User’s Guide to Pittnet Neural Network Educational Software

Brian Carnahan and Alice E. Smith
Department of Industrial Engineering
1048 Benedum Hall
University of Pittsburgh
Pittsburgh, PA 15261
aesmith@engrng.pitt.edu

April 1997

Funded by NSF Grant DMI-9502134 (CAREER)

User's Guide to Pittnet Neural Network Educational Software

Table of Contents

A. Introduction

A.1 *Objectives of the* **Pittnet** *Program*

The purpose of this computer program is to allow the user to construct, train and test different types of artificial neural networks. By implementing the concepts of templates, inheritance and derived classes from C++ object oriented programming, the necessity for declaring multiple large structures and duplicate attributes is reduced. Utilizing dynamic binding and memory allocation afforded by C++, the user can choose to develop four separate types of neural networks:

- Feedforward Network for control or prediction problems
- ART 1 Network for clustering binary signals
- Kohonen Self-Organizing Map for clustering real valued signals
- Radial Basis Function Networks for control or prediction problems

Once a network is selected, the user has the option of either uploading an existing neural network, or creating a new neural network. In designing each new network, the user can specify topology by selecting the number of layers and nodes per layer that are appropriate. The **Pittnet** program allows attributes of newly constructed networks to be stored for examination and also to be recalled for further training or testing. The networks developed by this program can supplement introductory instruction in the field of artificial neural computing.

A.2 *Hardware and Software Requirements*

With regard to hardware and software requirements, the **Pittnet** program must be run in the DOS environment only. A 486DX2 66 MHz personal computer (PC) or PC of higher grade is required for successful use of the program. The **Pittnet** program consists of two files:

1.	pittnet.exe	executable file of the **Pittnet** program
2.	pittnet.cpp	source code file of the **Pittnet** program

Both files can be stored on a 3.5 inch double sided, high density disk which contains 341,741 bytes of available memory space. The pittnet.exe file should be copied from the disk to the DOS environment of the PC prior to use.

The file containing the program's source code (pittnet.cpp) has been annotated so that a user can alter the computer code if he or she desires. It should be noted that if the source code is altered, the pittnet.cpp file should be recompiled on a Borland C compiler (version 3.0 or higher) in order to minimize compilation errors.

B. Program Initiation and Main Menu Interface

B.1 *Starting the Program*

To start the **Pittnet** program, simply type "pittnet" from DOS and hit the return key. The first input the program will require is the number of number of neural nets the user wishes to develop during the current session. For each network, the user must:

1. Specify the type of neural net he or she will use
2. Upload an existing neural net or design a new neural net to desired specifications
3. Train the neural network (optional)
4. Save the training results to a file (optional)
5. Save the testing results to a file
6. Test the neural network (optional)
7. Save the neural net to a file (optional)

The user must complete this sequence before the next network can be developed.

B.2 *Specifying the Type of Network*

From the main menu, the user must identify what kind of network he or she wishes to create, train or test. The interface presents the user with (5) options:

F. *Feedforward Network Using Backpropagation*

A. *Adaptive Resonance Theory Network for Binary Signals (ART 1)*

K. *Kohonen Self-Organizing Map*

R. *Radial Basis Function Network*

E. *Exit Program*

Options "F" and "R" are for problems concerning control or prediction, while options "A" and "K" are for problems which involve clustering data. The user can also select to exit the program once an network has been developed. The following sections will identify the required inputs for each network option.

C. Feedforward Network Using Backpropagation

C.1 *Signal Storage Requirements*

The signals used to train or test the Backpropagation network must be stored in the following configuration. Each row contains the values corresponding to the dimensions of an input signal followed by the values corresponding to the dimensions of its output signal. This file, as all files for the **Pittnet** program, must be space delimited. Three pairs of input / output vectors for the Backpropagation network are presented in the table below.

Table 1. A three-dimensional input vector and one-dimensional output vector.

