Bayesian Lab

October 5, 2004

The purpose of this lab is to give you experience with Bayes’ rule and to help you understand how it can be useful in the final project for this class. You will create a Bayes estimator that efficiently estimates the locations of five enemy agents in the BZFlag world. Your estimates for the enemy agents’ positions should get better over time. Your code should show the locations that your estimator believes to be the correct locations of each agent, and these estimates should approach the “true values” of the enemies’ positions on the playing field.

1 What to Turn In

To pass off this lab, you will turn in the following electronically:

- A declaration of time spent by each lab partner
- A write-up explaining how you implemented your Bayes estimator and what you learned from this lab.

You will also need to demonstrate to one of the TAs that your code works. Sign up for a time on the door of the office, or just come by while the TA is there.

2 The Distribution

The distribution for this lab differs from the previous distributions in the following ways:

- The agents have been made kinder and friendlier. Instead of tanks, we’re now using spheres.

- The enemy agents will all be placed on the field at positions where both coordinates are multiples of eight. That way there is a finite and relatively small number of locations where they can be located. Since there are only a finite number of possible enemy locations, this will make the selection of your prior (for the Bayesian computations) much easier.

- When you look at the cur_position values for enemy agents, instead of getting their actual positions, you will be getting a “noisy” value— a coordinate drawn from a Gaussian distribution centered at the true value and with a standard deviation of 50 units.

- A new code file called “BayesianAgent.cxx” is included, with some special functions, which will be explained below.

For these reasons, it will be necessary for you to download the new distribution and do a full compile, as with previous labs. Again, to do the full compile, go to the root directory of the distribution, and enter the following:

```bash
./configure
make
```
3 Description

After compiling the distribution, you will need to create the “ta\_bot” executable from the “SleeperAgent.cxx” code (as you’ve done many times before, just create a symbolic link called “470bot.cxx” to that code file and type “make” and then rename the bzflag executable to ta\_bot). As in the Search Lab, the enemy agents will be the green team, and there will be five of them. Your agent will be the red team, and you will have just one agent on the field. Neither your agent nor the enemy agents will actually move in this lab (like in the Search Lab). Your job is to use Bayes’ rule to estimate the positions of each of your five enemies based on noisy data, and to show those estimates on the field.

The “BayesianAgent.cxx” code has the following functions provided for you:

```c
void clearlines(); // Clears all lines drawn on the field
int numFromCallsign(const char *callsign); // Returns an integer for an enemy
// if you pass its callsign
void showEstimatedPosition(int enemyNum, double xPos, double yPos); // showEstimatedPosition draws a colored X on the field
// showing where you believe a particular enemy to be
```

4 The Bayes Estimator

The BZFlag playing field will be 800 x 800 units, with each coordinate (x, y) running from -400 to +400. Every enemy will be located at a position where both coordinates are multiples of 8. Thus, you can imagine a grid of 100 x 100 points (ok, it’s actually 101 x 101) which are possible locations for each enemy. What your program will do is maintain five 2-D arrays of probabilities that a given enemy is found at each of those possible points (one 2-D array for each enemy). When you get that working, you’ll need to change the granularity of your grid to include just the points whose coordinates are multiples of 16 and then again to include all the points whose coordinates are multiples of 4. So your code should be flexible enough to handle arrays with different granularities.

An alternative way to implement your grid for this lab is to have it always include all the points whose coordinates are multiples of 4 (so it would be 201 x 201). Then, when you initialize your priors for the 8 and 16 cases, you can simply set the values for those points that are not multiples of 8 (or 16) to zero. This approach will give the same results as the one described in the previous paragraph.

4.1 Bayes’ Rule

Recall that Bayes’ rule looks like this:

\[ P(s|e) = \frac{P(e|s)P(s)}{P(e)} \]

For the purposes of this lab, s represents a possible state of the world, such as “enemy #2 is located at position (24, 80).” The e here represents the evidence that we have been given, such as “my sensors tell me that enemy #2 is located at (33.4, 67.15).”

