COMP 232—Knowledge Based Systems—Spring 2001

April 8, 2001

This document reflects the current contents of the web page.

We will study several special topics in AI. So far, the list is as follows.

- Speech recognition with a heavy focus on HMMs.
- Combining logical and decision-theoretic planning.
- Logical planning with open worlds and information gathering.

Announcements

Fri_3/9: I have re-written the Testing section of the HMM project description below, and added a section to my ongoing document on Hidden Markov Models about evaluating the trained model.

Other WWW Documents

- My introduction to Hidden Markov Models.

1 Old Announcements

These old anouncements appear in reverse chronological order.

Fr 2/9: I have added important information about how the trainer should operate. See the very end of my notes on HMMs as well as the section on testing in my project description.

2 Administrative Information

Instructor: Jim Schmolze, office: Halligan 101 (EECS Office), phone: 7-3217, home page: http://www.eecs.tufts.edu/ sc email: schmolze@eecs.tufts.edu.

Office hours for Prof. Schmolze:

- Immediately after each class for at least 30 minutes.
- Immediately after COMP 131, which ends at 3:45 pm on Tuesdays and Thursdays, for at least 30 minutes.
- Or by appointment (the best method is to make an appointment with EECS office staff).

3 WWW Resources

The main resources (for now) can be found on the Web pages from other courses at Tufts.

- From COMP 131, Introduction to Artificial Intelligence: General resources for AI.

4 Speech Recognition & HMMs

4.1 Assignments

Wed 1/24: Read the section on speech recognition in the Russell & Norvig text.

Wed 1/31:

1. Read the article "An Introduction to Hidden Markov Models" by Rabiner and Juang, IEEE ASSP Magazine, 3(1), January 1986, 4-16.
2. Along with the above article, read my notes on this introduction.

Wed 2/7:

1. Try to develop the formulas that I failed to do in my notes on the HMM tutorial.
2. Start the HMM project (see below).

Wed 2/14:

1. Try to develop the formulas that I failed to do in my notes on the HMM tutorial.
2. Start the HMM project (see below).
3. Read the first 3 sections of "Stochastic Dynamic Programming with Factored Representations" by Boutlier, Dearden and Goldszmidt.

Wed 2/21: *Class canceled.*

Wed 2/28: Prepare your results ffrom the HMM project.

Wed 3/7: Be prepared to discuss section 4 of the Boutlier *et al* paper.

Wed 3/14: Work on wrapping up the HMM project.

Wed 3/21:

Wed 3/28:

Wed 4/4:

Wed 4/11:

Wed 4/18:

Wed 4/25:

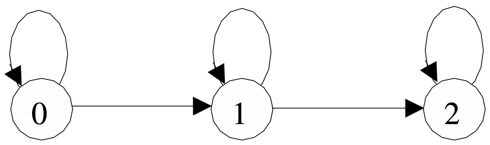
5 Projects

5.1 HMM Project

Let's build and train a very simple HMM to see if the theory really works. You can do this in any computer language you choose. C (or C-like) languages have the advantage of speed, which may prove important. You may work in pairs if you wish.

5.1.1 Simulate an HMM

First, build an HMM simulator so that you have a source of "observations" and a target to compare against. To do this, we will build an MM, and then we will proceed to "hide" it from the HMM trainer. I suggest creating code that can manage N states, numbered $0, \ldots N-1$, and M different kinds of observations, numbered $0, \ldots M-1$, and which produces observations of length T. In other words, N, M and T are parameters of your Markov model. Initially, I suggest that we attempt to train models where $N = 3$, $M = 4$ and T varies from 3 to 5. Here is an example.



If this above model splits its transition probabilities 3:1 in favor of going "right", then the transition probability matrix is as follows.

$a_{i,j}$	0	1	2
0	0.25	0.75	0
1	0	0.25	0.75
2	0	0	1.0

Let's assume that this model always starts in state 0.

π_i	0	1	2
	1.0	0	0

Finally, there are four observations, 0, 1, 2 and 3, with the following probabilities per state.

