CS 70 Discrete Mathematics for CS
Spring 2004 Papadimitriou/Vazirani

Lecture 25

Minesweeper

Our final application of probability is to Minesweeper. We begin by discussing how to play the game optimally; this is probably infeasible, but a good approximation is to probe the safest squares. This motivates the computation of the probability that each square contains a mine, which is a nice application of what we have learned about both logic and probability.

Optimal play in Minesweeper

We have seen many cases in Minesweeper where a purely logical analysis is insufficient, because there are situations in which no move is guaranteed safe. Therefore, no Minesweeper program can win 100% of the time. We have measured performance by the proportion of wins as a function of the initial density of mines. Can we find the algorithm that plays better than all others, i.e., has a higher probability of winning?

The first step in the argument is to move from *algorithms*, of which there are infinitely many, to *strategies*, of which there are only finitely many. Basically, a strategy says what to do at every point in the game. Notice that a strategy is specific to a particular number of mines M and total number of squares N (as well as the shape of the board):

Definition 25.1 (Strategy): A strategy for a Minesweeper game is a tree; each node is labelled with a square to be probed and each branch is labelled with the number of mines discovered to be adjacent to that node. Every node has 9 children with branches labelled $0,\ldots,8$; no node label repeats the label of an ancestor; and the tree is complete with $N-M$ levels.

Figure 1 shows an example. Even for fixed N and M, there are a lot of possible strategies:

$$\prod_{i=0}^{N-M-1} (N-i)^{9^i}$$

(For $N=6$, $M=3$, this is 685078892498860742907977265335757665463718379520000000000.) Still, the number is finite. It is easy to see that every (terminating) algorithm for Minesweeper (given fixed N, M, and

Figure 1: First two levels of a Minesweeper strategy.

board shape) corresponds to exactly one strategy. Furthermore, with every strategy there is an associated probability of winning. Indeed, it is possible to calculate this exactly, but it is probably easier to measure it using repeated trials.

OPTIMAL

Definition 25.2 (Optimality): A strategy for Minesweeper is **optimal** if its probability of winning is at least as high as that of all other strategies.

Clearly, there is an optimal strategy for any given N, M, and board shape. Moreover, we can construct an optimal method of play *in general* (for arbitrary N, M, and board shape): enumerate every strategy for that configuration and play the optimal one. The performance profile of this algorithm will dominate that of any other algorithm.

(Note that the existence of an optimal algorithm rests on the finite number of possible strategies. There are infinitely many algorithms, and it is easy to imagine problems for which any given algorithm can always be improved (e.g., by spending a little bit more computation time) so that no optimal algorithm exists.)

Unfortunately, it is probably the case that no *practical* optimal algorithm exists. Instead, we will try a simpler approach: pick a safe square when one is known, otherwise pick the safest square as judged by the probability that it contains a mine.

The probability space

The first step is to identify the set of random variables we need:

- As in the propositional logic case, we want one Boolean variable X_{ij} which is true iff square (i, j) actually contains a mine. Using N to refer to the total number of squares, we can also label these variables as $X_1, \ldots, X_N$, which will come in handy.
- We'll also have variables D_{ij} corresponding to the display contents *for those k squares that have been probed or marked as mines*. We can also label these variables $D_1, \ldots, D_k$ for simplicity. The domain for each of these variables is $\langle m, 0, 1, 2, \ldots, 8 \rangle$. We'll call these variables the *Numbers*, and the corresponding X_{ij} variables will be called *Known*. (Note that *Numbers* includes marked mines; we assume that only logically guaranteed mines are marked.)

Finally, we'll use and M to refer to the total number of mines. (Both N and M are constants.)

As an example, consider the following display:

MINES LEFT: 3

3				
2	m			
1	2		2	
	1	2	3	4

Here N is 12, M is 4, (1,1), (3,1), and (1,2) are *Known*.

Now we must write down our probability space, i.e., the joint distribution over all the variables. It turns out to be easiest to do this in two parts, using the chain rule:

$$P(X_1, \ldots, X_N, D_1, \ldots, D_k) = P(D_1, \ldots, D_k | X_1, \ldots, X_N) P(X_1, \ldots, X_N)$$

Why this way? Because (1) the prior distribution of mines, $P(X_1, \ldots, X_N)$, is easy to compute because mines are scattered uniformly at random, and (2) the display variables are determined by the underlying mines, so the conditional distribution $P(D_1, \ldots, D_k | X_1, \ldots, X_N)$ is also relatively easy to describe.

