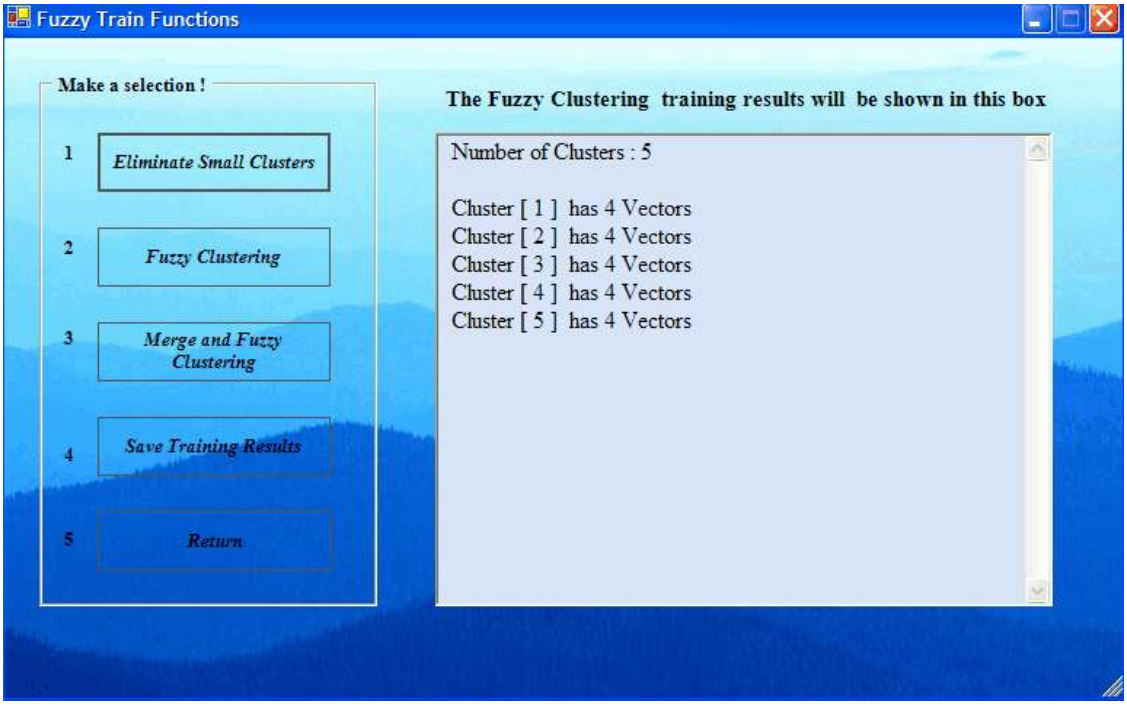


Figure 4. The preliminary results of clustering.



Click the ***Eliminate Small Clusters*** button next and then click it again. Repeat until the number of clusters does not change on a click. The results of this are shown in Figure 5. We may not know how many clusters K there should be, but we will use the Xie-Bene clustering validity to tell us what value of K gives the best clustering.

Figure 5. The results of eliminating small clusters.



Now click the ***Fuzzy Clustering*** button first to get the results in the window of Figure 6. Notice the clustering validity number. The greater this number, the greater the clustering goodness. While it is usually a good way of determining the best number K of classes, it does not always provide the best classification number, so if the we know how many clusters there should be, then we should use that number for K. For example, if a test is either benign or malignant, then the correct number of clusters should be K = 2. However, if some data does not fall into either, then there may be K = 3 classes: benign, malignant and indeterminate. Classification is an intuitive art as well as a science.

Next click the ***Merge and Fuzzy Clustering*** button of Figure 6 to get the window shown in Figure 7. Here a default parameter of 0.4 is shown. This is the proportion of the average distance between cluster centers to use as a threshold for merging two clusters. If the average distance between pairs of cluster centers is a, then if any distance between a pair of cluster centers is less than $0.4a$, the clusters will be merged into one cluster and fuzzy clustering is done again with one less cluster center. First use the default 0.4. If the number of clusters reduces, then repeat with the same parameter. Otherwise, increase the parameter from 0.4 (0.4, 0.5, 0.6,... is usually good). Note the clustering validity value for

each value of the number of clusters K. The K value with the highest clustering validity is usually the best unless there is other information to override it. Stop at K = 2 clusters and examine the clustering validity values, or possibly stop when K is the known value. For our data, when we stop at K = 2, the result is that in Figure 8.

Figure 6. The results of fuzzy clustering with clustering validity.

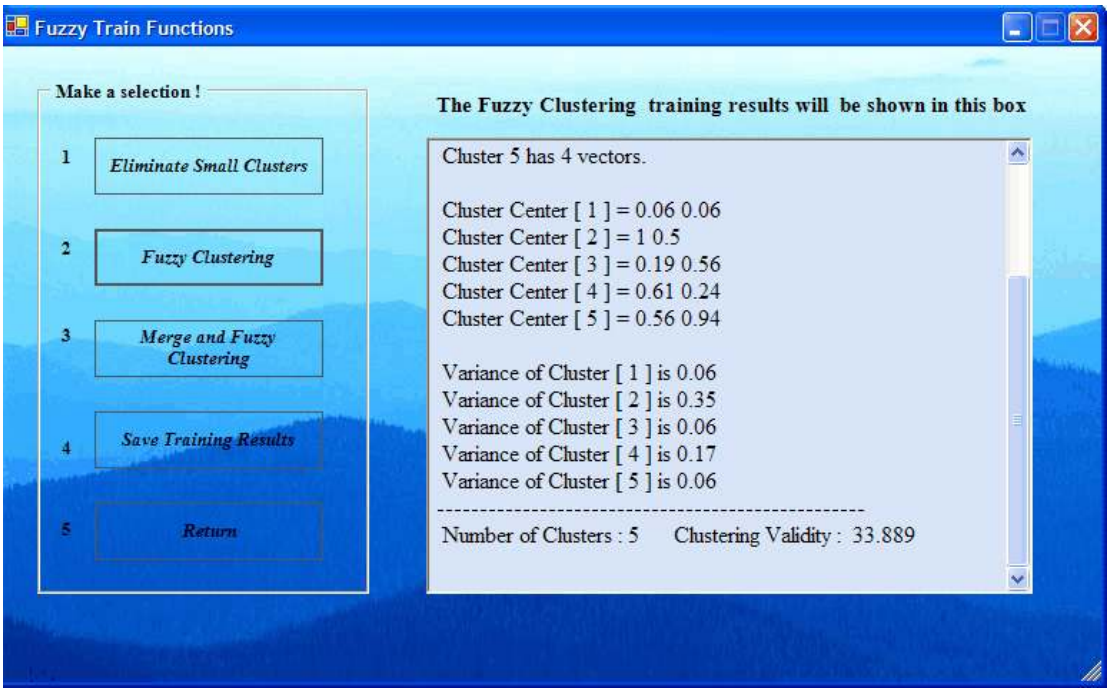


Figure 7. The fuzzy merging parameter.

