CSE 4392 / CSE 5392 - *Smart Home Technologies*

Homework 3- Spring 2006

Due Date: April 13 2006, 3:30 pm

Components

- Smart Home Simulator (Running on 129.107.12.179)
- C++ Software Packages:
  - SNNS Neural Network Simulator - See *http://www-ra.informatik.uni-tuebingen.de/downloads/SNNS/SNNSv4.2.Manu* for documentation.
  - C++ HMM code - See *http://www.cs.ualberta.ca/l̃indek/hmm.htm* for documentation.
- Java Packages:
  - JavaNNS Neural Network Simulator - See *http://www-ra.informatik.uni-tuebingen.de/software/JavaNNS/manual/Jav manual.pdf* for documentation.
  - jahmm HMM code - See *http://www.run.montefiore.ulg.ac.be/f̃rancois/software/jahmm/doc/jahmm-userguide-0.5.0.pdf* for documentation.

Problems marked with a * are required only for students in the graduate section (CSE 5392). They will be graded for extra credit for students of CSE 4392.
The goal of this assignment is to set up and evaluate predictors to track the location of the person within the Smart Home Simulator. For this assignment, you will construct Neural Network as well as Hidden Markov Model predictors. As part of this you will have to design initial models, train them with data sequences constructed from the output of the Smart Home Simulator, and then evaluate the precision of the resulting predicttors.

Smart Home Simulation

The Smart Home simulator is the same that was used for the second assignment and generates an event sequence which includes a time stamp, the device identifier and the state it switched to. The general format looks as follows:

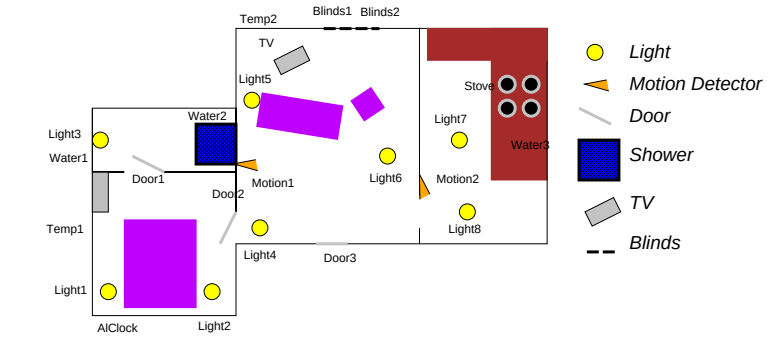
```
2006-03-01T07:00 AlClock on
```

The sensors in the environment are shown in the following map
and include the following:

Alarm Clock [on off]AlClock

Lights [on off]Light1, Light2, Light3, Light4, Light5, Light6, Light7, Light8

Faucets/Shower [on off]Water1, Water2, Water3



Door Crossing Sensors [on off]Door1, Door2, Door3

Motion Detectors [on off]Motion1, Motion2

Temperature Sensors [60-90]Temp1, Temp2

Intruder/Fire Alarm [on off]Alarm1, Alarm2

The simulator runs on port 8765 on $129.107.12.179$. Upon connecting to it, issuing the command run will start the simulation. During the simulation, events are delivered one at a time and issuing $next$ to the simulator causes it to issue the next event. Sending end to the simulator will end the simulation and disconnect. In addition to these commands, instructions to trigger a sensor or an alarm can be sent to the sensor in the form $SensorName\ State$, e.g. $Alarm1\ on$ to set off the intruder alarm.
There is a simple sample C and $Java$ client under Homework 2 on the web page.

Training sequences:

As a first step of the assignment you will have to derive location sequences from the data generated by the simulator. For this you should use the motion sensor and door sensor events to determine what room the person has moved to.
Then you should use this to write an output file containing a sequence of locations that you can use for training purposes. Note that for the Hidden Markov Model part you should generate a file that contains at least 20 sequences each of length at least 100.

