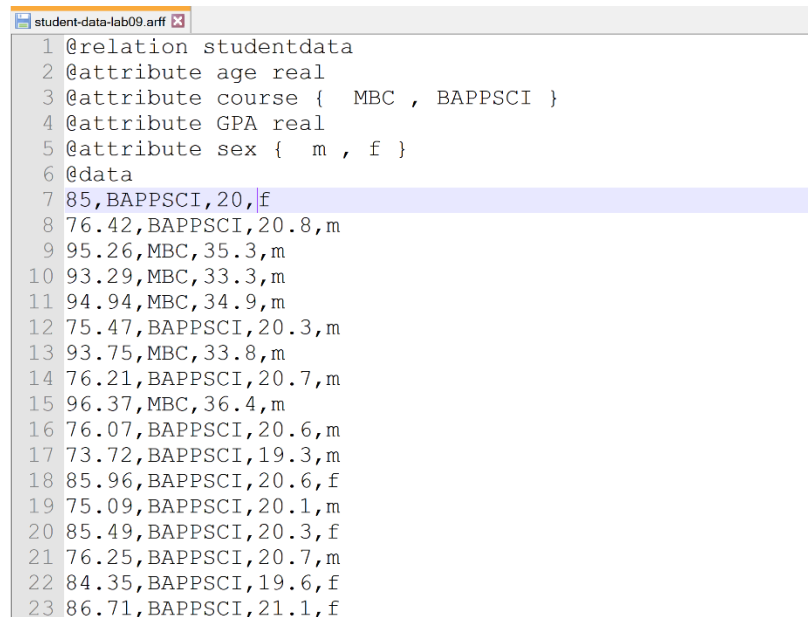


# Solution to Laboratory Week 9

This lab requires the files: code-and-scripts/iris-for-javanns.sh data/arff/student-data-lab09.arff in the directory /KDrive/SEH/SCSIT/Students/Courses/COSC2111/DataMining

The task is to generate javanns pattern files from the arff file, and train a neural network with these pattern files.

1. Examine the file student-data-lab09.arff and determine the necessary data encoding. The output variable is Sex.



```
1 @relation studentdata
2 @attribute age real
3 @attribute course { MBC , BAPPSCI }
4 @attribute GPA real
5 @attribute sex { m , f }
6 @data
7 85,BAPPSCI,20,f
8 76.42,BAPPSCI,20.8,m
9 95.26,MBC,35.3,m
10 93.29,MBC,33.3,m
11 94.94,MBC,34.9,m
12 75.47,BAPPSCI,20.3,m
13 93.75,MBC,33.8,m
14 76.21,BAPPSCI,20.7,m
15 96.37,MBC,36.4,m
16 76.07,BAPPSCI,20.6,m
17 73.72,BAPPSCI,19.3,m
18 85.96,BAPPSCI,20.6,f
19 75.09,BAPPSCI,20.1,m
20 85.49,BAPPSCI,20.3,f
21 76.25,BAPPSCI,20.7,m
22 84.35,BAPPSCI,19.6,f
23 86.71,BAPPSCI,21.1,f
```

Figure above shows student-data-lab09.arff opened in Notepad++. As it can be seen, this data set has 4 attributes.

The first attribute is age and it is numeric(real). No data encoding is required for numeric values.

