

# Lab 4

CS 3793 – Fall 2008  
Tom Bylander, Instructor

assigned November 14, 2008  
due midnight, December 5, 2008

In Lab 4, you will program an algorithm to play the target game. This game is a new addition to my agents collection. Your algorithm will be based on using probabilities to solve the game.

## Agent

Your task is to write an agent program to interact with the the target environment program, which is implemented by the `target` command in `/home/bylander/agents/linux/bin`. Assuming that you have one of the following shell commands (depending on your shell):

```
setenv A /home/bylander/agents/linux/bin
A=/home/bylander/agents/linux/bin
```

then `$A/target -h` will describe how you can interact with the target environment.

You can play the game by running `$A/target -v`. Enter two digits between 1 and 8 (no spaces) to select a square, e.g., 27, 83, 41, and 18. If there is a target at that square, then your move is a hit. If there is a target in a neighboring square, then your move is a near miss with a probability of at least 1/2. Otherwise, your move is a miss.

Here is a more detailed description. You have to find four targets on the board. Your means are simple—you select a square and then you get feedback about that square and neighboring squares. There are three possible outcomes:

1. Hit: The square you select is a target. Other than reducing the number of remaining targets by one, there is no information about neighboring squares.
2. Near Miss: If a square is a target, then it causes a near miss in a neighboring square with probability 1/2. That is, a near miss implies that at least one of its neighbors is a target. If  $n$  neighbors are targets, then a near miss occurs with probability  $1 - 2^{-n}$ .
3. Miss: If the outcome is not a hit or a near miss, it is a miss.

## Implementation

`/home/bylander/agents/linux/src/target/trandom.c` implements a player that performs random moves. To select better moves, you need to determine which squares are more probable to be targets.

## Calculating Probabilities

`/home/bylander/agents/linux/src/target/target-probabilities` contains useful information about prior and conditional probabilities for this game. These are numbers obtained from generating millions of simulated games, which appeared to be sufficient to be accurate

to 4 decimal places. For prior probabilities, you will want to use the priors that distinguish between types of squares. I did not generate different conditional probability tables for different types of pairs of squares; note that this would require 8 tables.

For each unknown square, your program should calculate the probability of different hypotheses (that it is a hit, near miss, or a miss) given the evidence (knowledge about other squares). Recall that:

$$P(H | E) = \frac{P(H, E)}{P(E)} = \frac{P(E | H) P(H)}{P(E)}$$

A Bayesian network would be a correct way to accomplish this calculation, but would require significant resources, both yours and the computer. Instead, I recommend that you assume conditional independence and only consider neighboring squares. Suppose  $E$  includes knowledge of four neighbors  $E_1, E_2, E_3, E_4$ . Then  $P(E | H)$  can be approximated by:

$$P(E | H) \approx P(E_1, E_2, E_3, E_4 | H) \approx P(E_1 | H) P(E_2 | H) P(E_3 | H) P(E_4 | H)$$

For example, suppose that we know that the neighbors of an unknown square include two misses, one near miss, and one hit. Then:

$$P(E | \text{hit}) \approx (0.4101)(0.4101)(0.5423)(0.0476)$$

$$P(E | \text{near miss}) \approx (0.5526)(0.5526)(0.2717)(0.1757)$$

$$P(E | \text{miss}) \approx (0.8225)(0.8225)(0.1431)(0.0344)$$

To complete the (approximated) calculation of  $P(E | H)P(H)/P(E)$  for the three different hypothesis, use the appropriate priors for  $P(H)$  and calculate  $P(E)$  as:

$$P(E) = P(E | \text{hit}) P(\text{hit}) + P(E | \text{near miss}) P(\text{near miss}) + P(E | \text{miss}) P(\text{miss})$$

## Selecting a Move

It seems natural that the best move would be to select the square with the highest probability of  $P(\text{hit} | E)$ . However, good evidence is needed, too, i.e., maybe selecting a square with a high probability of  $P(\text{near miss} | E)$  would be good, too. A way of balancing this would be to compute the amount of information that an unknown square has. If  $p_1, p_2, p_3$  are respectively  $P(\text{hit} | E)$ ,  $P(\text{near miss} | E)$ ,  $P(\text{miss} | E)$ , then the amount of information can be calculated by:

$$-p_1 \lg(p_1) - p_2 \lg(p_2) - p_3 \lg(p_3)$$

where  $\lg$  is log base 2.

Selecting the move with the highest information based on calculating probabilities as described above yields a player that averages about 35 moves a game, which is a lot better than the random player with an average over 50. There are probably many ways to do better.

## Competition

The labs will be tested on 100 target games. The best performing lab will receive 30 points extra credit, second place will receive 20 points extra credit, and third place will receive 10 points extra credit.

## Turning in Your Lab

Somewhere in your directory, you should create a `lab4` subdirectory. This directory *must* have a `Makefile` that compiles and links your agent program, which *must* be named `lab4`. Any source code that is linked to these programs *must* be in this directory. That is, you should be able to copy the files in your `lab4` directory to another directory and be able to run the following command sequence.

```
setenv A /home/bylander/agents/linux/bin or A=/home/bylander/agents/linux/bin
/bin/rm lab4 *.o
make
$A/interact $A/target ./lab4
```

Zip or tar the entire directory and submit it using WebCT. If you submit multiple times, only the last one is kept. Be sure that WebCT has registered your submission.

Provide a brief report describing the performance of your program and any differences between your program and the suggested implementation above. It should describe its performance on a sample of games both in terms of number of moves as well as how it appears to behave (use "`$A/target -v`" to see how it behaves).

Your grade on the lab will depend on the performance of your program. Assuming a reasonable report, your grade will be assigned consistent with the following.

1. A score of 80. Your program averages 40 on the 100 games.
2. A score of 90. Your program averages 35 on the 100 games. ‘
3. A score of 100. Your program averages 30 on the 100 games.