Laboration 3: The Bayesian Neural Network Model

1 Objectives

In this exercise you will study the operation of a one-layer Bayesian neural network. When you are finished you should:

- have some hands on experience with the operation of a Bayesian artificial neural network
- understand how neural networks can be used for classification
- understand how noise affects the recognition of patterns
- know that relaxation time is dependent upon the initial distance to the attractor and is longer for ambiguous/conflicting inputs
- know how accuracy in recall deteriorates when storage capacity is exceeded
- know that activities in a Bayesian neural network can represent confidence information

2 Tasks

The one-layer Bayesian neural network you will work with in this exercise has been implemented in the neural network program annx which you should use to solve all tasks. First you are going to run the Bayesian network as a recurrent neural network and investigate its storage capacity and noise tolerance. Then you will use the same neural network model in a classification task.

For a description of how to use the program, see the manual pages at the end of this instruction (or the file annx.ps).

3 Lab files

To be able to run the simulations it is necessary that you have the right environment. If you haven’t already followed the instructions in the section “Setting up the Course Environment” in the handouts. (It isn’t necessary to do this more than once, so if you did this at an earlier occasion, you don’t need to do it again.)

You should also create a directory for and copy the pattern and database files. Type the following commands at the shell prompt (>):

```
> cd ~/ann02
> cp -pr /info/ann02/labbar/lab3 .
> cd lab3
```
4 Basic functions

Start by training the network with all the patterns in the file symbol.pat (with the command learn). Thereafter, take one of the patterns in this file and stimulate the network with it (using the command input). The stimulation (input) to the network and the network activity (output) are shown in two windows. Iterate the network a few steps with step and observe how the activity changes.

The time shown is the total simulated time since the last reset of the network. A minus in front of the time means that the network has not reached stability yet. (It can take a few steps more to reach stability, even after the activity doesn’t seem to change any more. This is because the belief values are functions of internal state variables which are updated according to a differential equation. It takes a few steps for these states to converge.)

The activity above behaves quite uninterestingly. Instead, check what happens with some noise added to the inputs. Reset the network, set the noise level to about 0.4 and stimulate with a pattern again. Take one step at a time and observe the results. Continue until the output stops changing (and the minus in front of the time disappears. If it is too slow after a while, you might take more than one step at a time).

- Is the final pattern identical with the stimulated pattern? If not, why?
- Check whether it helps to remove the stimulus (clear) and continue with stepping to stability. (If the output is identical to the stimulated pattern even before this last step, increase the noise, or stimulate with another pattern.)

The command recall does all the steps above, i.e. stimulates with the current pattern, iterates to stability, removes the stimulus and iterates to stability again.

You can also reset, step and set input to individual units using the mouse buttons. Reset the activity by clicking the middle button in one of the two windows, clear the input by clicking the middle button with shift. Now, stimulate two or three units by clicking on them with the left button in the input window, and step the network with the right button (do several steps). Shifted right button does recall instead of step.

5 Noise tolerance

This task concerns how noise affects the recognition of patterns. Reset the noise level to zero, and train the network with all the patterns in symbol.pat (if they aren’t already trained after the previous task).

- Check that all patterns are correctly recalled (with test, note that the display format depends on whether a file name is given or not).
- Increase the noise level and see what happens. Run a few times with each noise level, to get good averages. If it is too slow, disable the X windows with the command xdisplay.
- Draw a diagram that shows how the percentage of correctly recalled patterns depends on the noise level.
6 Decision

Choose two of the patterns in the trained file, and make a sequence of mixed patterns from them, by gradually exchanging more and more of one of the patterns with parts from the other pattern. This can for example be done in Emacs. (For the lazy: There is already such a file, mix.pat.)

Train with the patterns in symbol.pat. Set the noise level to 0. (You can either enable your X windows of the network again, or use the output in the command window.)

- Let the new hybrid patterns be input to the network, by doing test on that file. What happens?