Input Vector			Output Vector
30.0	60.0	60.0	160.5086
30.0	60.0	80.0	174.7022
30.0	80.0	0.0	173.2051

Prior to training or testing, the dimensions of all vectors are normalized by the program to range in value from 0 to 1 using the following equation.

$$\text{Normalized Value} = \frac{(\text{Actual Value - Minimum Value})}{(\text{Maximum Value - Minimum Value})}$$

Altering this normalization procedure would entail changing the source code and recompiling the program.

C.2 *Required Inputs for Backpropagation Network Construction*

In order to construct a Backpropagation network, the program will prompt the user for inputs in the following sequence:

Input Parameter	Maximum Limit
* dimensions of the input signal vector	24 dimensions
* number of hidden layers within the network	2 layers
* number of nodes within each hidden layer	10 nodes
* dimensions of output signal / nodes in output layer	10 dimensions

* activation function for the hidden layers	binary or bipolar sigmoid
* activation function for the output layer	binary or bipolar sigmoid

Each input represents an integer that must be entered by the user. The activation function for each hidden layer, if there is more than one, must be the same. Based on these inputs, the program constructs a fully connected neural network with connection weights randomly initialized between 1.0 and -1.0. In addition, each hidden and output node in the network is assigned a bias, which is also assigned a random value between 1.0 and -1.0.

C.3 *Training the Backpropagation Network*

For training this network, the program prompts the user for the following inputs:

- maximum number of epochs used for training (an epoch occurs when all input / output vector pairs in the training set have been presented to the network and the connection weights within the network have been adjusted accordingly)
- learning rate constant (real value between 0 and 1) for Backpropagation algorithm

  <u>Given connected neurons *i* and *j*</u>:

  Weight Correction (Δw_{ij}) = Learning Rate Constant **x** Local Gradient **x** Input Signal (neuron j)

- non-normalized minimum average squared error (Backpropagation terminates if target is reached)
- presentation of signals in random or fixed order sequence (if random presentation is chosen, the order of signal within the training set will change with each epoch)
- the name of the file containing the training data
- the number of signals in the file to be used for training

This section of the program will also prompt the user to specify whether or not he or she wishes to save the results of the training to a file. Each row of the training output file contains the results from a single epoch. The first two columns of each row contains the current number of epochs completed and the non-normalized average sum squared error (ASE) for the output signal. The subsequent columns contain the maximum error difference and mean absolute error difference for each output node. The table below lists the training results from three epochs for a neural network with 1 output:

Epoch #	ASE	Max Error	Mean Error
1	125.393	22.3562	14.2710
2	121.962	22.7038	14.3253
3	120.505	22.2450	14.0434

Figure 1. Typical training output file.

C.4 *Testing the Backpropagation Network*

For testing the Backpropagation network the program prompts the user for the following inputs:

- number of separate data files to be tested on the network
- name of the file currently holding the vector inputs and outputs that will be used in the test
- number of vectors within the file to be used in the test
- name of the file that will hold the results of the test

As with training, the program normalizes the input and output parameters of the test signals prior to initiation through the network. The results file contains rows of data where each row corresponds to the test vectors run through the network. Each row contains the non-normalized vector representing the target output for each input signal of the test. This vector is followed by a series of data points containing the non-normalized vector output from each node in the output layer. The next series of data points correspond to the absolute difference between the non-normalized vector output of the network and the actual target output vector. The final value in the row represents the sum square of error taken across the output nodes of the network.

C.5 *Storing the Backpropagation Network*

The structural characteristics of the Backpropagation network are listed in a file with ".net" extension:

- network identifier number, "1" identifies net as Feedforward with Backpropagation
- activation function for output layer (1 = binary sigmoid, 2 = bipolar sigmoid)
- number of nodes in output layer

- number of inputs to each output node (i.e. the number of nodes in the previous hidden layer)
- bias of each node in output layer
- connection weights of each input node to the output node (one row for each output node)
- number of hidden layers
- activation function for hidden layer(s) (1 = binary sigmoid, 2 = bipolar sigmoid)

For each hidden layer:

- number of nodes in the hidden layer
- bias of each node in the hidden layer
- connection weights of each input to the hidden node (one row for each output node)

The next example presents a data storage file for a Feedforward Backpropagation neural network (BAC.net). This topology of this network is shown in the next figure.

