NumberLand: An Adaptive Tutoring System Progress Report, 11/21/2014

Nathaniel Raley STC, Fall 2014

The four principal goals of this tutoring system are (1) to define order and type of questions to ask, (2) to determine student understanding and misconceptions, (3) to decide when to provide hints and what hints to provide, and (3) to conclude when the student fully understands the subject.

I began by writing a pilot program to make sure that the project was feasible given my c++ ability and the time constraints. This program was a simplified mental arithmetic tutor: when run, it prompted the user to choose a level (additional, subtraction, multiplication),generated two random numbers, and asked the user to perform the selected operation on those numbers (if they chose addition, say, they might get “48+87=\_\_\_”). If correct, it tells them they are correct, generates two more random numbers, and creates a new question using the same operation. If they are incorrect, it tells them that they are incorrect and asks them to try again.   
 This program was fairly basic but it worked without any trouble and I have used it as the backbone of my current program, which I will now describe in its present state. My program is meant to guide elementary school students through the addition/subtract facts learning process. When you run the program, it looks like this:

Enter your user name:

After you enter your user name, you see

Welcome to Numberland Nathaniel! Let's learn something!

Choose your level (1-8) or exit (9)

1. Counting up
2. Doubles
3. Doubles +/- 1
4. How far apart
5. Break Down
6. 10 Combos
7. Working with 10
8. Crossing over 10
9. Exit Program

The program starts off at level 1, where students have to add just 1 or 2 to a single digit addend (e.g., 5+1, 3+2). The program keeps track of (1) the number of questions the user gets correct and incorrect at each level and (2) the average amount of time the user spends on each question at a given level (tossing out any observations above 1 minute). There are three criteria for advancing to the next level: the student must answer 10 questions correctly in a row, the student much average less than 5 seconds per question, and the student must have 95% correct out of all attempts. If they satisfy these criteria, the topic is considered mastered and they can move on to Level 2.

Level 2 teaches students to double a number (5+5, 7+7), and the same advancement criteria applies. Level 3 is “doubles +/- 1” (5+6, 7+8). Level 4 asks students how far apart two numbers are (e.g., how far apart are 3 and 9?). Level 5 is similar to Level 4 except it frames the questions in terms of equations (9 - \_\_ = 3, 4 + \_\_ = 7). The connection between Level 4 and Level 5 is emphasized in the teaching portion (see below). Level 6 trains students to quickly identify combinations of numbers that make 10 (7+3, 8+2, 9+1...), Level 7 teaches students about place value (10+7= \_\_, the 10s place doesn't change), and Level 8 teaches students to apply skills from Levels 1-7 to approach more difficult addition and subtraction problems. Level 8 is the most difficult level. Students have to answer questions like 13-7 by thinking “how far is 10 from 7? (3); how far is 10 from 13? (3); so how far total is 7 from 13? (3+3=6).” Though it seems really basic, training these skills in an ordered fashion helps children develop good number sense and lays a strong foundation for more difficult mathematics concepts. When a student has mastered all 8 levels, they start back over at level 1 and work through all the same problems except this time, the numbers are double digit numbers instead of single digit numbers; the implementation was very straightforward – I just changed from modulo 10 to modulo 100.

**Next Steps**

The foregoing description covers most of the work I have done for the project thus far, and there are many more things that remain to be accomplished. For instance, so far the program can only support a single user. I want to give the program multi-user capabilities so that it can keep track of multiple students over time; they would simply have to enter their user name to restore their previous session. This should not be very difficult to accomplish; a simple text file could link the user name to the average time spent per question per level, number of correct/incorrect answers, number of total questions, etc. Indeed, I have already programmed it so that, when you finish with a level, a textfile is generated with all of the information about that session. Below is sample output from the user data which was dumped to the file “user\_name.txt”:

nathaniel@ThinkPad:~/Desktop/ITSproject/tutorproject$ cat user\_name.txt

Nathaniel level = 7 % correct = 71.4286 avg time (seconds) = 2

The only thing keeping this from being multi-user is not having a good look-up function written, and I'm beginning to realize that I could implement this very easily.

Also, I would like to be able to break problems down into individual steps for students if they are struggling, and I want to build a feature where students can ask for hints if they get stuck. I will need to be sure to factor the hint requesting behavior into the criteria for mastery. This was part of my original proposal, but I haven't been able to get it working; I need specialized hints for each question type, and I wasn't anticipating the amount of trouble this would be. Still, I'm pretty happy that my program is able to gather information about average time spent per question; this is perhaps the biggest indicator of student understanding for basic mathematics concepts that can be gotten through the command line.

Also I need to improve the pedagogical component; as it stands, I provide an instructional overview at the start of each section which the student is to read before beginning to answer the questions. However, the way it is currently set up, the student cannot access this material over again. Originally, this was because I was going to include hints and instructional feedback, but this has not been implemented. When I do create a “hint library” I will re-present instructional material to remind students of each mini-lesson. Making sure students don't abuse the hint feature will be important, as I'm sure it will provide them with a way to cheat the system.

Additionally, I feel that I would be helpful to add a statistics feature which calls gnuplot from within the program, charts the student's progress over time, and reports relevant descriptive statistics on request. This would be especially useful for the multi-user version of the program (which still needs to be developed), because it would give the instructor important summary data (e.g., class level information) about the students' progress. I know this is not difficult to do, and I feel relatively comfortable with gnuplot after completing homework 8. Finally, I really want to use Bayes networks instead of a simple 3-criteria mastery check; I've read a lot about how to implement them, but I didn't want to spend all my time on this single technical feature and not be able to pull together a working program. Now that my program is working, I want to go back and try to use the evidence obtained from students (time spent, correct/incorrect) to update the probability estimate that a student understands each concept. I would need to obtain estimates of prior probabilities for each level too, but I could just use .5 or perhaps give a pretest.   
To end with, here is some sample I/O for level 7 (crossing over 10); notice how it gives questions in both formats ( x+y and x - \_\_\_ =y):

...

You are correct!

Avg. time = 2 seconds | % correct =100

Enter N for a new problem (Enter Q to return to main menu)

n

Crossing over 10:

8 + 4

3

Try again!

12

You are correct!

Avg. time = 2 seconds | % correct =80

Enter N for a new problem (Enter Q to return to main menu)

n

Crossing over 10:

7 + 5

12

You are correct!

Avg. time = 2.4 seconds | % correct =83.3333

Enter N for a new problem (Enter Q to return to main menu)

n

Crossing over 10:

10 - \_\_\_ = 5

5

You are correct!

Avg. time = 2.33333 seconds | % correct =85.7143

Enter N for a new problem (Enter Q to return to main menu)

q

Your percent correct was 85.7143%

Choose your level (1-9)

1. Doubles

2. Doubles +/- 1

3. 10 Combos

4. Working with 10

5. How Far Apart?

6. Break Down

7. Crossing over 10

8. Quiz

9. EXIT PROGRAM

9

Thanks for playing!

nathaniel@ThinkPad:~/Desktop/ITSproject/tutorproject$