Assignment 3: Classification and Clustering Using Artificial Neural Networks

1. Introduction to SNNS and neural networks

Goal

The goal of this exercise is to introduce the SNNS neural networks simulator, and to give an idea of what a neural network and neural network learning are.

Files

You can download the files (`letters.net`, `letters.pat` and `letters.cfg`) needed for this exercise from the course web page at: `http://www.cb.uu.se/~hamed/data_mining/faq.html`
These files contain an untrained neural network (a multi-layer perceptron, MLP), training data and a suitable simulator configuration, respectively.

Start the simulator

Use the `snns` command to start SNNS (*SNNS is installed on all UNIX-workstations in building No. 1 in Polacksbacken, the same building where the lad-session will take place, time: 8:15-12:00, date: 27/11-2001, place: workstations room 1515*). Log information will appear in the window where you started SNNS.
Click on the one containing the text ”SNNS V 4.2” if you want it to disappear.

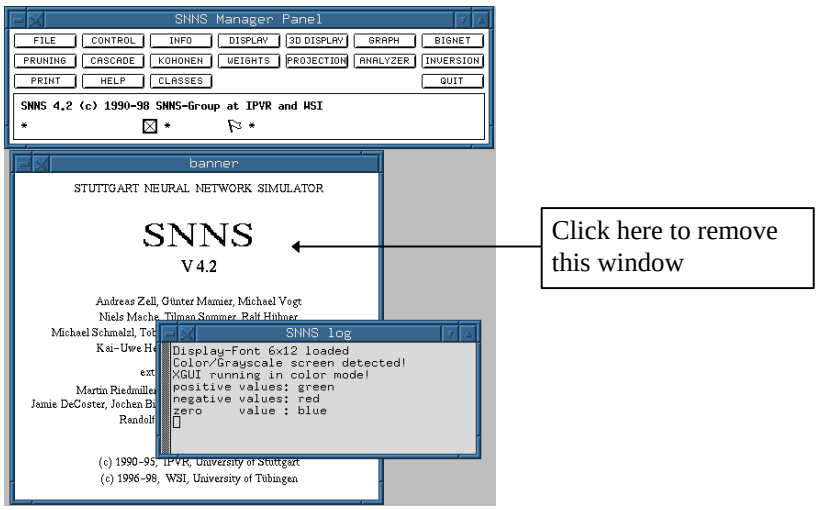

Figure 1: Three windows appear when SNNS is started.

The simulator is controlled from the ”SNNS Manager Panel” window.

Load the neural network and pattern files

A neural network is a network of simple processing elements, called neurons or nodes. Each node computes a single value at a time. Some nodes are dedicated *output nodes*, i.e. their computed values are also the output values of the whole network. There are also a number of *hidden nodes* in the network, nodes that only deliver values to other nodes.

All nodes ”listen” to the inputs and/or to each other through weighted connections. The network implements a function defined by these weights. Hence, to train a neural network is to find a set of weight values that implement the desired function.

In this exercise, we will teach a neural network to classify (”recognize”) bit maps of the capital letters A-Z of the English alphabet. The training data are pairs of input and output vectors. The input vectors are binary bit maps of the letters, and the corresponding outputs are binary vectors of 26 bits, with a 1 in the position representing the desired letter and 0’s in all the other positions.

Now you can load the files into SNNS. First, load the description of the neural network. To do this, press the FILE button in the manager panel to open a new window. In this window, press the NET button to list the files describing neural networks that are available in the current directory. In this case, the file name ” `letters`” should appear (the suffix ”`.net`” is not shown). Select this name by double–clicking on it. Then press the LOAD button.

You will be asked if you also want to load a configuration file (`letters.cfg`). Answer YES. Now a new window, labeled ”snns–display” appears. This window shows the untrained neural network. The network inputs (the bit map) are shown to the left, and the network outputs to the right. In between is a column of hidden nodes.

Values are shown both numerically and graphically. Blue boxes indicate 0 and green boxes 1. Values between 0 and 1 are shown by a proportional mix of the colors.

There are weighted connections between the nodes in the neural network, but they are not shown because the display window is set (through the configuration file) to display only connections having weights with absolute values greater than 2. In the untrained network, all weights have smaller absolute values.

Now load the data file: Press PAT in the FILE window, select ”`letters`” and press LOAD followed by DONE to close the window.

Prepare for training

Training and other operations on the network are performed from the control panel. To open it, press CONTROL in the manager panel. Then do the following in the control panel (see Figure 2):

- Press SHUFFLE. This will cause training data to be presented in random order to the network – almost always a good idea in neural network training.
- From the menu which pops up when the *top* button in the pile of four buttons labeled SEL. FUNC is pressed, select ”Std_Backpropagation”. You have now selected a learning algorithm called Backpropagation, which will be presented later in the course.
- Click on the INIT button to initialize all connection weights to small random numbers.