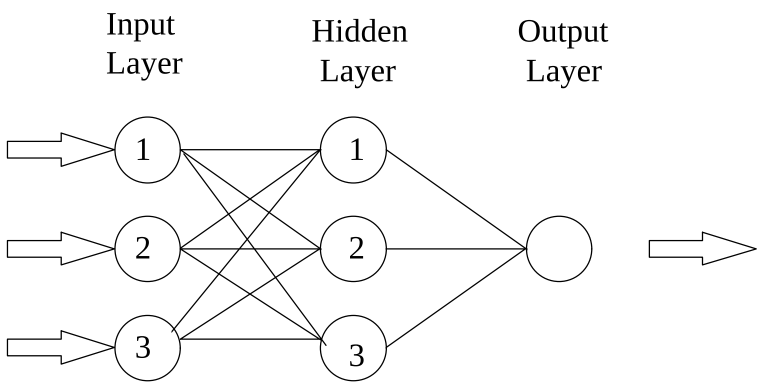

Figure 2. BAC.net - Feedforward neural network with backpropagation.

The data file contents of BAC.net are:

The data file values	*(Definitions)*
1	*(neural network identifier number)*
3	*(dimensions of input vector)*
1	*(activation function is a binary sigmoid)*
1	*(number of nodes in the output layer)*
3	*(number of inputs into each hidden node)*
0.610371	*(value of bias for output node)*
-1.108 0.09628 -0.056	*(connection weights from hidden nodes 1,2,3 to output node)*
1	*(number of hidden layers)*
1	*(activation function is a binary sigmoid)*
3	*(number of nodes in the hidden layer)*
-0.1128 0.8528 -0.285	*(bias values for hidden nodes 1,2,3)*
0.3331 0.9748 -0.719	*(connection weights from input nodes 1,2,3 to hidden node 1)*
-0.8779 0.2179 -0.303	*(connection weights from input nodes 1,2,3 to hidden node 2)*
-0.4176 -0.3144 0.186	*(connection weights from input nodes 1,2,3 to hidden node 3)*

D. Adaptive Resonance Theory Network 1 (ART 1)

D.1 *Signal Storage Requirements*

The signals used to train or test the Adaptive Resonance Theory Network 1 (ART 1) must be stored in the following configuration. Each row contains a binary string corresponding to the dimensions of an input vector. Five examples of a five dimensional input vector for the ART 1 network is presented in the table below.

Table 2. Typical input file for ART 1 network.

1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

D.2 *Required Inputs for ART 1 Network Construction*

In order to construct a ART 1 network, the program will prompt the user for inputs in the following sequence:

- the dimensions of the input signal vector (integer value)
- the vigilance parameter (real value between 0 and 1)

The first input establishes the number of nodes in the interface layer of the network. For the second input, the larger the vigilance parameter, the greater the similarity that must exist between input vector patterns in order to be assigned to the same cluster.

D.3 *Training the ART 1 Network*

For training this network, the program prompts the user for the name of the file containing the input vector patterns. The program scans this file in order to determine the total number of input vector patterns. This value will determine the total number of cluster nodes available to the network. The results of the training can be saved to a file with the following format:.

A.	B.	C.
1	10000	1
2	01000	2
3	00110	3
4	00101	3

(example of training results)

Figure 3. Typical ART 1 training results file.

The output from training is comprised of three columns of data. The first column (A.) represents the order number of the input vector pattern, with "1" corresponding to the first input pattern vector in the training file.

The second column (B.) represents the input vector pattern that was clustered. The third column (C.) represents the cluster node to which the input vector pattern in column B. was assigned by the ART 1 network. These nodes are used in sequence by the ART 1 network.