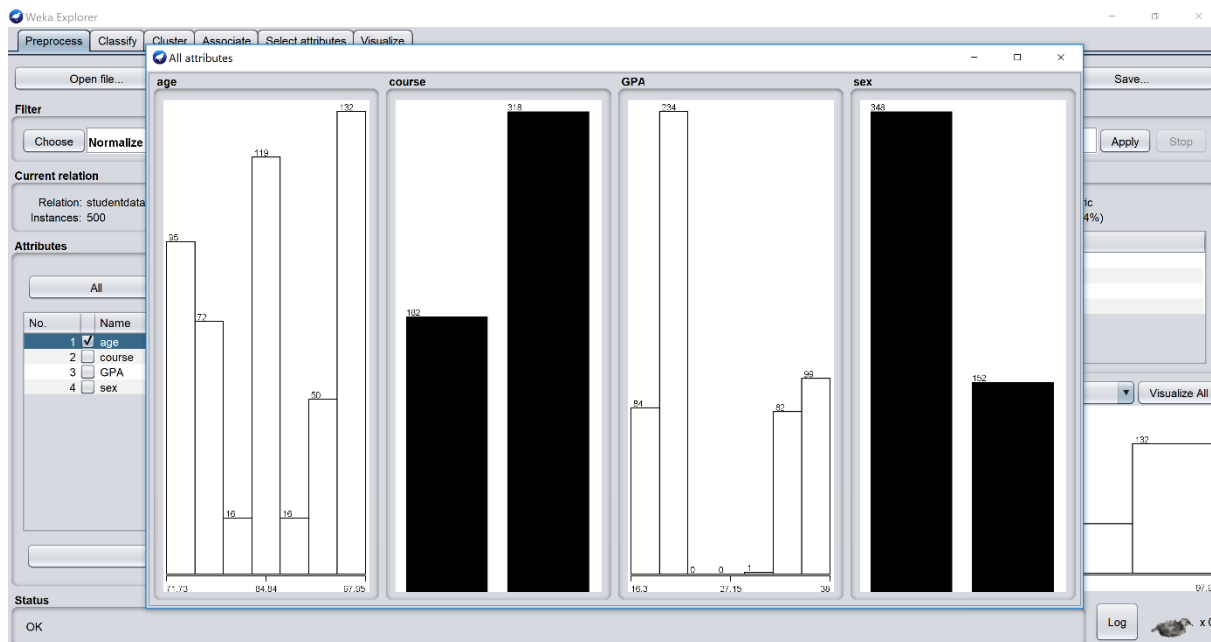
The second attribute is course which is categorical including two variables: MBC and BAPPSCI. Data encoding is required for categorical data. Because here, we have two categories, so we can encode them as MBC as 0 1 and BAPPSCI as 1 0 using two input nodes. Or we can encode as MBS as 0 and BAPPSCI as 1 using one input node.

The third attribute is GPA and it is real. So no data encoding is required.

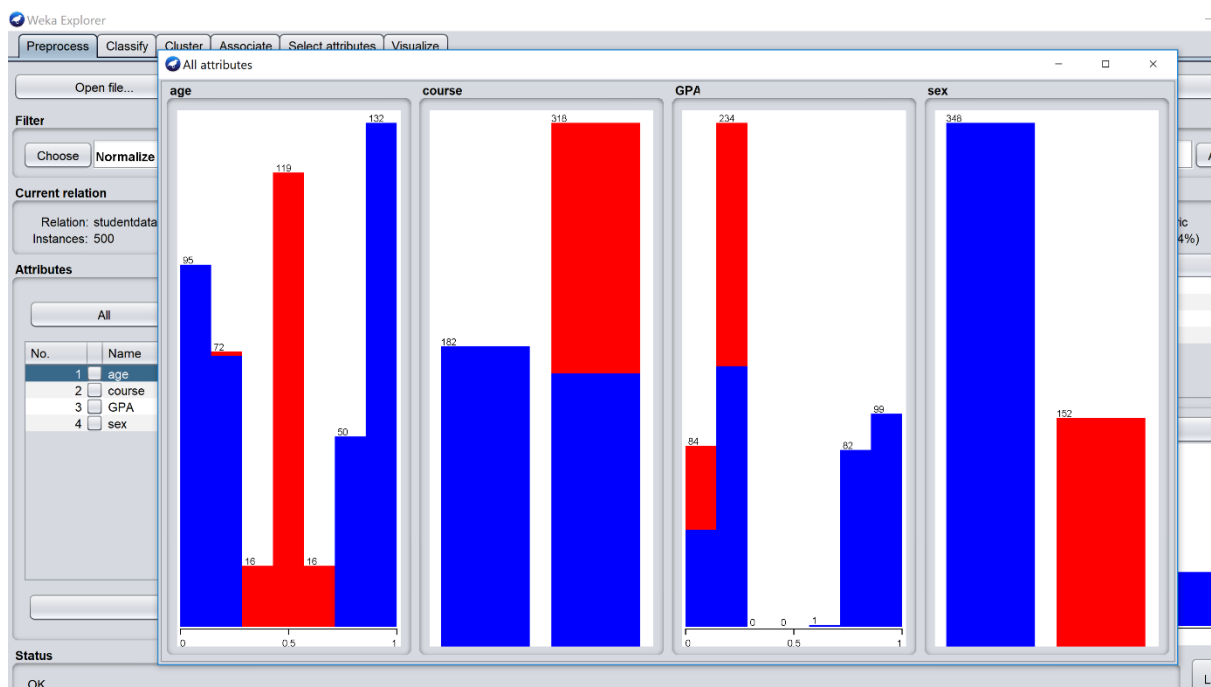
The fourth attribute that is target value or class attribute (output) is categorical including two categories of f and m. So data encoding for this attribute would be like f as 01 and m as 10 using two input nodes or f as 1 and m as 0 using 1 input nodes.