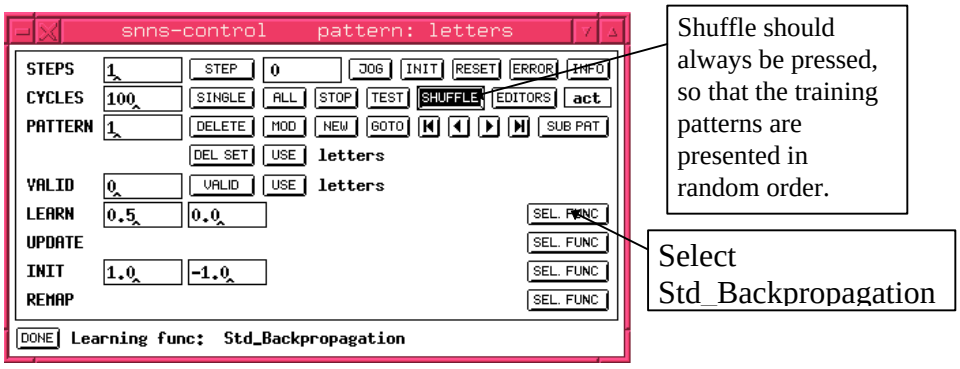

Figure 2: The control panel.

Now study what outputs the untrained network produce for different inputs. Press the TEST button to see the next input bit map and the corresponding output produced by the network. To view the desired output values that the network should learn, use the arrow buttons (below the SHUFFLE button) to browse the training set. Obviously, the untrained net does not solve the letter recognition problem.

Train the neural network

Backpropagation is an incremental learning algorithm. This implies that the training set must be presented to the neural network several times. After each pattern presentation, the learning algorithm adds small correction terms to the weights.

To see how the training proceeds, press the GRAPH button in the manager panel to open the graph window, where a *learning curve* will be plotted as the training proceeds. The x-axis of the diagram shows the number of cycles over the entire training set, and the y-axis shows a measure of the error. The default error measure is SSE, sum of squared errors,

which is computed by summing the squares of the differences between the actual outputs and the desired outputs for all the 26 outputs and all the 26 patterns. Change the error measure to SSE divided by the number of output units, by clicking on the text SSE and selecting SSE/#out from the menu which pops up. Also try re-scaling the axes by clicking the arrow buttons (”Scale X:” and ”Scale Y:”). Then reset the x-range to [0, 400] and the y-range to [0, 1].

Now the training may begin:

- The value in the box labeled CYCLES in the control panel indicates how many times the training set should be presented to the network. Change this value to 100 and press the ALL button. See what happens in the graph window and in the log window.
- When the 100 cycles are completed, step through the training set with the TEST button, and study how the outputs of the network have changed. Do you see an improvement compared to the untrained network?
- Press the ALL button again (train for 100 cycles more), and study the network’s outputs.
- Repeat this once again.
- Finally change the value in the CYCLES box to 700 and press ALL. After completion, the training set has been presented to the network 1000 times in total. Study the network outputs to see what it has learnt.

Probably, but not certainly, the network can now correctly classify bit maps of letters. During training, several of the connection weights have grown to absolute values greater than 2, and are therefore visible in the snns-display window. To hide them, press the SETUP button in that window, and click on the highlighted ON button in the row labeled ”links:”.

Save the trained network by pressing FILE in the manager panel, clicking on NET, typing a name for the network in the box above the buttons and pressing SAVE and DONE.

After saving, try the following:

- Re-initialize the weights, and retrain the network for 1 000 cycles. Compare the two learning curves in the graph window. Are they identical? If not, can you explain why?

On-line help

SNNS has a built-in help function, which is useful as a quick reference manual. Try it by pressing HELP in the manager panel. To browse the help document, either use ”emacs-like” keyboard commands (C-p, C-n, C-v, M-v, C-l etc., and searching with C-s), or the scroll bar (which is quite awkward). Try it! To search for a topic, press the TOPIC button, mark an item, and press the LOOK button.

When working with SNNS, the easiest way of accessing a user manual is to start a Web browser, for example Netscape, and open the document

The on-line manual is for SNNS version 4.1, but should be helpful anyway. A printed copy of the SNNS manual (v. 4.2) is available in computer lab 1412.

Quit

Press the QUIT button in the manager panel.

Questions

1. Suggest a strategy to know the best possible number of training cycles/epochs. Explain why it should work better than the way used in the exercise above.
2. Explain, by drawing a sketch for the ANN used above and writing the corresponding equations, how to use the trained ANN to classify/cluster the data.
3. In the control-panel, the left-most value in the LEARN row is the learning rate. The default value is 0.5. What happens if this learning rate is too small? What happens if it is too large? Try different values in the range 0.001 … 20 and see what happens with the error curve in the Graph window.
4. For what real-world data mining problems may a neural network, like the one you have just tried, be useful in practice?

2. DemoGNG

Goal

The goal of this exercise is to introduce the DemoGNG neural networks simulator/demonstrator, and to give an idea of what kinds/variants of neural networks exist for unsupervised learning.

What is DemoGNG?