First, the prior $P(X_1,\dots,X_N)$. Remember that the first square selected is always safe. Without loss of generality, let us call this square X_1; we know that $X_1 = false$. For the remaining $N-1$ squares, M mines are scattered at random. There are $\left(_{N-1M}\right)$ ways to do this and each is equally likely, so we have

$$P(x_2,\dots,x_N) = \begin{cases} \left(_{N-1M} \quad \text{if} \quad {}^{1/}_{\#(x_2,\dots,x_N)=M}\right) \\ 0 \quad \text{otherwise} \end{cases}$$

where $\#(x_2,\dots,x_N)$ denotes the number of "trues" in the sample point $x_2,\dots,x_N$. To check, we can calculate $P(X_i)$, the prior initial probability of a square X_i ($i \neq 1$) containing a mine: this is given by

$$\left(\left(_{N-2M-1}\right)/ \right)_{N-1M=M/(N-1)}$$

which is in accord with our expectations.

(From the assumption of uniform scattering and the fact that $P(X_i)$ for all $i \neq 1$, one is tempted to write the joint distribution as

$$P(X_2,\dots,X_N) = P(X_2)P(X_3)\dots P(X_n) \quad \text{WRONG}$$

But independence does not hold, because the total number of mines is fixed! For example, if the first M squares all get mines, then the next square has probability 0 of getting a mine.)

Turning to the display variables, we know they are determined precisely according to the rules of Minesweeper, given the actual contents of all the squares. I.e., for all combinations of values $d_1,\dots,d_k,x_1,\dots,x_N$,

$$P(d_1,\dots,d_k|x_1,\dots,x_N) = \begin{cases} 1 \text{ if } d_1,\dots,d_k \text{ correctly displays } x_1,\dots,x_N \\ 0 \text{ otherwise} \end{cases}$$

Finding safer squares

Now we need to compute, for each unknown square (i,j), the quantity $P(X_{ij}|known, numbers)$. We develop a simple and computable expression for this probability in a series of steps.

First, we need to massage the expression into a form containing the terms that we know about—the prior over all the X-variables, and the conditional probability of the display variables given those variables. The expression $P(X_{ij}|known, numbers)$ is missing the unknown variables other than X_{ij}; call these $Unknown$. For example, in the 4×3 display above, X_{ij} might be $X_{2,1}$ and there are 8 squares in $Unknown$. We introduce them by summing over them (the standard summation over constituent sample points for an event):

$$P(X_{ij}|e) = P(X_{ij}|known, numbers) = \sum_{unknown} P(X_{ij}, unknown|known, numbers)$$

For example, with 8 unknown squares, this is a sum of $2^8 = 256$ terms. We'd like to have an expression with *numbers* conditioned on X-variables, so we apply Bayes' rule:

$$\begin{aligned} P(X_{ij}|e) &= \alpha \sum_{unknown} P(numbers|known, X_{ij}, unknown)P(X_{ij}, unknown|known) \\ &= \alpha \sum_{unknown} P(numbers|known, X_{ij}, unknown)P(X_{ij}, unknown, known)/P(known) \\ &= \alpha' \sum_{unknown} P(numbers|known, X_{ij}, unknown)P(X_{ij}, unknown, known) \end{aligned}$$

So far, so good; the variables $known, X_{ij}, unknown$ constitute all the X-variables, so we have terms here that we already defined in our probability space given earlier. The only problem is that we have too many of them! The number of unknown variables could be as large as N, so the summation is over $O(2^N)$ cases.

The solution to this problem is to identify a subset of these variables that affect the display and to simplify the expressions using conditional independence so that the summation covers only this subset.

Let *Fringe* denote those unknown variables (not including X_{ij}) that are adjacent to numbered squares. For example, if X_{ij} refers to the square (2,1), then *Fringe* contains (2,2), (3,2), (4,2), (4,1). The idea is that the *Numbers* are completely determined just by *Known*, *Fringe*, and X_{ij} (if it is adjacent to a number). Given these, the *Numbers* are conditionally independent of the remaining unknown variables, which we call *Background*. In our example, *Background* consists of the top row: (1,3), (2,3), (3,3), (4,3).

There are actually two slightly different cases to deal with, depending on whether X_{ij} is adjacent to a number.

Case 1: X_{ij} is adjacent to a number.

$$\begin{aligned}
&P(X_{ij}|e)\\
&\quad = \alpha' \sum_{fringe, background} P(numbers|known, fringe, background, X_{ij}) P(known, fringe, background, X_{ij})\\
&\qquad \text{replacing } unknown \text{ by } fringe, background\\
&\quad = \alpha' \sum_{fringe, background} P(numbers|known, fringe, X_{ij}) P(known, fringe, background, X_{ij})\\
&\qquad \text{by conditional independence}\\
&\quad = \alpha' \sum_{fringe} P(numbers|known, fringe, X_{ij}) \sum_{background} P(known, fringe, background, X_{ij})\\
&\qquad \text{because the first term doesn't depend on } background
\end{aligned}$$

