Department of Computer Science
Hong Kong Baptist University
COMP7560

Assignment 3 – 6 Marks
Due: 3/December. Submission at 17:30 (5:30pm). Submission must be done via the subject's Web site. Online submission only.

Note:

- **This is an individual assignment. Plagiarism will result in having 0 marks for all students involved.**
- **Late submissions will be marked with 25% deducted for each late day. Only full days are counted. Fractions of a day will always be rounded up (i.e. a submission 2 hours late will count as a one day). Submissions 4 or more days late will not be marked.**

Answer all of the following questions.

Question 1: Classification task **3 + 1 marks**

Problem description: Many data mining problems involve the dealing with complex data structures. For example, the mining of images and other multi-media content is becoming increasingly important. This assignment will be concerned with the mining of images from a collection of 3750 images depicting houses, ships, and policemen. The collection is known as the *Policeman benchmark*, a dataset which is used to test the performance of data mining software capable of dealing with graphs.
In general, images are popularly represented by a region adjacency graph (RAG). A RAG is created by marking regions in an image which are of uniform (or similar) color. These regions will be represented as a node in the RAG. If two regions are connected on the image then the associated nodes feature a link. The result is a RAG, one for each image. A common property of a RAG is that the nodes are connected, and that the links are undirected. The RAG algorithm has been adapted so that a directed acyclic rooted graph is extracted from each image. For your convenience, this has already been done for you. You are given a set of 3750 graphs which correspond to the 3750 images. Each graph consists of a set of labeled nodes. The label attached to each node gives a brief description of the associated region in the image. In the dataset, the label is a two-dimensional vector representing the center of gravity of the region within the image. Given this set of data, there are two tasks to complete: Solve a given classification task, and solve a clustering task for these images.

Back-propagation through Structure (BPTS) can be used to classify directed graphs. BPTS is a training algorithm for recursive MLP networks which are capable of processing sets of directed acyclic graphs (In literature, the term BPTS and recursive MLP are often used interchangeably). Your task is to train a recursive MLP (BPTS) on a given set of training data, and to test the performance of the trained BPTS on a given test set. There are a number of training parameters that need to be set for this task.

Do the following to train (and test) a recursive MLP on the given set of data. Download the BPTS software package from the subject's web site. The software package is a ZIP archive which needs to be uncompressed. The contents of the ZIP archive are:
BPTSNetGen.exe, a tool to generate a randomly initialized network.
BPTSTrain.exe, a tool to train the network on a given set of training data
BPTSSim.exe, a tool to test the classification performance of a trained network
cygwin1.dll, a library required by the tools mentioned before
bhsmulticnurmal.gph, the training data set.
bhsmulticnurmal_test.gph, the test data set.

As an example, the following sequence of commands would create, train, and test a recursive MLP:

Step 1: Create a newly initialized network, then save it to file “init.net”

```
BPTSNetGen.exe -two -seed 7 -inputs 2 -channels 6 -states 4 -outputs
14 -hiddens 5 -out init.net
```

The command line parameters used in this example mean the following:

`-two:` This selects the state-MLP architecture (see lecture notes).

`-seed 7:` This initializes the random number generator with a seed value. The seed value here is 7.

`-inputs 2:` Specifies the dimension of the label attached to each node. The dataset will contain 2-dimensional data labels, and hence, we specify 2 here.

`-channels 6:` Specifies the maximum out-degree of any node in the dataset. The policemen benchmark has an outdegree of 6.

`-state 4:` Specifies the number of neurons in the state layer. Here we use 4 state neurons.

`-outputs 14:` Specifies the dimension of the output layer. The target values of the policemen benchmark are 14-dimensional, and hence, the output layer must be 14-dimensional.

`-hiddens 5:` Specifies the number of neurons in the hidden layer. Here we use 5 neurons inside the hidden layer.

`-out init.net:` Specifies the name of a file to which the newly generated network is to be saved to. Here we specify that the network is to be saved to the file named “init.net”.

Step2: Train the network on the given dataset

```
BPTSTrain.exe -rprop -e 1000 -r 0.2 -t 0 init.net bhsmulticnurmal.gph
```

The command line parameters used in this example mean the following:

`-rprop:` Use RPROP instead of standard back-propagation. RPROP is a modification of the standard back-propagation algorithm which is insensitive to the size of the gradient.