## 2. Load the file into weka and normalize the attributes with a filter.

You need to normalize the numeric attributes that are age and GPA. As you can see before normalization, the range of values of GPA is within 16.3 and 38 and age ranges is 71.73 to 97.95.



To normalize your numeric variables load data into weka. Go to preprocess tab. Go to Filter. Open unsupervised -> attribute->Normalized then click on apply tab. Then your numeric values are normalized and now they are within range of (0,1). You can see the output of normalization in the figure below.



### 3. Examine the file iris-for-javanns.sh. This file takes the iris data and randomly splits it into training, validation and test pattern files.

C:\Users\zhino\Documents\RMIT\DataMining\Week9\datamininglabdatasetweek9\iris-for-javanns.sh - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

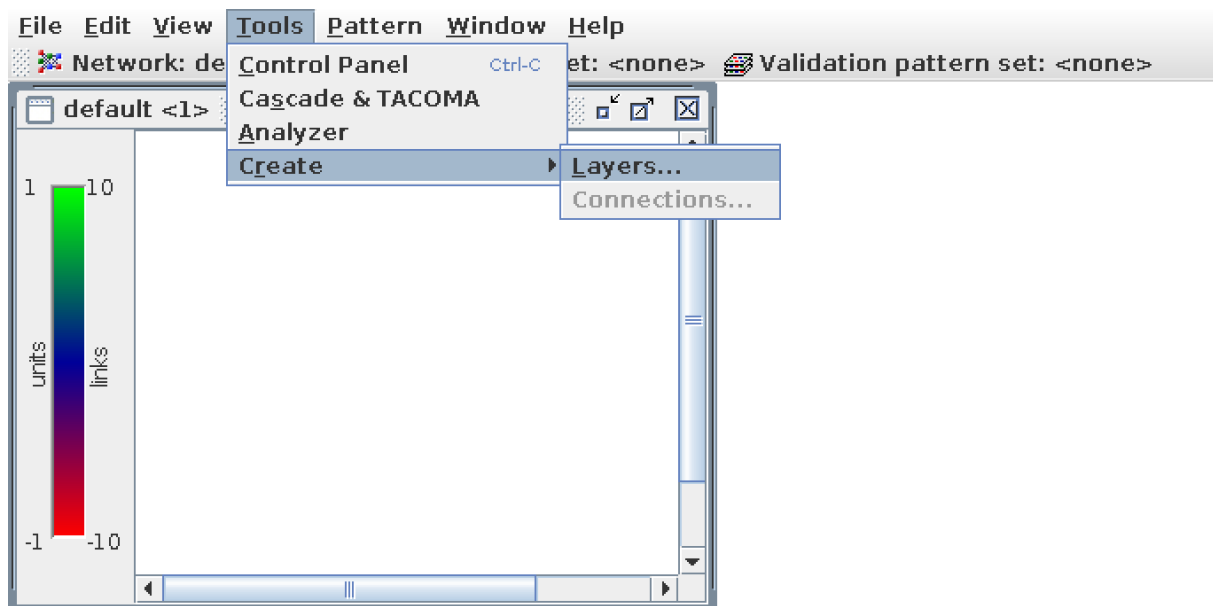
```
student-data-lab09.arff x iris-for-javanns.sh x
1  #!/bin/bash
2  TRAIN=100
3  VALID=20
4  TEST=30
5
6  tail -n +73 /KDrive/SEH/SCSIT/Students/Courses/COSC2111/DataMining/data/arff/UCI/iris.arff | \
7  fgrep -v "\"" | shuf |
8  sed -e"s/,/ /g" > templ.txt
9  # The training file
10 /bin/echo "SNNS pattern definition file V3.2" >iris-train.pat
11 /bin/echo "generated at Mon Apr 25 15:58:23 1994" >>iris-train.pat
12 /bin/echo "" >>iris-train.pat
13 /bin/echo "" >>iris-train.pat
14 /bin/echo "No. of patterns : $TRAIN" >>iris-train.pat