D.4 *Testing the ART 1 Network*

For testing the ART 1 network the program prompts the user for the following inputs:

- number of separate data files to be tested on the network
- name of the file that hold the vector inputs and outputs for the test (optional)
- number of vectors within the file to be used in the test
- name of the file that will hold the results of the test

The **Pittnet** program utilizes the same storage format used in training to save the results of tests on the ART 1 to a file. See section D.3 above.

D.5 *Storing the ART 1 Network*

The structural characteristics of the ART 1 network are listed in a file with ".net" extension:

- network identifier number, "2" identifies net as ART 1
- the dimensions of the input vector patterns (an integer)
- the weight update parameter
- the vigilance parameter
- the number of cluster nodes currently in use within the ART 1 network
- future cluster node tag - this integer represents the identify of the next cluster node to be activated in future training of this ART 1 network
- the maximum number of cluster nodes available to the ART 1 network

<u>for each node in the interface layer (equal to the dimensions of the input vector):</u>

- input weight vector (real value ranging from 0 to 1) - each value in the row corresponds to each dimension of the input signal vector.

<u>for each node in the cluster layer (equal to the number of input patterns used in training):</u>

- the cluster node identification number (an integer)

- input weight vector (real value ranging from 0 to 1) - each value in the row corresponds to each node in the interface layer.

The next example presents a data storage file for an ART 1 neural network (ART.net). The topology of this network is shown in the next figure.

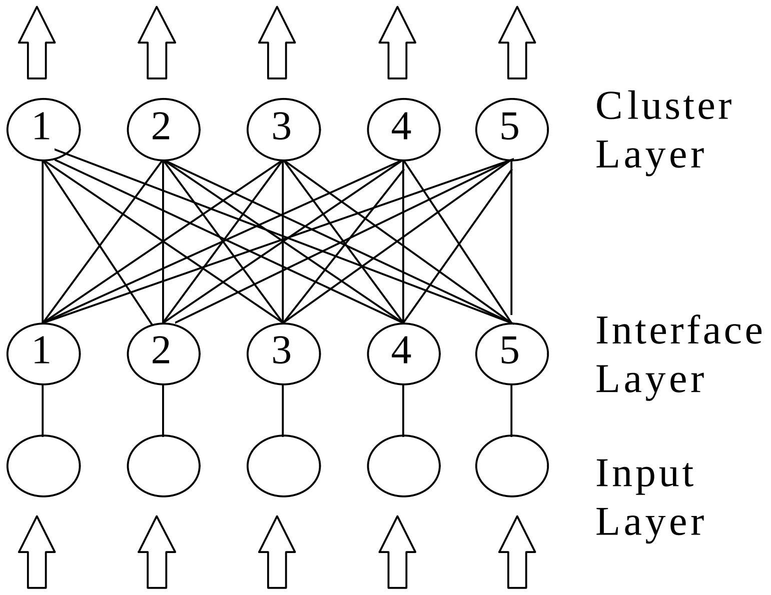

Figure 4. ART.net - an ART 1 neural network.

The ART.net file contains:

The data file values	*(Definitions)*
2	*(neural network identifier number)*
5	*(dimensions of input vector)*
2	*(weight update parameter)*
0.9	*(vigilance parameter)*
4	*(number of cluster nodes currently used)*
5	*(future cluster node tag)*
5	*(maximum number of clusters available)*
0 0 0 0 0	*(input weight vector: interface layer - node 1)*
1 0 0 0 0	*(input weight vector: interface layer - node 2)*
0 1 0 0 0	*(input weight vector: interface layer - node 3)*
0 0 1 0 0	*(input weight vector: interface layer - node 4)*
0 0 0 1 0	*(input weight vector: interface layer - node 5)*
1	*(cluster layer - node 1)*
0.167 0 0 0 0	*(input weight vector: cluster layer - node 1)*
2	*(cluster layer - node 2)*
0 0.167 0 0 0	*(input weight vector: cluster layer - node 2)*
3	*(cluster layer - node 3)*
0 0 0.167 0 0	*(input weight vector: cluster layer - node 3)*