$b_i(k)$	0	1	2	3
0	0.5	0.5	0	0
1	0	0.5	0.5	0
2	0	0	0.5	0.5

To build the MM, use a *Monte Carlo* simulation. First, you'll need a random number generator. In C, use the random() function (see manual page), seeded by srandom() using the data from a call to time() (see manual page).[1] Once random() is seeded, you can easily define a function randomint(lb, ub) that returns a random integer x in the range lb$\leq x <$ub.[2]

```
int randomint (int lb, int ub) {
   return lb + random() % (ub - lb);
}
```

You can also define a function that returns a random floating point number x in the range $0 \leq x \leq 1$.

```
double randomreal () {
```

[1] Lisp users should call the built-in function random–see the documentation. To seed the random function in Lisp, execute the following:
(setf *random-state* (make-random-state)).

[2] I will use C-like notation. Lisp users should be able to translate easily.

```
    return ((double) random()) / RAND_MAX;
}
```

To generate a random choice over a given distribution, simply do the following. Assume we wish to generate the initial state according to π.

1. Generate a random number x such that $0 \leq x \leq 1$.
2. $i \leftarrow 0$.
3. $x \leftarrow x - \pi_0$
4. While $i < N$ and $x > 0$ do
5. $\quad i \leftarrow i + 1$
6. $\quad x \leftarrow x - \pi_i$
7. $\quad$ End while loop.
8. If $i < N$ then i is the randomly selected state else *error* (sum of probabilities is < 1).

Use the above method to:

- select the initial state of the HMM,
- select the next state, and
- select the observation that appears in each state.

You are now ready to define a function that produces a single observation sequence.

function newsequence (N, M, π, a, b, T)

1. Create an HMM according to the given model.
2. Let s be the initial state, chosen according to π.
3. Let O be the empty sequence of observations.
4. While $length(O) < T$ do
5. $\quad$ Let o be the observation for current state s, chosen according to b_s.
6. $\quad O = concatenate(O, o)$ (i.e., add o to the end of O).
7. $\quad s \leftarrow$next state according to a_s.
8. $\quad$ End while loop.
9. Return O.

5.1.2 Build an HMM Trainer

Using the formulas in the tutorial and in my notes, write code that will adjust a model given a starting model and an observation sequence. Then start off with a random model and see how easily it converges to the correct model. You'll need some difference measure between models. I suggest using the sum of the squares of all the model parameters.

The issue of how to test the model that results from training is not an easy one. See my HMM notes for a discussion.

5.1.3 Testing

In order to evaluate how well our HMM trainer does, we need a test to determine how well our trained model mimics the original model. Please refer to the HMM document for a discussion. In it, I define a quantity called $\nu(\lambda\,\lambda')$ that measures the accuracy to which the trained model λ' mimics the original model λ. We wish to know how various parameters of the training process affect $\nu(\lambda\,\lambda')$.

Our HMM trainer has a number of parameters. Two parameters refer to the Markov model that we use as the "actual" process, i.e., the one being simulated.

N: The number of states in the Markov model being modeled.

M: The number of observations in the Markov model being modeled.

There are at least three parameters controlling the training process.

T: The length of each observation sequence.

L: The number of observation sequences.

C: The maximum number of iterations allowed for the trained model to converge. This may be irrelevant if convergence is fast.

For your evaluation of HMM training, I would like to see a 4-dimensional (maybe 5) table of results showing statistics on $\nu(\lambda\,\lambda')$ for various values of N, M, T and L (and maybe C). For each table entry, you must do a number of trainings against a number of λ's that are generated randomly. In this way, we will have some measure of the HMM training process for those parameters. For example, set $N = 3$, $M = 4$, $T = 5$ and $L = 20$ and run, say, a dozen experiments. In each experiment, generate a new λ rendomly, train a λ' and calculate $\nu(\lambda\,\lambda')$. For these, say, dozen values of $\nu(\lambda\,\lambda')$, report the mean and standard deviation. Now do this for a large variety of values of the variables explained above. With luck, we will see some trends in the data.