15 /bin/echo "No. of input units : 4" >>iris-train.pat
16 /bin/echo "No. of output units : 3" >>iris-train.pat
17
18 head -n $TRAIN templ.txt |
19 sed -e"s/Iris-setosa/1 0 0/g" |
20 sed -e"s/Iris-versicolor/0 1 0/g" |
21 sed -e"s/Iris-virginica/0 0 1/g" >>iris-train.pat
22
23 # The validation file
24 /bin/echo "SNNS pattern definition file V3.2" >iris-valid.pat
25 /bin/echo "generated at Mon Apr 25 15:58:23 1994" >>iris-valid.pat
26 /bin/echo "" >>iris-valid.pat
27 /bin/echo "" >>iris-valid.pat
28 /bin/echo "No. of patterns : $VALID" >>iris-valid.pat
29 /bin/echo "No. of input units : 4" >>iris-valid.pat
30 /bin/echo "No. of output units : 3" >>iris-valid.pat
31 FROM=$(expr $TRAIN + 1)
32 tail -n +$FROM templ.txt | head -n $VALID |
33 sed -e"s/Iris-setosa/1 0 0/g" |
34 sed -e"s/Iris-versicolor/0 1 0/g" |
35 sed -e"s/Iris-virginica/0 0 1/g" >>iris-valid.pat
36
37 # The test file
38 /bin/echo "SNNS pattern definition file V3.2" >iris-test.pat
39 /bin/echo "generated at Mon Apr 25 15:58:23 1994" >>iris-test.pat
40 /bin/echo "" >>iris-test.pat
41 /bin/echo "" >>iris-test.pat
42 /bin/echo "No. of patterns : $TEST" >>iris-test.pat
43 /bin/echo "No. of input units : 4" >>iris-test.pat
44 /bin/echo "No. of output units : 3" >>iris-test.pat
45 FROM=$(expr $FROM + $VALID)
46 tail -n +$FROM templ.txt | head -n $TEST |
47 sed -e"s/Iris-setosa/1 0 0/g" |
48 sed -e"s/Iris-versicolor/0 1 0/g" |
49 sed -e"s/Iris-virginica/0 0 1/g" >>iris-test.pat
50
51
52
53
```