4	*(cluster layer - node 4)*
0 0 0 0.167 0	*(input weight vector: cluster layer - node 4)*
5	*(cluster layer - node 5)*
0 0 0 0 0.167	*(input weight vector: cluster layer - node 5)*

E. Kohonen Self-Organizing Map

E.1 *Signal Storage Requirements*

The signals used to train or test the Kohonen Self-Organizing Map must be stored in the following configuration. Each row contains a series of real valued numbers corresponding to the dimensions of an input vector. Five examples of a four dimensional input vector for the Kohonen network is presented in the table below:

Table 3. Typical input file to Kohonen network.

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

As with input data used in the Feedforward network, the dimensions of all vectors are normalized by the program to range in value from 0 to 1 prior to training and testing.

E.2 *Required inputs for Kohonen Self-Organizing Map Construction*

In order to construct a Kohonen Self-Organizing Map, the program will prompt the user for inputs in the following sequence:

• the dimensions of the input signal vector	(integer value)
• the maximum number of clusters available to the network	(integer value)

The first input will establish the dimensions of the input weight vector to each of the cluster nodes. The weight of each dimension will be initialized with a random number between 0 and 1. Unlike the ART 1 network, the Kohonen requires the user to specify *a priori* the number of clusters he or she wishes to use in order to separate the input vector patterns.

E.3 *Training the Kohonen Self-Organizing Map*

In order to train the Kohonen network, the user is prompted to enter the following inputs to the program:

- the name of the file containing the input vector patterns
- the maximum learning rate parameter (real value between 0 and 1)
- the minimum learning rate parameter (real value between 0 and 1)
- the maximum number of epochs used for training (an integer value)

The last three inputs are used to establish the learning gradient. This gradient is needed to update the weights of a cluster node whose input weight vector has the minimum squared Euclidean distance to the input vector pattern. The maximum learning rate also serves as the initial learning rate. In updating the weight of each dimension, the following formula is applied:

$$\text{weight}_{\text{new}} = \text{weight}_{\text{old}} + [\text{learning rate} \times (\text{input pattern value - weight}_{\text{old}})]$$

After all input patterns have been run through the network (this constitutes an epoch), the learning rate itself is updated via:

$$\text{learning rate}_{\text{new}} = \text{gradient} \times \text{learning rate}_{\text{old}}$$

The gradient is determined by the following formula:

$$\text{gradient} = \text{max learning rate - epochs completed} \times \frac{\text{(max learning rate - min learning rate)}}{\text{(max number of epochs - 1)}}$$

E.4 *Testing the Kohonen Self-Organizing Map*

For testing the Kohonen network, the program prompts the user for the following inputs:

- number of separate data files to be tested on the network
- name of the file that hold the vector inputs and outputs for the test (optional)
- number of vectors within the file to be used in the test
- name of the file that will hold the results of the test

The results of the test are saved to a file in the following format.

A.	B.	C.
1	2042	1
2	2150	2
3	3032	3

(example of testing results)

```
 4  3023   3
```

Figure 5. Typical Kohonen training results file.

The output from training is comprised of three columns of data. The first column (A.) represents the order number of the input vector pattern, with "1" corresponding to the first input pattern vector in the testing file. The second column (B.) represents the input vector pattern that was clustered. The third column (C.) represents the cluster node to which the input vector pattern in column B. was assigned by the Kohonen network.

E.5 *Storing the Kohonen Self-Organizing Map*

The structural characteristics of the Kohonen are listed in a file with ".net" extension:

- network identifier number, "3" identifies net as a Kohonen Self-Organizing Map
- the dimensions of the input vector patterns (an integer)
- the number of cluster nodes (an integer)
- the input weight vector for each cluster node (real values ranging from 0 to 1)

The dimensions of the input weight vector for each cluster node are equal to the dimensions of the input signal vector. The next example presents a data storage file for a Kohonen Self-Organizing Map (KOH.net). This topology of this network is shown in the next figure.