DemoGNG is a Java applet which implements several methods/algorithms related to competitive learning. It is possible to experiment with the methods using various data distributions and observe the learning process. A common terminology is used to make it easy to compare one method to the other and judge each method's particular strengths and weaknesses. The following algorithms are available:

- LBG (Linde, Buzo, Gray)
- LBG-U (Fritzke)
- Hard Competitive Learning (standard algorithm) with
  - constant learning rate
  - decreasing learning rate
- Neural Gas (Martinetz and Schulten)
- Competitive Hebbian Learning (Martinetz and Schulten)
- Neural Gas with Competitive Hebbian Learning (Martinetz and Schulten)
- Growing Neural Gas (Fritzke)
- Growing Neural Gas with Utility (GNG-U, Fritzke)
- Self-Organizing Map (Kohonen)
- Growing Grid (Fritzke)

If you need/want to go into further details: look at the technical report "Some Competitive Learning Methods" by Bernd Fritzke, available from the DemoGNG web site.

Start DemoGNG

You can run DemoGNG directly from the DemoGNG web site at:

```
http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn
      /DemoGNG/HCL.html
```

You can find a link to this web site from the course web page at:

```
http://www.cb.uu.se/~hamed/data_mining/faq.html
```

After the Java applet has been launched, the DemoGNG main window appears. This window can be divided into four regions (as shown in figure 3):

- Network Model: here you can choose the desired algorithm.
- Drawing Area: to show the network for the selected algorithm. The geometric figure in the background of the area reflects the probability distribution. In every phase of the algorithm you can select a node and drag it to an arbitrary location within the drawing area. Additional information is displayed in the corners of this region:
  - upper left: Number of input signals occured so far
  - upper right: Version number
  - lower left: Number of nodes
  - lower right: Some additional information
- General Options: This region contains the non-specific parameters for all algorithms. There are three kinds of interface elements: buttons, check-boxes and pull-down-menus.

- Model Specific Options: This region shows the model specific parameters. Each time a new model is selected, the necessary parameters are displayed. For a complete description of the models and their parameters look at the technical report ”Some Competitive Learning Methods” by Bernd Fritzke which is available from the DemoGNG web site.

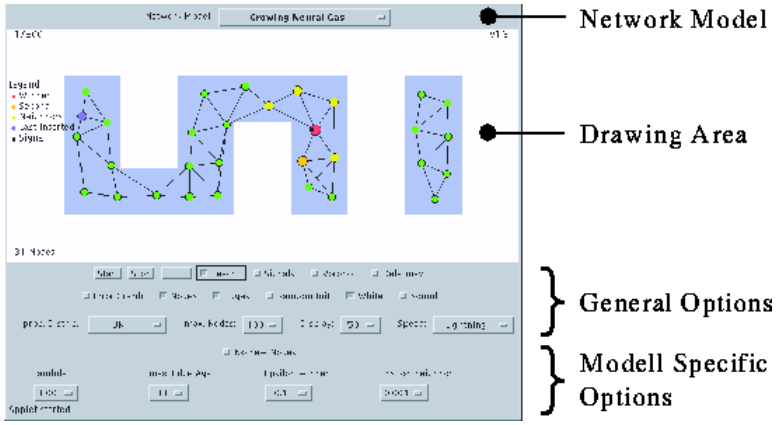

Figure 3: Overview of DemoGNG.

Questions

Try all the methods/algorithms presented in DemoGNG, combined with all of the probability distributions, and study their behavior for different choices/combinations of options/parameters (both general and specific options). After doing that, explain the following:

1. In the case of using ”Hard Competitive Learning”, what is the effect of changing the parameters called ”epsilon” and ”t-max” on the behavior, performance and the result of the neural network, and why?
2. In the case of using ”Self-Organizing Map”, what is the effect of changing the parameters called ”grid size”, ”epsilon”, ”sigma” and ”t- max” on the behavior, performance and the result of the neural network, and why?
3. In the case of using ”Growing Grid”, what is the effect of changing the parameters called ”lambda”, ”epsilon” and ”sigma” on the behavior, performance and the result of the neural network, and why? Explain also when and how this grid grows. Is it similar, when looking at the structure of the grid, to any other method/algorithm?
4. In the case of using ” Growing Neural Gas”, what is the effect of changing the parameters called ” lambda”, ”max. edge age”, ”epsilon” ”alpha”, ”beta” and ”utility” on the behavior, performance and the result of the neural network, and why? Explain also when and how this neural network grows. Is it similar, when looking at the structure, to any other method/algorithm? Compare the way this algorithm is using when adapting, to the following algorithms: ” Neural Gas”, ” Competitive Hebbian Learning” and ” Neural Gas with Competitive Hebbian Learning”.
5. Try to give qualitative comparisons (for the performance) of the methods/algorithms for the different cases which you have studied.
6. And finally, for what real-world data mining problems may neural networks, like the ones demonstrated by DemoGNG, be useful in practice?

Explain why a specific parameter has such an effect by referring to an equation and/or explaining what happening. Activate the general options: ”Teach”, ”Signals”, ” Voronoi”, and ”Delaunay”, to study the behavior of the neural networks.

3. Report

A written report which includes answers for all the questions (for both SNNS and DemoGNG) is required for this assignment.

(Comments about the assignment are encouraged and appreciated)