As you can see from the above figure, this script split iris data set into iris-train.pat, iris-valid.pat, and iris-test.pat. The iris data set includes 150 instances that in this script 100 are randomly taken as train, 20 as valid and 30 as test.

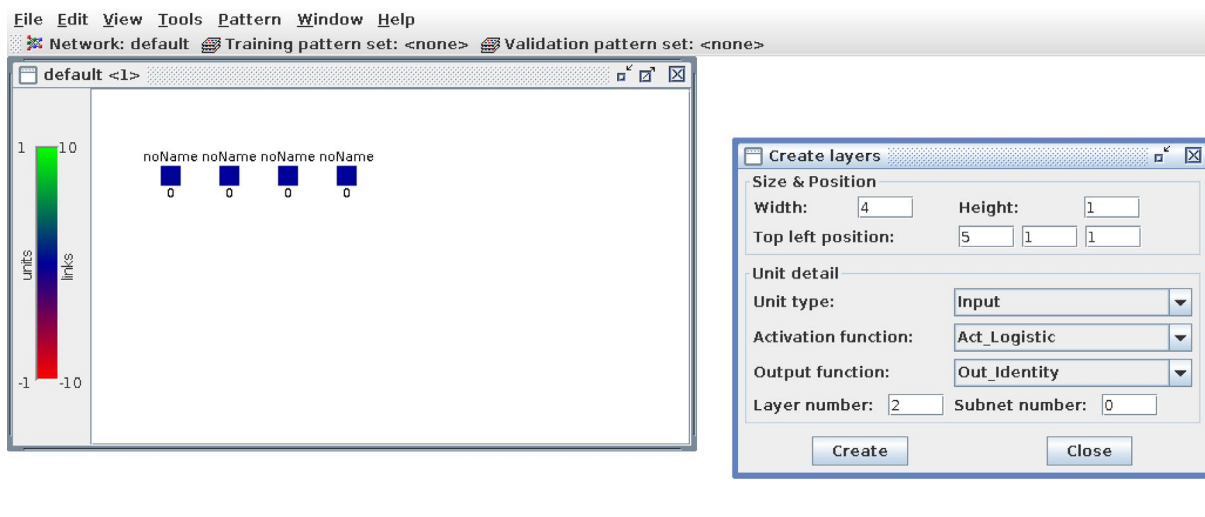
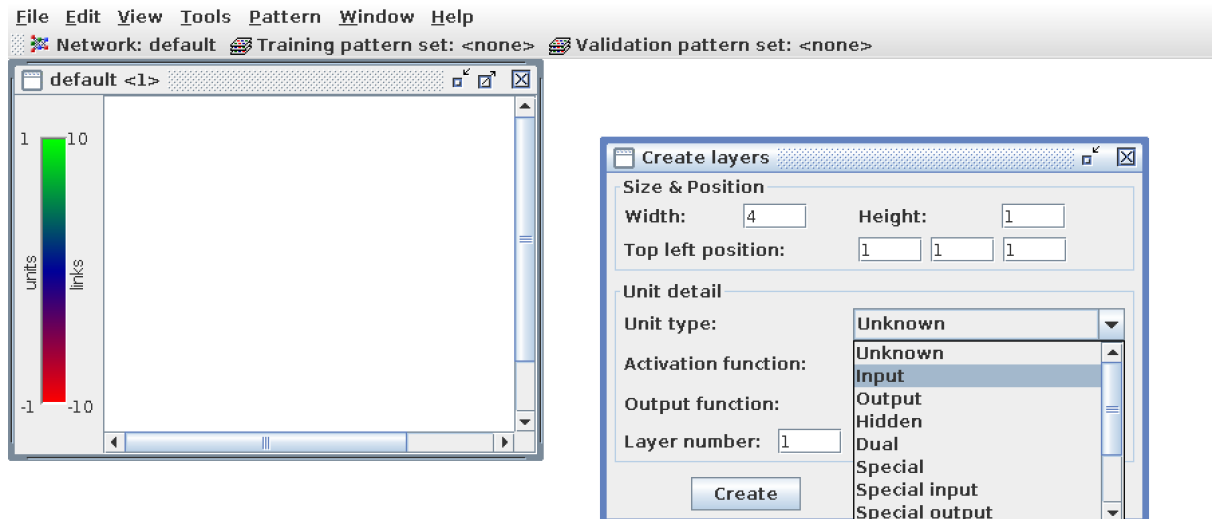
4. Modify the file `iris-for-javanns.sh` to only generate `student-data-train.pat`, and exit immediately.

```
1  #!/bin/bash
2  TRAIN=334
3  VALID=83
4  TEST=83
5
6  tail -n +9 /home/ehl/E05051/HDrive/DM/Normalized-student-data-lab09.arff | \
7      fgrep -v "%" | shuf |
8      sed -e"s/, / /g" > templ.txt
9  # The training file
10 /bin/echo "SNNS pattern definition file V3.2" >student-data-train.pat
11 /bin/echo "generated at Mon Apr 25 15:58:23 1994" >>student-data-train.pat
12 /bin/echo "" >>student-data-train.pat
13 /bin/echo "" >>student-data-train.pat
14 /bin/echo "No. of patterns : $TRAIN" >>student-data-train.pat
15 /bin/echo "No. of input units : 4" >>student-data-train.pat
16 /bin/echo "No. of output units : 2" >>student-data-train.pat
17
18 head -${TRAIN} templ.txt |
19     sed -e"s/MBC/1 0/g" |
20     sed -e"s/BAPPSCI/0 1/g" |
21     sed -e"s/f/1 0/g" |
22     sed -e"s/m/0 1/g" >>student-data-train.pat
23
24 exit
```

5. Using `javanns`, generate a suitable network and verify that your `student-data-train.pat` can be loaded and be successfully used for training.