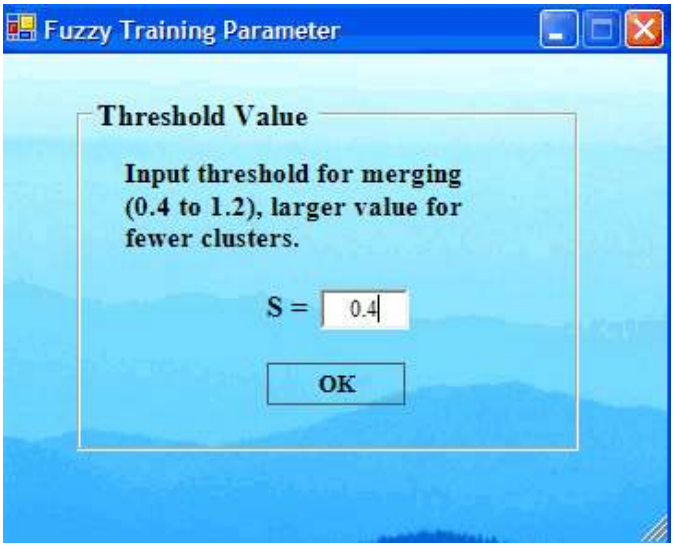


Figure 8. The final results of merging and fuzzy clustering.

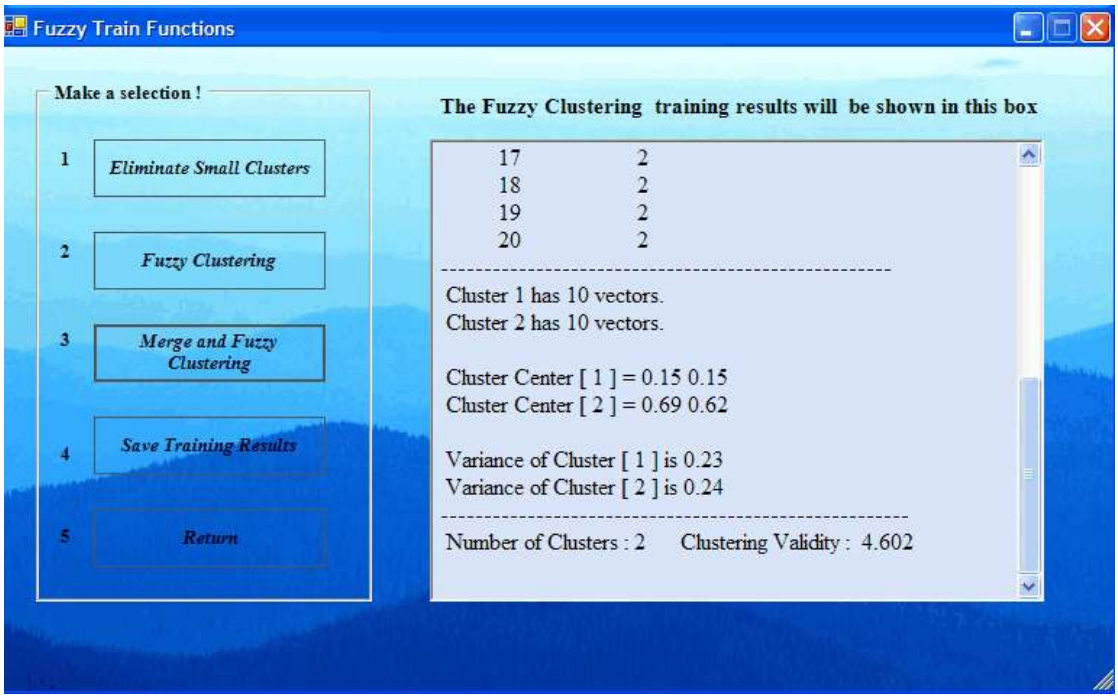


Figure 8 shows the results and the clustering validity is higher than for K = 3 or 4, but it is not as high as for K = 5, which is the correct number of clusters for this synthetic data (see Figure 6 where the clustering validity is 33.889 for 5 clusters) that was designed with 5 good clusters. After returning and training again with K = 5, we click on the ***Save Training Results*** button in Figure 8 to save the training parameters in the file (**.cen*) whose name we provided before. We will use these parameters next when we classify unknown feature data.

2.2 Classifying Unknown Data with the Trained Fuzzy Clustering System. Now we click ***Return*** and then click another ***Return*** to return to the window of Figure 2. This time we click the ***Do Fuzzy Classification*** button to bring up the window shown in Figure 9. Here we first load the file of training parameters that we saved as a **.cen* file. Then we load in the data file (**.dta*) of unknown feature vectors to be recognized as belonging to a particular class. The last file we must select is the file for saving the results of the classification. This file must have the name of the form **.fuz* and can be sent to the printer or examined by an editor or use of *Microsoft Excel* spreadsheet program (the input data files can also be viewed or entered by use of *MS Excel*. The overall goodness of classification numbers are also shown in Figure 10.

We click ***Run*** at the bottom to let the fuzzy clustering system recognize the input feature data by assigning it to the appropriate class. Figure 10 shows the results of this click. The results file (**.fuz*) is shown in Figure 11 where in each row the first number is the vector number, the next is the class, and the third is the fuzzy belief of classification.

Figure 9. Loading the training parameters and the data for classification.

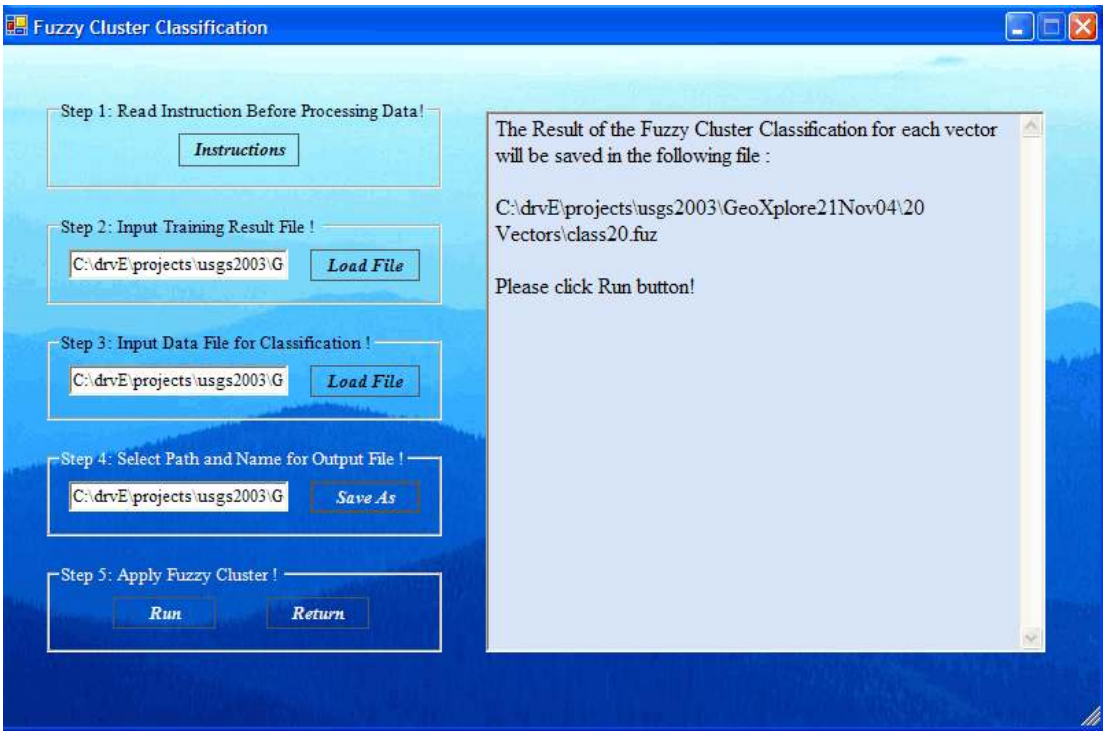


Figure 10. The results of running the fuzzy clustering classification function.