The expression \( P(s|e) \) is called the posterior, the result we get after the computation is complete; \( P(e|s) \) is called the likelihood, and \( P(s) \) is the prior, or our belief about the probability of state s before we gather any evidence. \( P(e) \) is in the equation as a normalizer, in order to make all of the probabilities over all possible states \( s \) sum to 1.

4.2 Updating Your Estimates Using Bayes’ Rule

You can calculate the value of the likelihood \( P(e|s) \) based on the knowledge that the noisy readings for enemy location are generated from a Gaussian distribution with a mean \( \mu \) at the true value of the agent’s position, and a standard deviation in each coordinate of 50 units. Since this is a 2-dimensional Gaussian, its probability density function looks like this:
$P(e|s) = \frac{1}{\sqrt{(2\pi)^2 \text{det}(K)}} \exp\left(-\frac{1}{2}(e - s)^T K^{-1}(e - s)\right)$

A few important notes about this function:

- Both $e$ and $s$ are 2-dimensional vectors (they have two values). They are column vectors, so they look like this:

$$\begin{pmatrix} x \\ y \end{pmatrix}$$

- $K$ is the **covariance matrix** for the noisy readings. Since there is zero-mean noise with standard deviation 50 in each coordinate, the $K$ for this lab is:

$$\begin{pmatrix} 50 & 0 \\ 0 & 50 \end{pmatrix}$$

- The expression $\text{det}(K)$ means the determinant of matrix $K$. In this case, $\text{det}(K) = 2500$.

- $K^{-1}$ is the inverse of matrix $K$. In this case, it happens to be:

$$\begin{pmatrix} \frac{1}{50} & 0 \\ 0 & \frac{1}{50} \end{pmatrix}$$

- The $\exp(x)$ function is just $e^x$ (but don’t confuse this $e$, which is the constant 2.7182818... with the $e$ that represents the evidence!

So that’s how you can calculate your likelihood $P(e|s)$ for each possible state $s$. The prior, $P(s)$, is simply carried over from the posterior of the previous iteration. In other words, $P(s)$ at iteration $t$ is the value of $P(s|e)$ from iteration $t - 1$. For the first iteration, you should initialize all your priors $P(s)$ to the same value, in such a way that the sum over all priors is 1. So calculate the total number of possible states $n$, and initialize the priors all to $1/n$.

The normalizer value, $P(e)$, does not actually have to be calculated. We recommend that instead, you calculate the posterior $P(s|e)$ for each state $s$ simply as $P(e|s)P(s)$ and keep a running total of all the values you calculate, then do another pass through the array and divide each value by the total. This has the needed effect of normalizing the table so that the probabilities all sum to 1.

At each time step, you will gather data about each of your five enemy agents, based on the noisy readings given to you in the current position values of the player’s struct, and then update each of your five 2-D arrays of probabilities using Bayes’ rule as described above. Then you will go through each array and find the state for which the probability is the highest for the corresponding agent, and call the showEstimatedPosition function to show where you believe each enemy agent to be. The “X” drawn by this function is a different color for each of the five enemy agents, to make it easier to tell which X’s correspond to which enemies.

## 5 Pass-off

For the pass-off, you will need to run your code with three different grids of probabilities:

- One where grid points are placed every sixteen integers across the field
- One where grid points are placed every eight integers across the field, and
- One where grid points are placed every four integers across the field

The enemies will always be placed at locations on the field whose coordinates are both multiples of eight. Therefore, in the case of grid points being placed every sixteen integers, there is the possibility that the agent will not actually be found at any of your grid points. As we watch your estimator compute the estimated positions of each enemy, the estimates should get better and better with time. The code will automatically
clear the X’s off the field periodically so we can watch how the newer X’s are better estimates than the old X’s.

Another important part of the pass-off will be to show to the TA how you implemented Bayes’ rule. There are a lot of different ways that you could successfully estimate the true agent positions, but the purpose of this lab is to teach you about Bayes’ rule.

Good luck! ☺️}