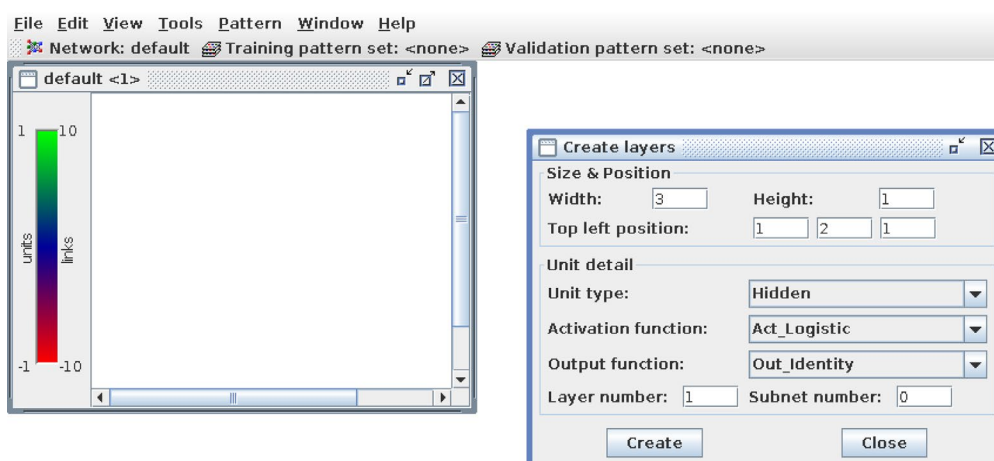
To create your network, you need to go to Tools, then select Create and then select Layers as shown in the above figure.

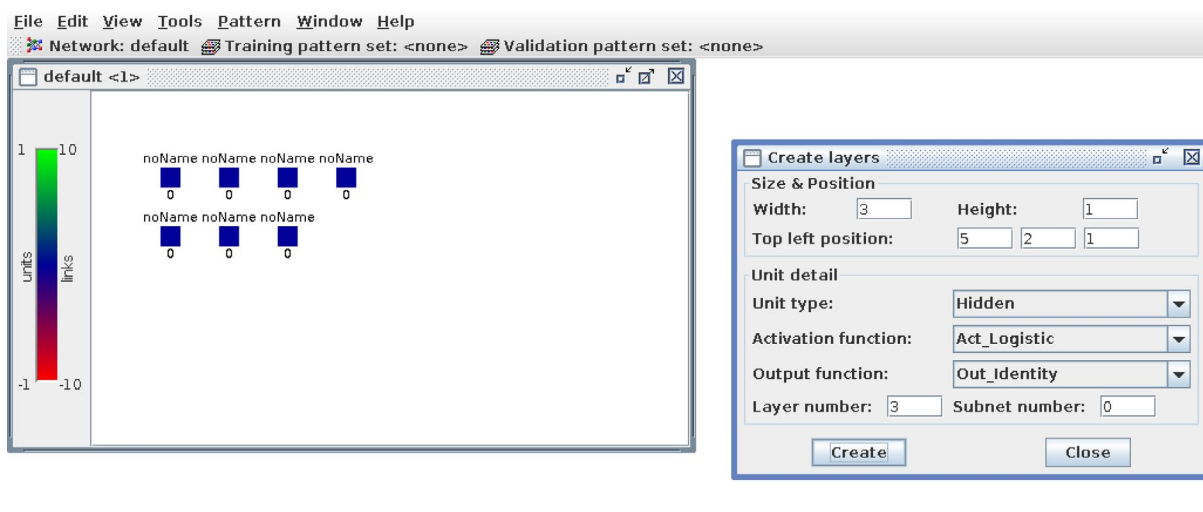


Now, you need to step by step select your input nodes, where for this data set you have totally 4 input nodes, one node for each numeric attribute and 2 nodes for categorical attribute.