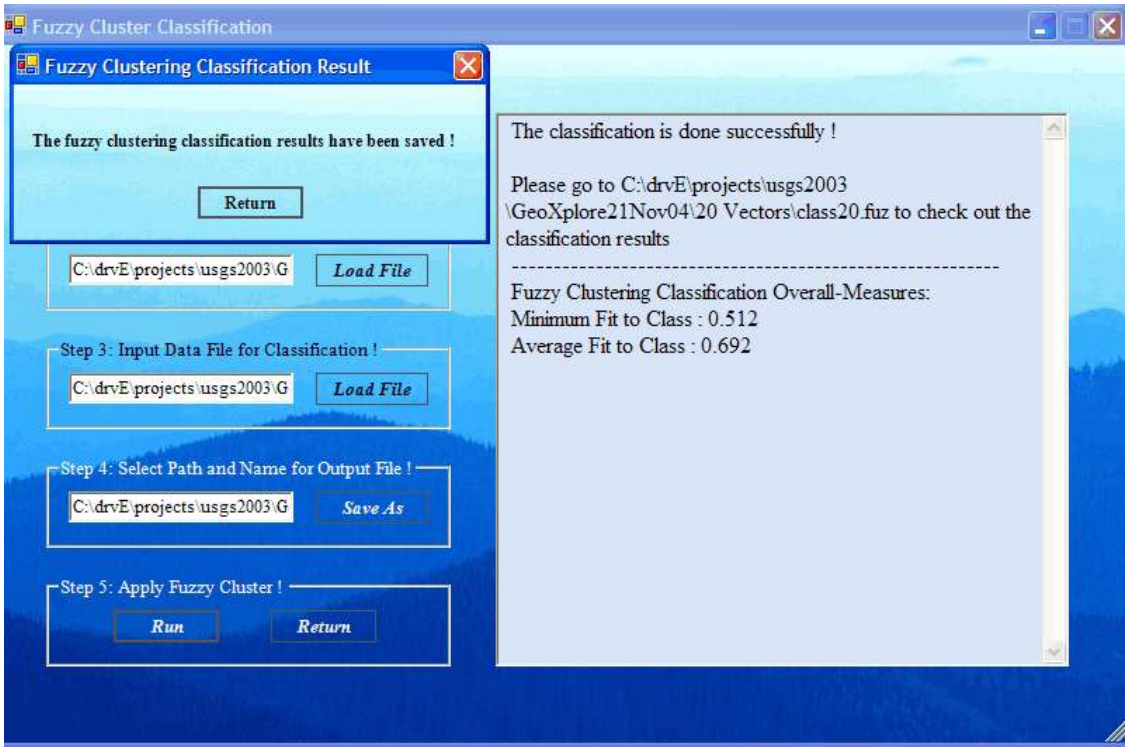


Figure 11. The classification results of the fuzzy clustering system.

1,1,0.976
2,1,0.973
3,4,0.966
4,4,0.963
5,3,0.938
6,3,0.956
7,5,0.968
8,5,0.967
9,2,0.953
10,2,1.000
11,1,0.973
12,1,0.968
13,4,0.966
14,4,0.963
15,3,0.952
16,3,0.992
17,5,0.973
18,5,0.972
19,2,0.953
20,2,1.000

Now we click **Return** to get back to the window of Figure 2, and click **Return** again to go back to the window of Figure 1. From there we can select **RBFLN** or **PNN** to train one of those systems and then use if to recognize new unknown feature data.

2.3 When Fuzzy Clustering is a Good Method. When the classes form groups that fit inside circles of a fixed radius, then fuzzy clustering is an excellent method that is somewhat robust by being immune to outliers.

However, in some cases the classes may be elongated in shape or a shape that curves like a crescent or other irregular object. In this case it is better to cluster the feature vectors into many small circular clusters of which each will be a subclass of a class of feature vectors. In such cases it is often simpler to use the RBFLN or the PNN.

3. The Radial Basis Functional Link Net (RBFLN)

3.1 Training the the RBFLN System. When GeoXplore is run as described in Section 1.1, the window in Figure 1 comes up. We click on the ***RBFLN*** button in the figure to obtain the window shown in Figure 12 below.

Figure 12. The first *RBFLN* window.

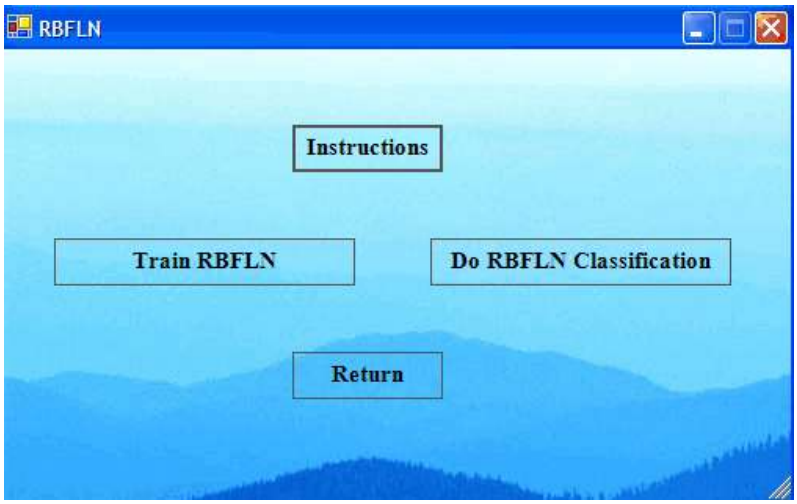
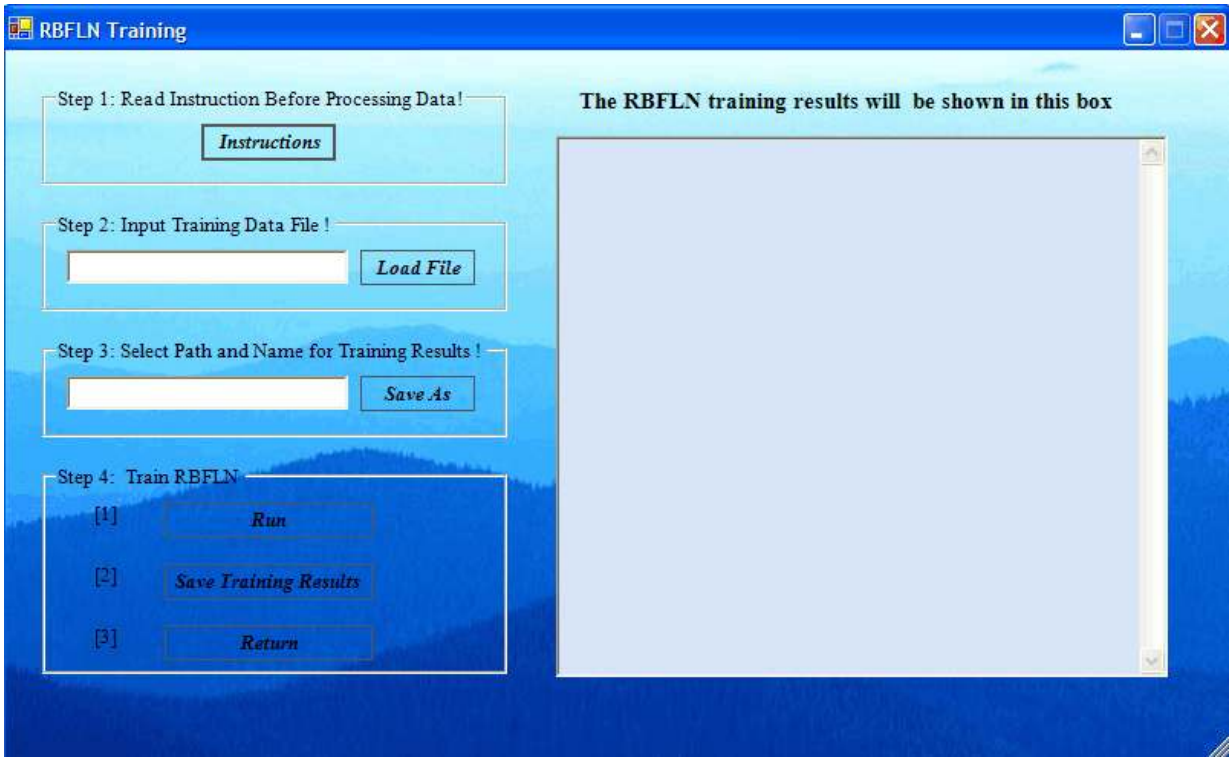