`-e 1000:` Specify the number of training iterations. Here we specify that we wish to train for 1000 iterations.

`-r 0.04:` Specify the learning rate. Note that RPROP is largely insensitive to the learning rate because it uses an adaptive mechanism for the learning rate.

`-t 0:` Specify the target error rate. Here we specify that we aim at achieving an error or zero.

The remaining two command line parameters specify the name of the network that we wish to train (init.net), and the name of a file containing the training data (bhsmulticnurmal.gph).

By default, the program BPTSTrain will then save the trained network to a file named “TooEpochs.net”.

Step 3a: Obtain the classification performance on the training set

./BPTSSim -e 100 TooEpochs.net `bhsmulticnurmal.gph`

The command line parameters used in this example mean the following:

`-e 100:` This is a threshold value for rejecting a classification which is weaker than the threshold. Here we specify (in effect) that we do not wish to reject anything.

The remaining two command line parameters specify the name of the network that we wish to evaluate (TooEpochs), and the name of a file containing the data which we wish to use for testing the network (bhsmulticnurmal.gph).

BPTSSim prints a “confusion matrix” which shows how all the input patterns were classified by the network. In other words, the element in the i-th row and j-th column of the matrix counts how many

input patterns belonging to class i were classified by the network as belonging to class j. Ideally, we would like to see a diagonal matrix. The more we differ from a diagonal matrix, the worse the classification performance of the network. This is reflected by the "accuracy" value printed just below the confusion matrix.

Step 3b: Obtain the classification performance on the test set

./BPTSSim -e 100 TooEpochs.net `bhsmulticnurmal_test`

1a: Vary the three parameters -state, -hiddens, and -e (do not vary any of the other parameters), and compute for each combination of parameters that you have tried the confusion matrix. Develop a "feel" for which parameter is more important for improving the classification performance. Order these three parameters in the order of importance to the classification performance, and report the best overall classification rate for the training set, and for the test set. What parameters did you use to achieve the best performance? Are the parameters which produced the best performance on the training set the same as the parameters which produced the best test performance? Explain.
1b: Use a value for -state, -hiddens, and -e which produced your best performance on the test data set, then re-train a new network by solely varying the parameter -seed. This will change the initial network condition. What effect does the change of the seed value have on the classification performance of the network (on the test dataset).
1c: The classification performance for the training set is normally better than the classification performance on the test set. Why is this so?
1d: You are likely to observe that neither the classification performance of the trained network for the training set nor the classification rate on the test set reaches 100%. Explain why is is so hard to reach 100% classification performance.
1e: Bonus question: To date, the best classification rate on this benchmark problem (when using the state-MLP) is 99.55% on the training set, and 99.38% on the test set. If you are able to produce a better performance then report your result, and the provide the complete set of commands, parameters, etc which produced this result (your result must be re-producible) **1 bonus mark**

Notes: For question 1, training times of 10 minutes or more (depending on hardware) is normal.

Question 2: Unbalanced datasets **2 marks**
An unbalanced dataset is a set of data whose classes or clusters are of differing sizes. Formally, let n_i denote the size of a class i, then a dataset is said to be balanced if $n_i = n_j$ for all i,j. Otherwise the dataset is said to be unbalanced.
2a: Is the dataset used in question 1 balanced? Explain your answer.
2b: The use of a severely unbalanced training set causes most machine learning methods to perform poorly (on the smaller classes) whereas the use of an unbalanced test dataset does not normally cause any problems with machine learning methods.. How can this be explained?
2c: Propose one approach which can be taken to balance a training dataset.

Question 3: Association rules **1 mark**
Suppose we have market basket data consisting of 100,000 transactions and 250 items. If the support for item a is 25%, the support for item b is 85% and the support for itemset {a, b} is 20%. Let the support and confidence thresholds be 10% and 60%, respectively.

3a: Compute the confidence of the association rule **{a} → {b}**. Is the rule interesting according to the confidence measure?

3b: Compute the interest measure for the association pattern **{a, b}**. Describe the nature of the relationship between item **a** and item **b** in terms of the interest measure.

Note: For the experiments, I have provided the necessary executables, datasets, and libraries for SOM-SD and BPTS. These can be downloaded from the subject's web site.

-END-