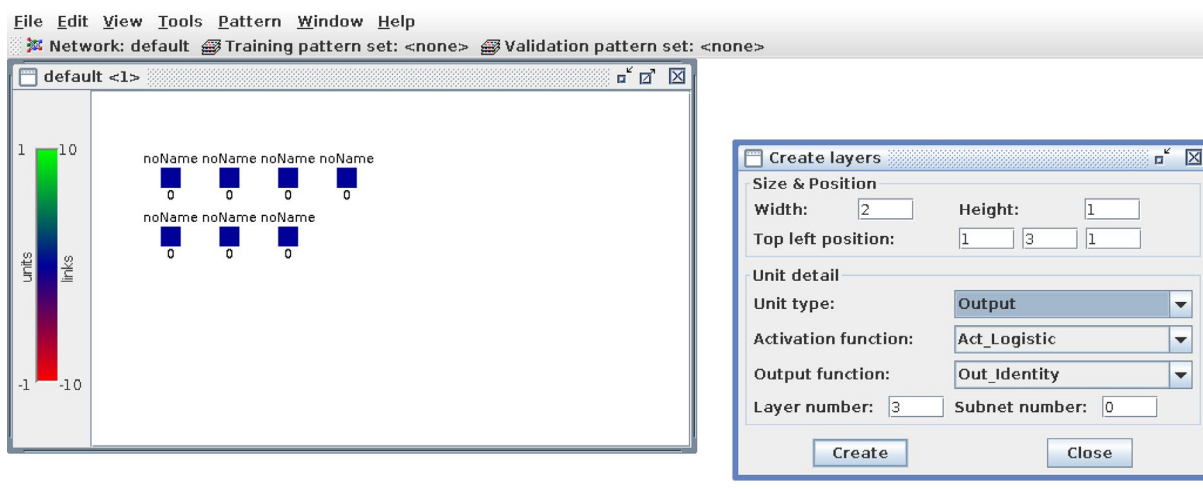
Then you have to key 4 in the Width box. And select Unit type as Input. Then click on create and the input layer will be created as shown in the below figure.

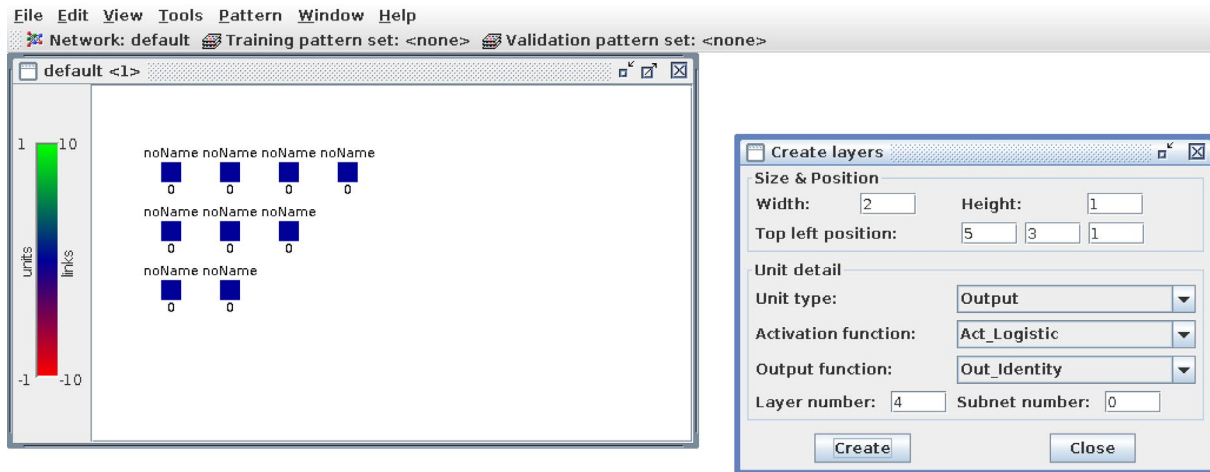
Now for creating your hidden layer, you need to choose some number of hidden nodes that here we choose 3, then put this number in the Width box as 3. After that you need to modify the position of hidden layer in Top left position boxes as you can see from the above figure.