Figure 13. The *Train RBFLN* window.

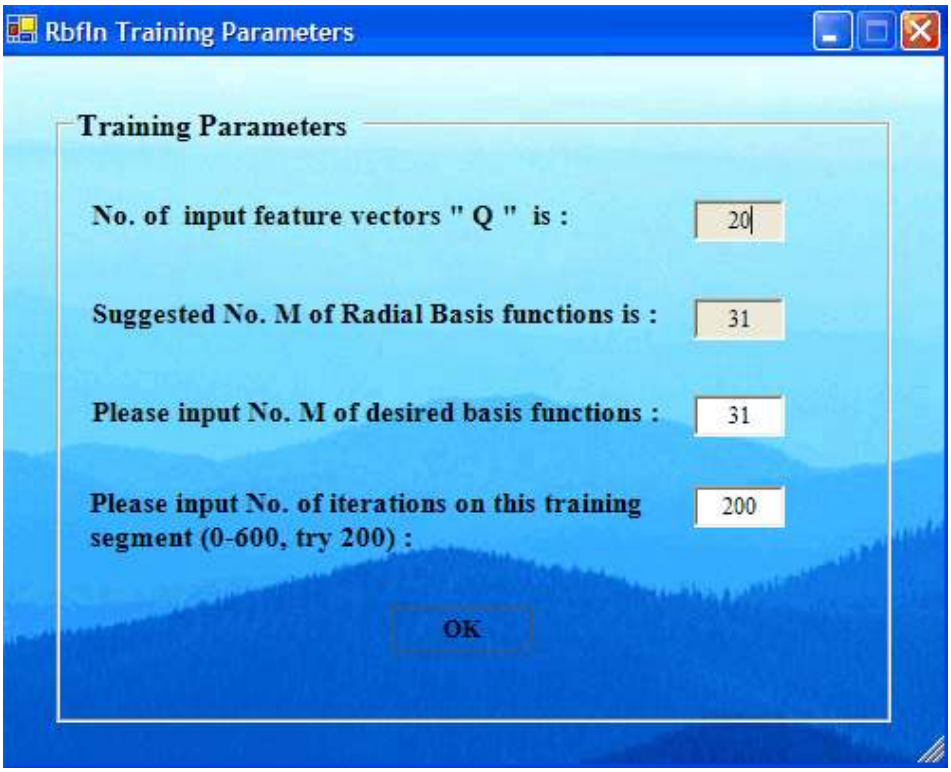


We now click the ***Train RBFLN*** button of Figure 12 to bring up the window in Figure 13. Next, we click the ***Load File*** button and select the path and filename of the training file to use to train this system. A window will come up and allow the selection of a data file for training whose name has the format **.dta* . Either click twice on the file name or click once on it and then click ***Open*** on the lower right. That file will be read in and closed.

Next, click the ***Save As*** button to provide a filename for the file that will hold the training results, that is, the parameters that were learned during training. In this case, this file holds the fuzzy centers and the sigma values for the Gaussian radial basis functions, the weights on the lines from the Gaussians to the output nodes, and the weights on the extra lines from the input nodes to the output nodes that bypass the Gaussians at the middle layer of neurodes. These files have the name format **.par* to indicate that the parameters for the network are stored here.

We are now ready to train, so click the ***Run*** button at the bottom of the window of Figure 13. The window of Figure 14 comes up with the run parameters shown. The top two parameters are for the users information and can not be changed. The third value can be changed, but it should not vary too much from the suggested value given in the second value. For a larger number M of radial basis functions, the neural network will slower but more accurate up to a point. However, an M value that is too large will have extraneous error building up. For important training, the user may need to experiment by using a smaller M and a larger M and noting the overall measures of training goodness, where a higher belief between 0 and 1 is better (1.0 is perfect).

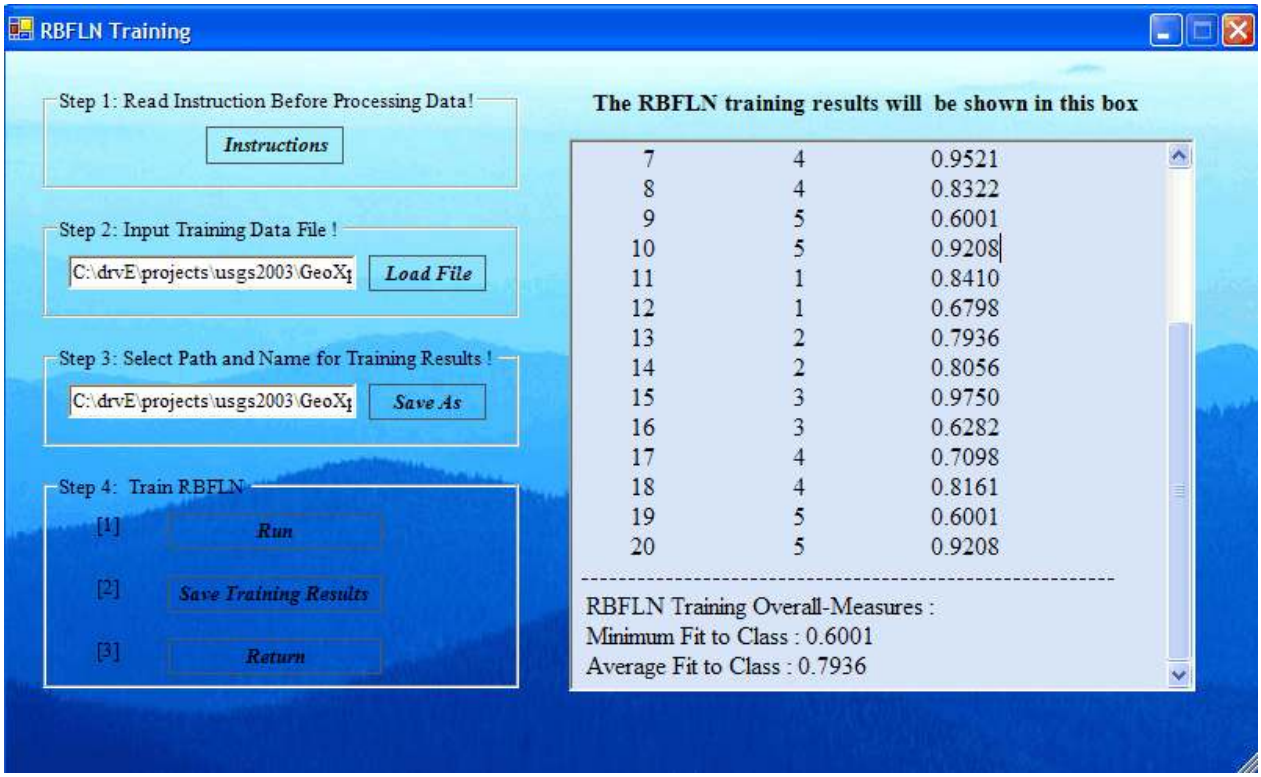
Figure 14. The run parameter window for RBFLN.