Data Representation Hints

Neural Network:

Neural Networks make only a single prediction based on the current input to the network. As a result, the input representation has to contain all the data necessary from the past location observations to predict the next location. Since the next location in this example is not completely determined by the current location of the inhabitant, the input representation for the network has to include not only the current location but also a set of previous location observations. Each of the used location observations would then correspond to an input node

of the Neural Network and to obtain the current prediction, the input nodes would be set to values corresponding to the location estimates associated with each of the nodes.
For Example, if the last 3 location estimates were used as the input to the network and the observation sequence so far was ..., 1, 2, 3, 4, 5 , then the 3 input nodes would be set to a value of 3, 4, and 5, respectively. The output of the trained network, in turn, would represent the prediction of the next location. Note that if you are using sigmoid units on the output of the network, the output would be inherently limited to values in the range [0..1] and you would have to translate the output to the particular prediction (e.g. By multiplying the output by 7 and then using the integer component of the scaled output as the prediction).
To generate training data, you will have to translate the observed sequences in the data file into training pairs (x, f(x)), where x is the input to your network and f(x) is the desired output. For example, in the example used previously (including the scaling for sigmoid units), the sequence 1, 2, 3, 4, 5 would translate into the training pairs ((1, 2, 3), 0.571) and ((2, 3, 4), 0.7143) .

Hidden Markov Model:

Hidden Markov Models build a model of the process that generated the observations. Training data for the HMM learner is therefore a set of sequences of observations. As a result, very limited processing of the data provided in the datafiles is needed to create the input to the HMM learner (what is needed is largely reading the information from the file and inserting it into the observation sequence vector used by the HMM package).

Generating Data

1. Generate location sequences from the Smart Home Simulator and save them to files.

Part A: Setting up and training a Neural Network

In this part you are to design a neural network for the prediction task. The network will be built inside the neural network simulator and be trained on data recorded from Simhouse. For this assignment you do not have to integrate the network directly into Simhouse, rather you will record the sequence of occupancy and lighting information into a file and use this file to train and evaluate the predictor.

2. Familiarize yourself with the Neural Network Simulator and design an input and output representation for the Neural Network predictor. (Remember that the input to the Neural Network has to be a description of the current situation, potentially containing past events and the output has to represent the prediction. All this information has to be encoded into numbers to enable the use of a Neural Network)

3. Translate the recorded data from Simhouse into a set of training pairs for the Neural Network simulator (i.e. Into pairs of input /output values using the representation derived above) and build two different sized Backpropagation Neural Networks in SNNS (or JavaNNS) and train them with the training data. Record the learning curves as a measure of the network's performance.

Turn in: (1) your training data set, (2) your Neural Network , (3) a snapshot of the learning curve and of your network, and (4) a short description of your input/output representation and of your experiences when designing the predictor, including a comparison of the performance of the two networks you used.

Part B: Train a Hidden Markov Model predictor

In this part you are to learn a Hidden Markov Model for prediction purposes. The predictor will again be trained on the recorded SimHouse data and you will make use of the HMM algorithm library provided by hmm or Jahmm to learn the model and to evaluate the best prediction.

4. Make yourself familiar with the functions provided by the HMM package and set up a program to learn a Hidden Markov Model to describe the behavior of the inhabitant in terms of lighting and locations (you will be using the Baum-Welch algorithm to learn the model).

5. Translate the recorded data from Simhouse into a set of training sequences for the Hidden Markov Model program (i.e. Into a set of sequences of observations) and learn a Hidden Markov Model that models the training data.

6.* Extend the program to use the model to predict the next location of the inhabitant. (You will be using the Viterbi algorithm to determine the current state within the learned model and then predict the next location as the most likely location event to follow this state).

7.* Compare the performance of the HMM predictor to the one of the Neural Network predictor.

Turn in: (1) your training data set, (2) your Java code, (3) a snapshot of the HMM that your system learned, and (4) a short description of your input/output representation and of your experiences with HMM modeling. Graduate students only: (5) a comparison of the performance of the Neural Network and the HMM predictors.