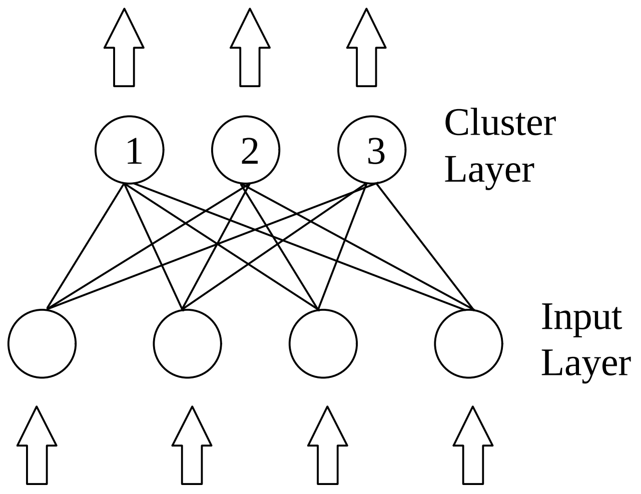

Figure 6. KOH.net - a Kohonen Self-Organizing Map.

The KOH.net file contains:

The data file values (*Definitions*)

3	*(neural network identifier number)*
4	*(dimensions of input vector)*
3	*(number of cluster nodes)*

0.437 0.0689 0.692 0.563	*(input weight vector for cluster node 1)*
0.419 0.984 0.739 0.0728	*(input weight vector for cluster node 2)*
0.948 0.126 0.192 0.771	*(input weight vector for cluster node 3)*

F. Radial Basis Neural Network

F.1 *Signal Storage Requirements*

The signals used to train or test the Radial Basis Function neural network (RBFN) must be stored in the configuration as those utilized by the Feedforward net with Backpropagation (see Section C.1).

F.2 *Required Inputs for Radial Basis Function Network Construction*

In order to construct a RBFN, the program will prompt the user for inputs in the following sequence:

- the dimensions of the input vector (an integer)
- the number of nodes in the cluster layer (an integer)
- the number of nodes in the output layer (an integer)
- the activation function for the output layer (binary or bipolar sigmoid)

The program will construct an artificial neural network with an initial layer of cluster nodes followed by a second layer of output nodes. The input vector weights for both types of nodes are randomly initialized and range between 0 and 1. The bias value for each output node is also randomly initialized and ranges between -1 and 1.

F.3 *Training the Radial Basis Function Network*

In order to train the RBFN, the user is prompted to enter the following inputs to the program:

<u>for training the cluster layer</u>:

- the maximum learning rate (real value between 0 and 1)
- the minimum learning rate (real value between 0 and 1)
- the maximum number of epochs for training (an integer)

<u>for training of the output layer</u>:

- the maximum number of epochs
- the learning rate (real value between 0 and 1)
- the minimum average square error (real value)

The training of the cluster layer is carried out first, using the same steps as those used in the training of the Kohonen Self-Organizing Map (see section E.3). Once completed, the output layer is subjected to supervised learning as used in the Feedforward network with backpropagation (see section C.3). This section of the program will also prompt the user to specify whether or not he or she wishes to save the results of the training to a file. The training output and format used to store the output are identical to those used by the Feedforward network with Backpropagation (see section C.3).

F.4 *Testing the Radial Basis Function Network*

For testing the RBF network the program prompts the user for the following inputs:

- number of separate data files to be tested on the network
- name of the file that hold the vector inputs and outputs for the test
- number of vectors within the file to be used in the test
- name of the file that will hold the results of the test

As with training, the program normalizes the input and output parameters of the test signals prior to initiation through the network. The results file contains rows of data where each row corresponds to the test vectors run through the network. Each row contains the non-normalized vector output from each node in the output layer. The next series of data points correspond to the absolute difference between the non-normalized vector output of the network and the actual target output vector. The final value in the row represents the sum square of error (SSE) taken across the output nodes of the network. An example of a results file for a three output RBF network is shown in the table below.