The number of iterations also affects the quality of training. If the training data set is small, then 100 iterations will be sufficient. The default of 200 is a good trade-off between under training and over training (where the noise is learned), but for larger training data sets more may be required. The user can use, say 100, then 200 and then 400 and examine the results.

Click the ***OK*** button in Figure 14 next for the training with the number of iterations and the number of radial basis functions selected. This will run the training and bring up the results shown in Figure 15. The overall goodness of training measures at the bottom of the textbox are: i) mininum fit to class; and ii) average fit to class. Each fit value is a fuzzy belief (truth) of the classfication.

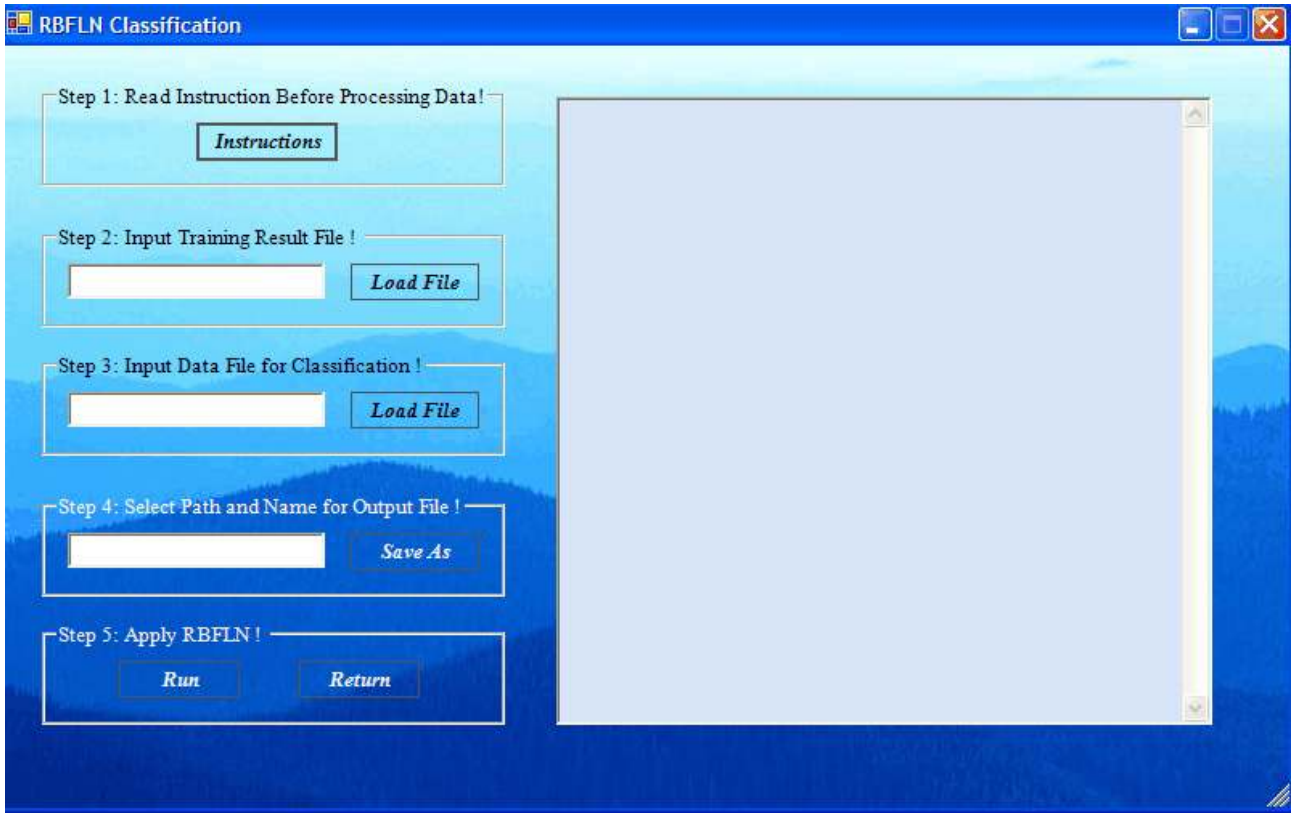
Figure 15. The results of training the RBFLN.



The training may be repeated multiple times with different run parameters (Figure 14) until the overall measures of goodness of training are near optimal. At this point the user should save the training parameters by clicking on the ***Save Training Results*** button in Figure 15. At this point, the user should click the ***Return*** buttons to go back to the window shown in Figure 12. Unknown feature vector data from similar sources can now be classified by use of the the trained RBFLN.

