Annex 1: How-to-use

Requirements

In order to run the solution **Python 2.5** or greater is required. Additionally to that, the following Python libraries are required:

- **ScyPy** (`sudo apt-get install python-numpy python-scipy python-matplotlib ipython ipython-notebook python-pandas python-sympy python-nose`)
- **Numpy** (Included in the ScyPy installation)
- **ffnet** (`sudo pip install ffnet`)

Executing the solution

The execution of the solution depends on the file `main.py`, present in the source code of the application. Running it is rather simple and it follows the following structure:

```
python -i <path to main.py> <path to data file> <structure of the network> <algorithm>
```

where:

`<path to data file>`: Is the path to the input data file, exported from the generator given for this project. The data files used for this project are included in the folder `dataFiles`.

`<structure of the network>`: This is the definition of the structure to use. It's format is: `[#L1,#L2,…,#Ln]`, where `#L1` is the number of nodes in the first layer. In our case we will use the structure: `[4,2,1]`

`<algorithm>`: Here you can specify which one of the solutions you want to use to solve the problem: `bp` for the Backpropagation algorithm implementation or `tn` for the truncated Newton method.

Examples:

Using the input file `TestData1024.dat` and the backpropagation algorithm the solution can be run in the following manner:

```
python -i main.py dataFiles/TestData1024.dat [4,2,1] bp
```

Which gives us the following output:

```
>> Input Information
    Neural Network Configuration: [4, 2, 1]
    Solving Algorithm Chosen:     Back Propagation Algorithm
    Total number of records:      1024
    Size of the training set:     819
    Size of the testing set:      205

>>Weights of Neural Network:
    W(00.10)  = -0.11    W(01.10)  = -0.49    W(02.10)  = -0.93    W(03.10)  = 0.59
    W(00.11)  = 0.30   W(01.11)  = 0.15   W(02.11)  = -0.48   W(03.11)  = 0.84
    W(10.20)  = -0.80    W(11.20)  = 0.66

>> Results of training set:
    172.0 correct prediction(s) of 205 testing records
    Accuracy:           83.90%
    Standard Deviation: 0.026
    Variance:           27.688

>> Total Duration:
    3.942 seconds
```

Another example running the truncated Newton algorithm, with a different structure would be:

```
python -i main.py dataFiles/TestData1024.dat [4,3,2,1] tn
```

Which returns the following output:

```
>> Input Information
    Neural Network Configuration: [4, 3, 2, 1]
    Solving Algorithm Chosen:     Truncated Newton Algorithm
    Total number of records:      1024
    Size of the training set:     819
    Size of the testing set:      205

>>Weights of Neural Network:
    W(00.10)  = 90.56    W(01.10)  = -100.00    W(02.10)  = -99.99    W(03.10)  = 94.40
    W(00.11)  = -29.41    W(01.11)  = -100.00    W(02.11)  = 49.83    W(03.11)  = 50.62
    W(00.12)  = 49.07    W(01.12)  = 50.12    W(02.12)  = -31.97    W(03.12)  = 14.35
    W(10.20)  = -2.16    W(11.20)  = -7.65    W(12.20)  = -13.97
    W(10.21)  = 54.54    W(11.21)  = -18.61    W(12.21)  = -100.00
    W(20.30)  = -2.16    W(21.30)  = -7.65

>> Results of training set:
    205.0 correct prediction(s) of 205 testing records
    Accuracy:           100.00%
    Standard Deviation: 0.000
    Variance:           0.000

>> Total Duration:
    2.644 seconds
```