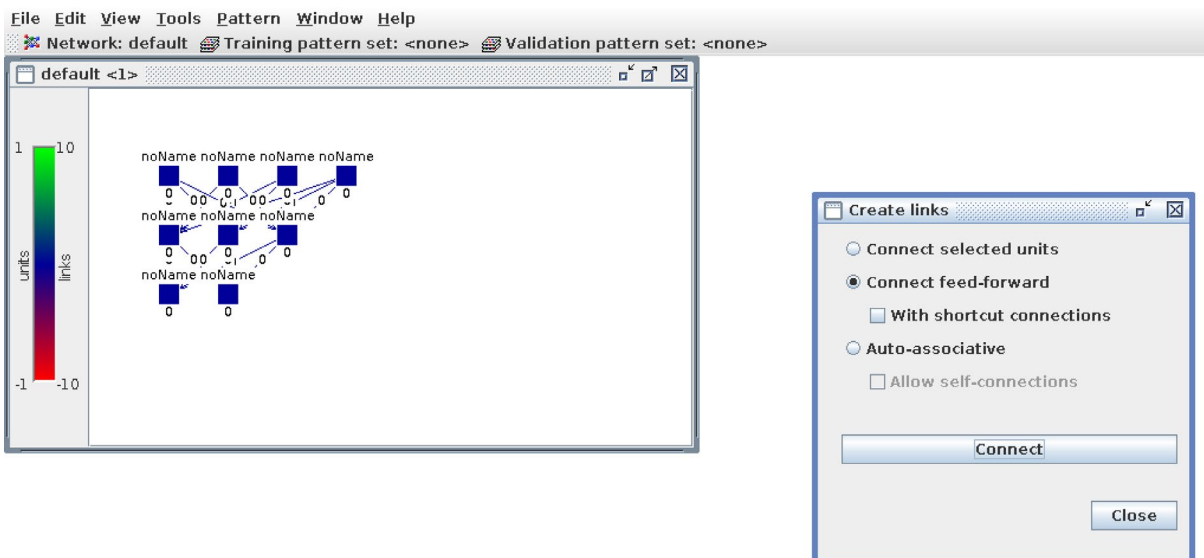
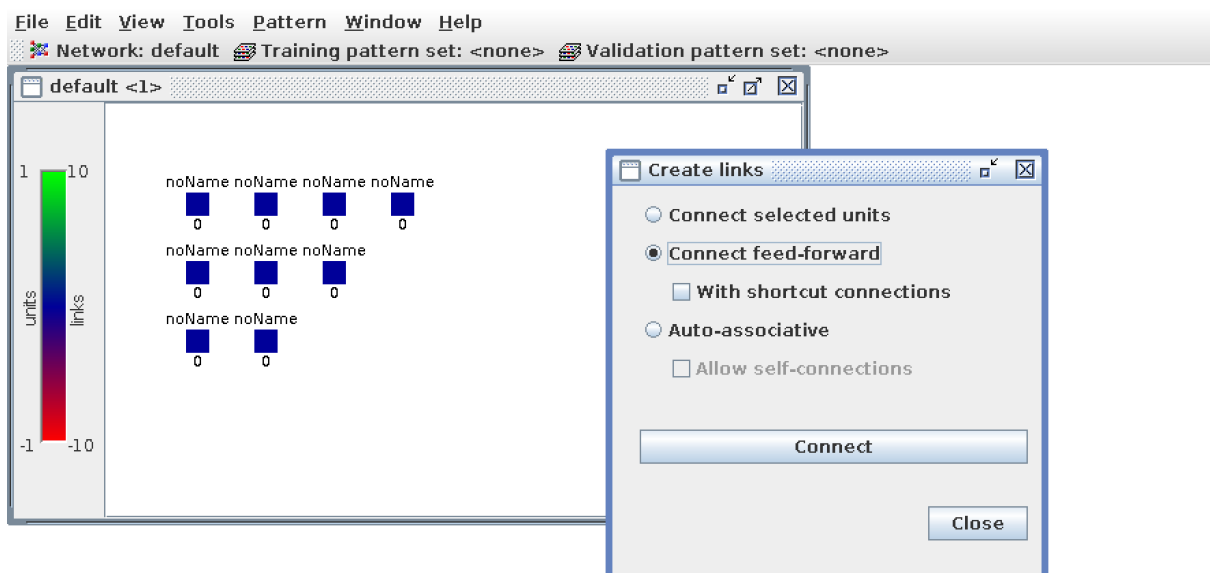


The last step is to create output layer. For this data set you can go with either 1 output node or 2 output nodes as you have two categories.





After creating your layers, you need to connect the layers by going to Tools, create, and selecting connection and select connect feed-forward and clicking connect.



**6. Carry out a training run and get a result file. Use the analyze program to get the classification error rate.**

After running the script through linux server, you would get a .pat file. After building your network as shown above, you need to load .pat file into javanns and start training then save result and after that using analyse check on the classification error.

**7. Extend the script to generate student-data-valid.pat and student-data-test.pat**

Take the same steps as yo did for training.

**8. Repeat the training with all 3 pattern files and get the result file.**