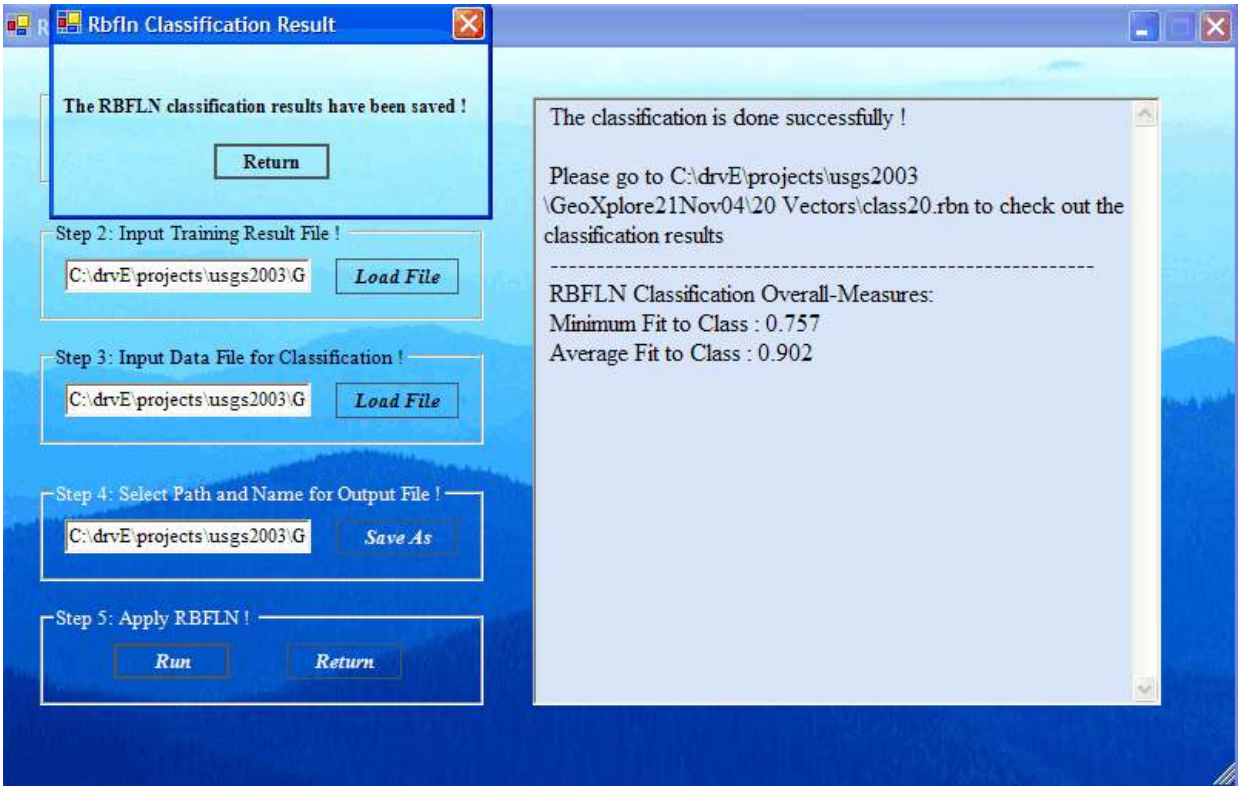
2.2 Classifying Unknown Data with the Trained RBFLN System. To start the classifying process, click the ***Do RBFLN Classification*** button of Figure 12. This will bring up the RBFLN classification window as shown in Figure 16. Now we click the top ***Load File*** button and select the file of parameters that was saved during the training, which has a name of the form **.par*. Next we click the lower ***Load File*** button and select the file of data to be classified whose name has the form **.dta.* Now we must provide a file name for the file where the results of the classification will be stored in a file with name format **.rbn*, so click the ***Save As*** button and provide a file name. Then click the ***Run*** button.

Figure 16. Loading the training parameters and data for RBFLN classification.



The RBFLN network executes with the loaded training parameters and processes the classification input data into a class for each input unknown feature vector. The results are written to the file with name format **.rbn* selected above and can be viewed or printed out with an editor program. The window now appears like the window shown in Figure 17.

Figure 17. The results of running the RBFLN classification function.

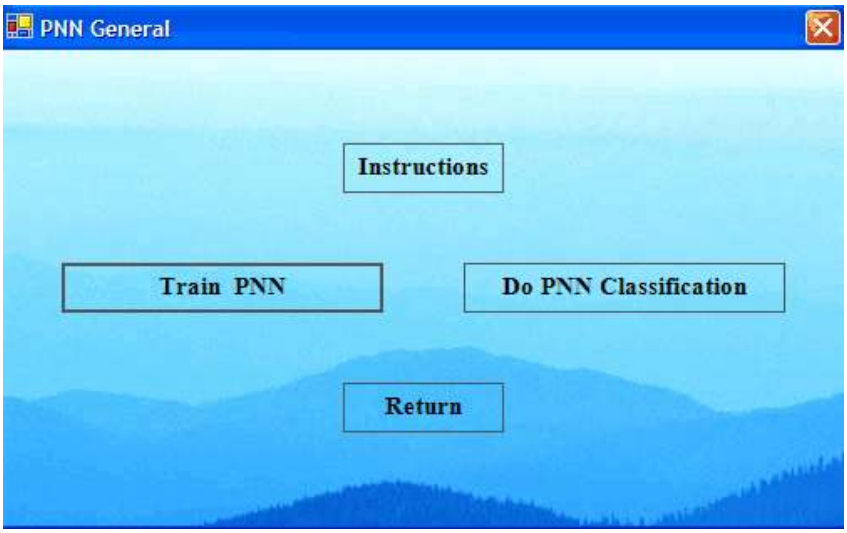


The goodness of fit (classification) numbers are shown at the bottom of the textbox in the form of a minimum goodness of fit and an average goodness of fit, both of which are fuzzy beliefs in the classification of this data with the training parameters loaded previously. The classification results file will have the same format as in Figure 11.

4. The Probabilistic Neural Network (PNN)

4.1 Training the the PNN System. When GeoXplore is run as described in Section 1.1, the window in Figure 1 comes up. We click on the ***Probabilistic NN*** button in the figure to obtain the window shown in Figure 12 below that is similar to the window for the RBFLN shown in Figure 18.

Figure 18. The first *PNN* window.



We now click the ***Train PNN*** button to bring up the window in Figure 19. Next, we click the ***Load File*** button and select the path and filename of the training file to use to train this system. A window will come up and allow the selection of a data file for training whose name has the format **.dta* . Either click twice on the file name or click once on it and then click ***Open*** on the lower right. That file will be read in and closed.

Next, click the ***Save As*** button to provide a filename for the file that will hold the training results, that is, the parameters that were learned during training. In this case, this file holds the fuzzy centers and the sigma values for the Gaussian probability density functions. These files have the name format **.prb* to indicate that the parameters for the PNN network are stored here.

We are now ready to train, so click the ***Run*** button at the bottom of the window. The window of Figure 20 will then come up with a proportion parameter as shown. This parameter is the proportion of the average distance between training feature vectors to use as a threshold. If any vector is closer to another than this proportion of the average distance, it is eliminated as a center of Gaussians to be summed to construct a probability density function (*pdf*) for a class. A larger threshold eliminates more vectors and increases computational speed, but it is a trade-off between speed and accuracy up to a point. A small proportion for a large number of training vectors uses too many Gaussians and causes extraneous error to build up during computation of the *pdfs*. The user should experiment with this.

Figure 19. The *Train PNN* window.