Table 4. Typical RBF testing results file.

Test Signal #	Target Output	Network Output	Absolute Error	SSE
1	30 60 90	32.4 57.8 90.5	2.4 2.2 0.5	5.43
2	90 30 60	87.1 28.7 61.5	2.9 1.3 1.5	6.18
3	60 30 90	59.3 30.0 92.7	0.7 0.0 2.7	3.89

F.5 *Storing the Radial Basis Function Network*

The structural characteristics of the RBFN are listed in a file with ".net" extension:

- network identifier number, "4" identifies net as a RBFN
- the dimensions of the input vector patterns (an integer)
- number of nodes in output layer
- activation function for output layer (1 = binary sigmoid, 2 = bipolar sigmoid)

- the number of cluster nodes (an integer)
- the bias for each node in the output layer
- the transfer function width for each node in the cluster layer

<u>for each output node</u>:

- the input weight vector (real numbers)

<u>for each cluster node</u>:

- the input weight vector (real numbers)

The next example presents a data storage file for a Radial Basis Function network (RBF.net). This topology of this network is shown in the next figure.

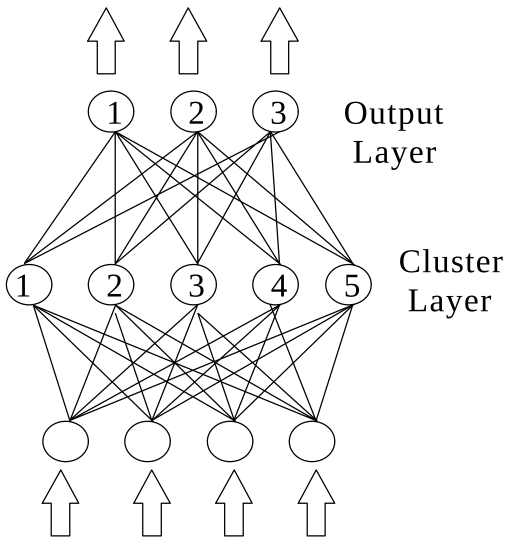

Figure 7. RBF.net - a Radial Basis Function Network.

The RBF.net file contains:

The data file values	*(Definitions)*
4	*(network identifier number)*
4	*(dimensions of input signal vector)*
3	*(number of nodes in output layer)*
1	*(activation function is binary sigmoid)*
5	*(number of nodes in cluster layer)*
1.20 -9.56 2.34	*(bias for output nodes 1, 2, and 3)*
-4.65 -0.72 4.68 -3.83 -3.86	*(input weight vector for output node 1)*
7.21 12.67 2.89 3.38 -7.23	*(input weight vector for output node 2)*
1.96 -9.94 -11.10 -1.57 8.93	*(input weight vector for output node 3)*
0.72 0.84 1.03 0.65 0.84	*(transfer function widths for cluster nodes 1-5)*
0.35 0.14 0.62 0.52	*(input weight vector for cluster node 1)*

0.93 0.38 0.65 0.067	*(input weight vector for cluster node 2)*
0.18 0.69 0.071 0.057	*(input weight vector for cluster node 3)*
0.66 0.44 0.69 0.62	*(input weight vector for cluster node 4)*
0.59 0.55 0.79 0.91	*(input weight vector for cluster node 5)*

G. INFORMATION FORM

To make the **Pittnet** software and User's Guide better, please mail or email the following information to:

Alice Smith
1031 Benedum Hall
Pittsburgh, PA 15261
aesmith@engrng.pitt.edu

Your Name: ____

Institution: ____

Address: ____

Email: ____________________________ Phone: ____________________

I have used Pittnet for:

Instruction ____ Class Name: ______________________________________

Class Level: __________________________ Number of Students: __________

Research ____

Type of Project(s): ____

____

____

Exploration / Instructional Consideration ______

Comments on **Pittnet**: ____

____

____

____