# RMIT University
# School of Science
# COSC2110/COSC2111 Data Mining
## Laboratory Week 8 Basic JavaNNS

---

Aims of this lab

- Learn how to train a neural network classifier with javanns
- Using an exiting network and existing pattern files

---

**IMPORTANT: Running JavaNNS**
You can run `JavaNNS` on the core teaching servers (saturn, jupiter, titan). or on a PC.
I have been told that there is a Mac version at
http://homepages.di.fc.ul.pt/ lcorreia/JavaNNS.jar
that previous students have used successfully.

**Running JavaNNS on titan/saturn/jupiter**

1. Log into `titan.csit.rmit.edu.au` with an X11. connection.
   (a) **Note** At the moment xming and putty from lab PCs are not working properly. ITS is working on fixing the problem. In the meantime you can use this workaround:
   (b) With file explorer, go to
   `K:/SEH/SCSIT/Students/Courses/COSC2111/DataMining/xming-and-putty`
   (c) Click XLaunch and click next to the end
   (d) Click putty in this folder, go to jupiter.csit.rmit.edu.au with X11 forwarding.
   (e) Login and run 'xeyes &' to verify that the X11 connection is working.
2. `cd /KDrive/SEH/SCSIT/Students/Courses/COSC2111/DataMining/`
3. `source javanns-env.sh`
4. `javanns &` [Note: The first time only, you will need to set the SNNS library. Browse to javanns and select `libSNNS_jkr.so`]

**Running JavaNNS on a PC**

[Note: It might need a lot of attention to detail to get this working]

1. Download the software to a local PC folder from:
   `http://www.ra.cs.uni-tuebingen.de/downloads/JavaNNS/JavaNNS-Win.zip`
2. UNZIP it to your HDrive, double click JavaNNS.jar to start.

3. If you have trouble starting JavaNNS on a PC, copy the file
   `/KDrive/SEH/SCSIT/Students/Courses/COSC2111/DataMining/`
   `javanns/JavaNNS.bat`
   to your JavaNNS-win folder, edit JavaNNS.bat to point the folder where java is installed and start with JavaNNS.bat.

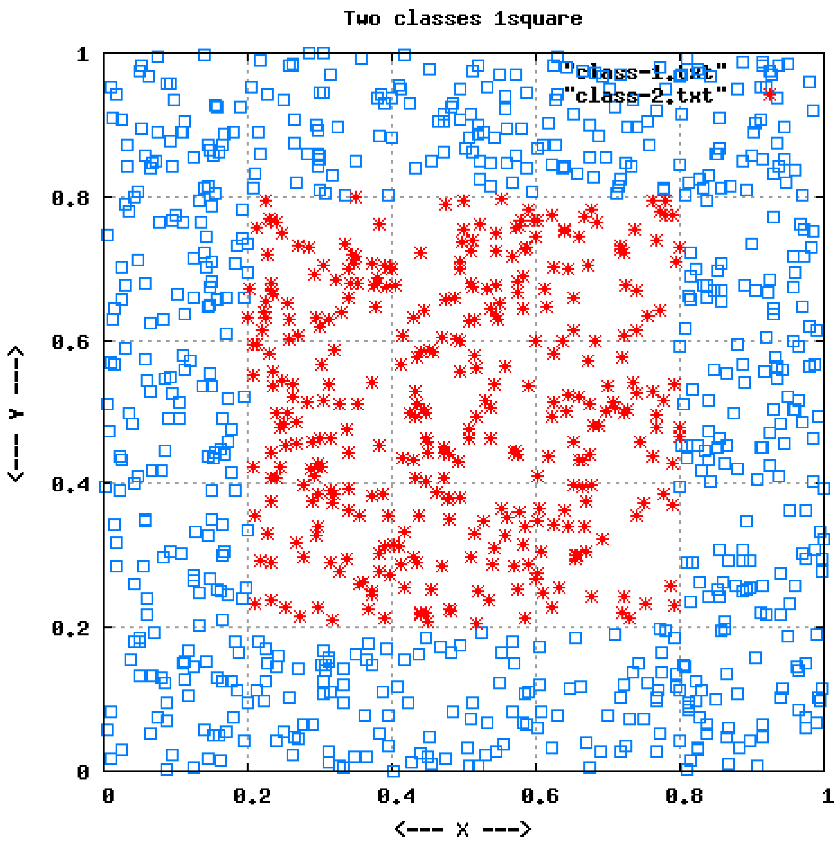
In this exercise we train a neural network to learn the XOR function.

1. Start JavaNNS
2. Load the file containing the training data, xor.pat. This file can be found in sub-directory, named examples, of the JavaNNS home directory.
   File --> Open xor.pat --> Select the examples folder, and then open `xor.pat`.
3. Now load a pre-designed network from the file `xor_untrained.net`. You may need to resize the window to have a full view of the network.
   File --> Open --> Select the examples folder, and then open xor_untrained.net.
4. Now get the error graph. Go to View --> Error Graph (or press Ctrl+E). You can view the training errors in this window. You may want to adjust the range of the axes by clicking on the arrows near the axes.
5. Go to Tools --> Control Panel. This is the main interface for training the network. You can also invoke this by pressing Ctrl+C whenever JavaNNS is in focus. We will need to use the settings in the Learning tab to do the training.
6. Use the default settings in the control panel. Click Learning, then init then Learn all. What can you see in the Error Graph? Keep clicking Learn All intil the error gets close to zero. You may want to increase the number of cycles.
7. Repeat the init and train procedure several times. What patterns do you notice in the error graphs?
8. Try different different learning rates. Can you find any relationship between learning rate and error trajectory?
9. Try Batch Backpropagation as the learning function. How is it different?
10. Try Backprop-Momentum as the learning function?

In this exercise we train a classifier with training, validation, and test sets.

1. Restart JavaNNS and reload `xor_untrained.net`. This has 2 inputs and one output, which suits the 1square data.
2. Load the 3 files from
   `/KDrive/SEH/SCSIT/Students/Courses/COSC2111/DataMining/data/other/`:
   `1square-train.pat`
   `1square-test.pat`
   `1square-valid.pat`
3. In Control Panel --> Patterns set the training file to 1square-train.pat and the validation file to 1square-valid.pat

4. Train the network for 5000 cycles with Init and Learn All.
5. Generate a result file, `1square.res`, with Save Data. Include the input and output patterns. [You will need to specify a folder for which you have write access.]
6. In a shell window, get the classiification rate with the default analyse strategy with *(Put the following two lines on one command line)*
   ```
   /KDrive/SEH/SCSIT/Students/Courses/COSC2111/DataMining/
   javanns/analyze -c -i 1square.res
   ```
   Note: If you have used `source javanns-env.sh` you can use
   `analyze -c -i 1square.res` or
   `analyze -s -e band -l 0.4 -h 0.4`
7. What do you conclude? Is more training needed? More hidden nodes? A different interpretation strategy for analyze?
8. Try with `2-1-1.net`
9. Write a script to convert the `iris.arff` to suitable training, validation and test files for JavaNNS. How does the JavaNNS accuracy compare with the Weka classifiers?


Two classes 1square
"class-1.txt"
"class-2.txt"
<--- X --->
<--- Y --->