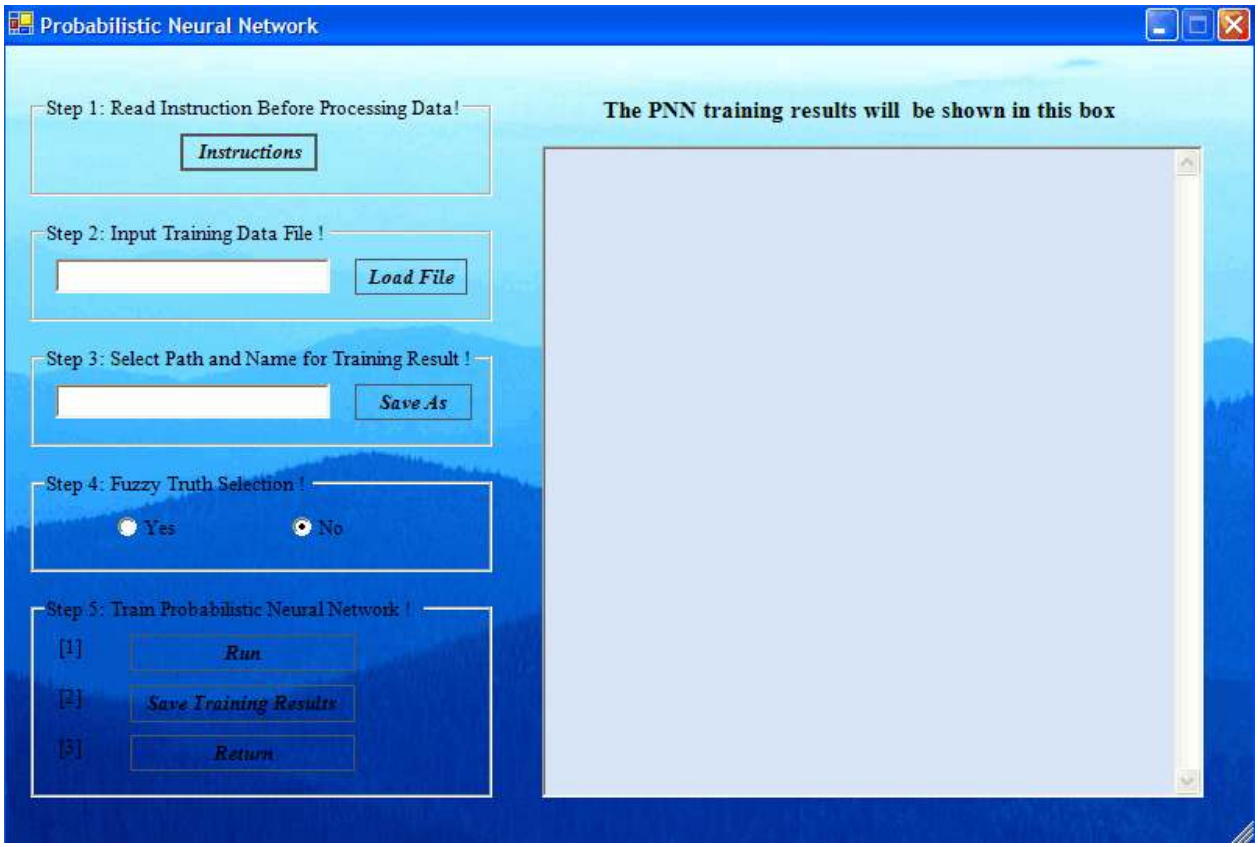
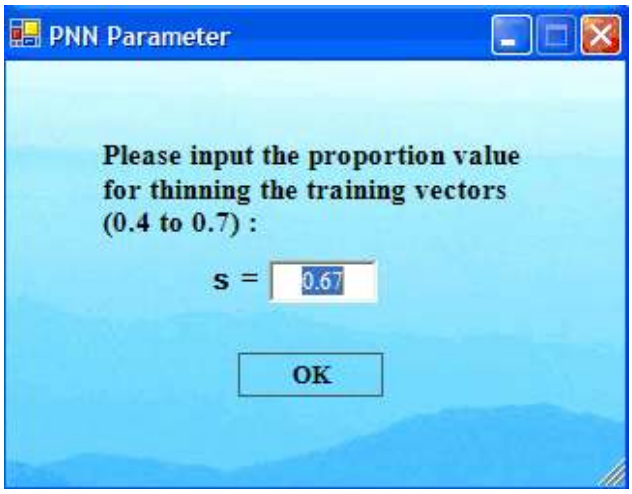
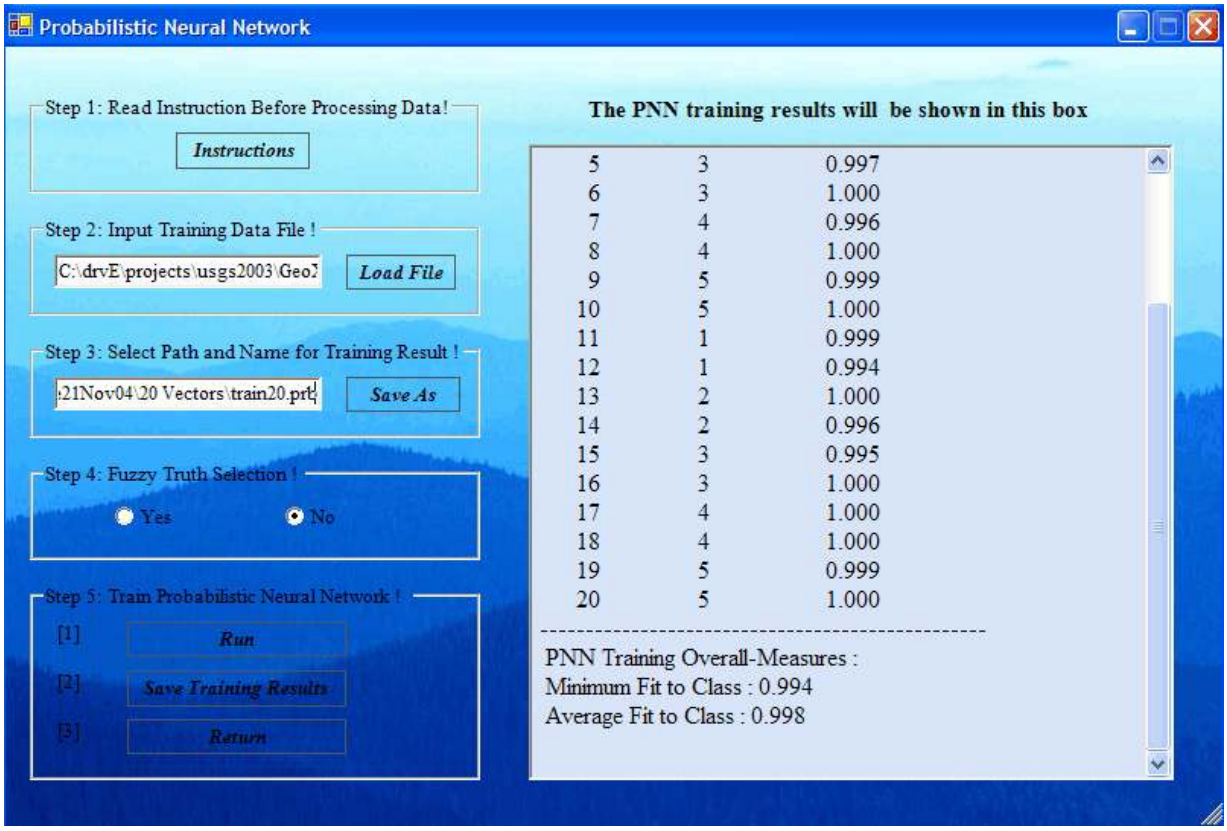


Figure 20. The run parameter window for PNN.



Click the ***OK*** button of Figure 20 for the training with the proportion selected. This will run the training and build the *pdfs* as sums of Gaussians. The results are shown in Figure 21. The overall goodness of training measures are at the bottom of the textbox and are: i) mininum fit to class; and ii) average fit to class. Each fit value is a standardized pdf value that serves as a fuzzy belief (truth) of the assigned classification by the trained PNN.

Figure 21. The results of training the *PNN*.