Alternatively, it is possible to do the mutation of patterns manually in the X windows. Start by stimulating one complete pattern, and note the relaxation time. Then mutate it by removing activation from a few of its units, and activate some units of another pattern. Note the new relaxation time, and the resulting pattern, and continue in the same way until all units are changed to the second pattern.

- Draw a diagram showing the relaxation time, as a function of how large the part of the hybrid which comes from one of the original patterns (much of the time is spent doing “housekeeping”, so the relative difference is not so big). Also note which pattern results in each case.

- What conclusions can be drawn?

7 Storage capacity

The file random200-100.pat contains too many random patterns, to be stored in the network without errors in recall. The task here is to find out how the error frequency depends on the number of stored patterns. Here it is wise to disable the X windows, since it will take too much time to update them.

- Use learn n and test n, and draw a diagram of percent correctly recalled patterns as a function of the number of trained patterns.

8 Classification

Train with the symbolic animal database in the file animals.1 (use dblearn, since it is a symbolic file). The file is included at the end of this instruction. Now you will get five windows describing the network. The “Network” window shows the output (or “belief” values) of all units in the network. There are two “property” windows and two “Class” windows, one of each sorted alphabetically, and the others sorted by belief value. If they take up too much space, one alternative is to close the alphabetically sorted “Class” window, and the “Property” window sorted by belief. (You can see which is which if you stimulate some unit and step the network.) There are two units for each property and class, one positive and one negative. This is to make it possible to explicitly state that a property does not occur, rather than “no indication” as zero stimulation of the positive unit would mean. The left unit in a pair is the positive one. Clicking it with the left mouse button toggles the pair between “yes”, “maybe” and “no”. Clicking the right unit, toggles the other way around.

Stimulate one or a few of the properties in the database (with input or the left mouse button) and see what happens with relaxation (using either recall or...
shifted right button). Check both by stimulating a set of properties that indicates one animal unambiguously, and some set that can belong to a couple of animals. (Don't forget to reset the network with both clear and reset between the tests.) Also stimulate some erroneous property together with the correct ones for the animal you think of.

- What happens in the different cases?
- Repeat the different tests above, but iterate just one step (right mouse button) instead of to stability. Is anything different?

After one step of relaxation in the Bayesian neural network, the activities of the units represent estimations of probabilities for the different classes given the stimulated properties. For a given unit, such a probability can be regarded as confidence information for the hypothesis that the input data belongs to the class which the unit represents. By relaxation to stability the output is forced out to a “corner”, even if there isn't enough information for an unambiguous classification.

9 A query-reply system based on the neural network

The command query makes the system generate questions from the state of the network. First clear all input to the network, think on an animal in the database, and then start the query-loop. Answer the questions in accordance with the animal in the database. Look at the end of this instruction. (It could be wise to check in the database that the animal really has the properties you think it has, because that is not always the case with this database.) 1 means “yes”, 0 “no” and “-” means “maybe”. Just a press on return will leave the loop. It is possible to continue the loop by repeating the command query. The systems hypothesis, if there is any, is displayed in brackets.

You can of course answer questions by clicking with the left button in the “Property” window. The left unit in a pair means “yes” and the right unit means “no”. A second click on the same unit means “maybe” or “don’t know”. To get the next question, use the right button. The middle button starts a new query session, which means that all answers so far are erased.

- Count the number of questions needed to reach a hypothesis, if all the answers are in accordance with the database.
- Clear the input to the network again, and think on a new animal, but make an erroneous answer on some question. When the answers isn’t in accordance with the state of the network, the system generates an inconsistency question. This means that the system asks again on a property that is likely to be erroneously answered, so the user gets a chance to correct a possible mistake.

Anytime during question generation it is possible to leave the query-loop and stimulate something not asked for, and then resume the loop (or if you use the mouse, just click on some other property than the one asked for). Also try with stimulating a few properties initially that hold for 2 to 4 animals, start the query-loop, and confirm that the query system can adapt the questions to known facts.