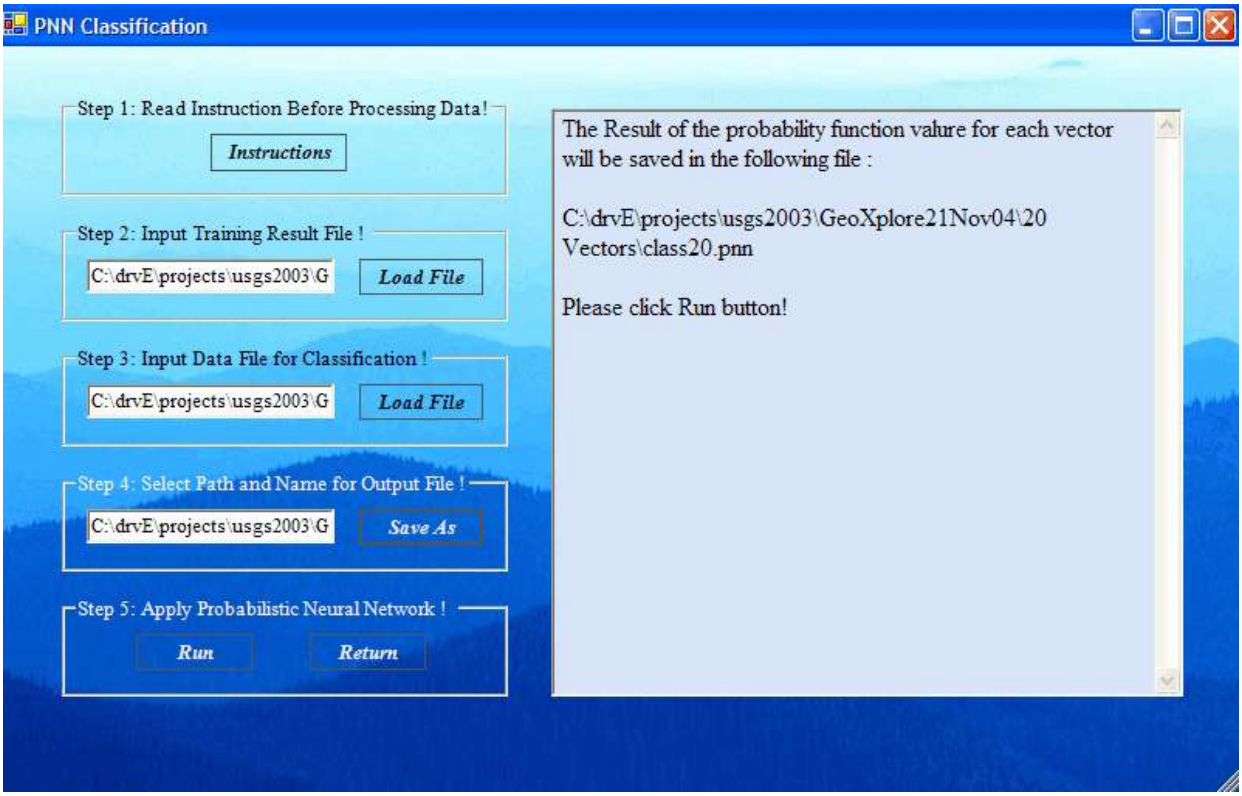


The training may be repeated multiple times with different proportions until the overall measures of goodness of training are near optimal. These measures are shown at the bottom of the textbox displayed in Figure 21. When the user is ready to accept this training, then the training parameters can be saved by clicking on the ***Save Training Results*** button. Click the ***Return*** buttons of Figure 21 and 19 to go back to the window shown in Figure 18. Unknown feature vector data from similar sources can now be classified by use of the the trained PNN.

4.2 Classifying Unknown Data with the Trained PNN System. To start the classifying process, click the ***Do PNN Classification*** button of Figure 18. This will bring up the PNN classification window as shown in Figure 22. Now we click the top ***Load File*** button and select the file of parameters that was saved during the training, which has a name of the form **.prb*. Next we click

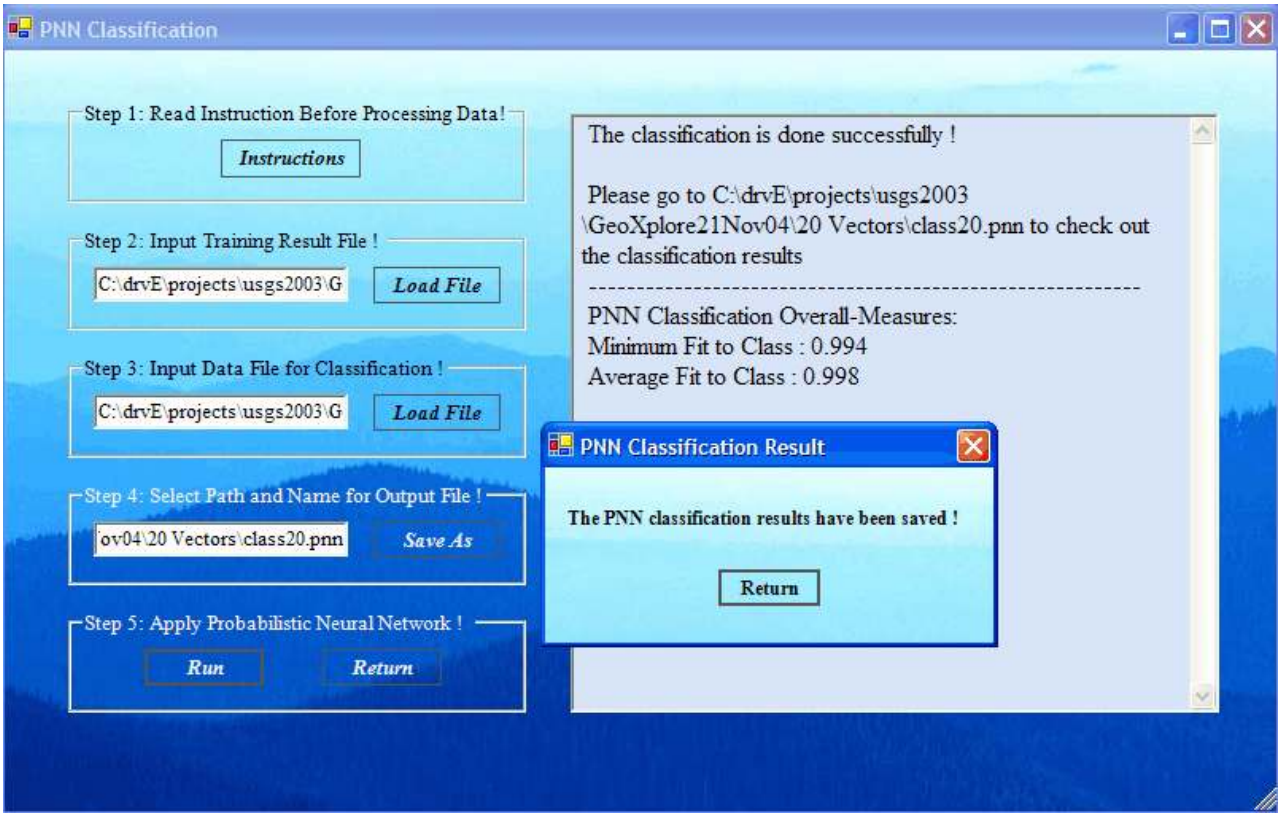
the lower ***Load File*** button and select the file of data to be classified whose name has the form **.dta*. Now we must provide a file name for the file where the results of the classification will be stored in a file with name format **.pnn*, so click the ***Save As*** button and provide a path and file name. Then click the ***Run*** button.

Figure 22. Loading the training parameters and data for PNN classification.



The PNN network executes with the loaded training parameters and processes the classification input data into a class for each input unknown feature vector. The results are written to the file with name format **.pnn* selected above and can be viewed or printed out with an editor program. The window now appears like the window shown in Figure 23.

Figure 23. The results of running the PNN classification function.



The goodness of classification numbers are shown at the bottom of the textbox in the form of a minimum goodness of fit and an average goodness of fit, both of which are fuzzy beliefs in the classification of this data with the training parameters loaded previously. The classification results file have the same format as shown in Figure 11.