Now, in this last expression, the term $P(numbers|known, fringe, X_{ij})$ is 0 unless, to use our logical terminology, the assignment denoted by $fringe, X_{ij}$ is a *model* of the CNF expression implied by the evidence. If it *is* a model, then the probability is 1. So the summation over $fringe$ reduces to a simpler summation over the models of the evidence:

$$P(X_{ij}|e) = \alpha' \sum_{\{fringe:\, \langle fringe, X_{ij}\rangle \in models(e)\}} \sum_{background} P(known, fringe, background, X_{ij})$$

The sum over the background variables, which may still have a huge number of cases, can be simplified because the prior probability term $P(known, fringe, background, X_{ij})$ is $1/\binom{N-1}{M}$ or 0, depending on whether $\#(known, fringe, background, X_{ij}) = M$. Therefore, we just have to count the number of cases where the background has the right number of mines. This is given by $\binom{B}{M-L}$, where B is the size of the background and L is $\#(known, fringe, X_{ij})$, i.e., the number of mines not in the background. Finally, we obtain

$$\begin{aligned}
P(X_{ij}|e) &= \alpha' \sum_{\{fringe:\, \langle fringe, X_{ij}\rangle \in models(e)\}} \binom{B}{M-L} \Big/ \binom{N-1}{M}\\
&= \alpha'' \sum_{\{fringe:\, \langle fringe, X_{ij}\rangle \in models(e)\}} \binom{B}{M-L} \qquad (1)
\end{aligned}$$

which is simple to compute provided we can enumerate the models for the fringe variables. This costs $O(2^{|Fringe|})$, roughly the same as the logical algorithm. Notice that the term $1/\binom{N-1}{M}$ disappears into the normalizing constant because it does not depend on $fringe$.

Now we apply this formula to compute $P(X_{2,1}|e)$ in our example. First, let's enumerate the models. When $X_{2,1} = true$, these are the fringe models:

3	?	?	?	?
2	m		**m**	
1	2	**m**	2	
	1	2	3	4

3	?	?	?	?
2	m			**m**
1	2	**m**	2	
	1	2	3	4

3	?	?	?	?
2	m			
1	2	**m**	2	**m**
	1	2	3	4

When $X_{2,1} = false$, these are the fringe models:

3	?	?	?	?
2	m	**m**	**m**	
1	2		2	
	1	2	3	4

3	?	?	?	?
2	m	**m**		**m**
1	2		2	
	1	2	3	4

3	?	?	?	?
2	m	**m**		
1	2		2	**m**
	1	2	3	4

For each of these models, $\binom{B}{M-L} = \binom{4}{1} = 4$, so we have

$$P(X_{2,1}|e) = \alpha'' \langle 3\times 4, 3\times 4\rangle = \langle 1/2, 1/2\rangle$$

(This is in accord with the "intuitive" argument that says there is exactly one mine in (2,1) or (2,2), and it is equally likely to be in either.) We can show similarly that $P(X_{3,2}|e) = \langle 1/3, 2/3\rangle$ (i.e., 1/3 probability of a mine).

Case 2: X_{ij} is not adjacent to a number.
The derivation is very similar, but X_{ij} acts like a background variable rather than a fringe variable:

$$\begin{aligned} P(X_{ij}|e) &= \alpha \sum_{fringe,background} P(numbers|known, fringe, background, X_{ij}) P(known, fringe, background, X_{ij}) \\ &= \alpha \sum_{fringe,background} P(numbers|known, fringe) P(known, fringe, background, X_{ij}) \\ &= \alpha \sum_{fringe} P(numbers|known, fringe) \sum_{background} P(known, fringe, background, X_{ij}) \end{aligned}$$

Now the fringe variables constitute the entire model:

$$P(X_{ij}|e) = \alpha \sum_{fringe\in models(e)} \sum_{background} P(known, fringe, background, X_{ij})$$

and the final expression is very similar.

$$P(X_{ij}|e) = \alpha' \sum_{fringe\in models(e)} \binom{B}{M-L} \qquad (2)$$

It is clear that this expression is the same for all X_{ij} that are not adjacent to a number, so we need only do this once to get the probability of safety for background squares. (Each fringe square must be evaluated separately.)[1]

Let us calculate, say, $P(X_{1,4}|e)$ in our example. We have $B = 3$ and $M = 4$.
When $X_{1,4} = true$, $L = 4$, so $\binom{B}{M-L} = \binom{3}{0} = 1$.

[1]Note: the meaning of *Fringe* and *Background* differs in Case 1 and Case 2. In the first, *Fringe* includes all number-adjacent variables except X_{ij}, while *Background* includes all other variables. In the second, *Fringe* includes all number-adjacent variables, while *Background* includes all other variables except X_{ij}. This makes for somewhat simpler mathematical expressions. The code in `minep.scm` uses the opposite convention: the fringe is all number-adjacent variables, and the background is all other variables.

```
Total mines: 6  To find: 6

  - - - -    0.298  0.298  0.298  0.737
  - 2 - -    0.298  --2--  0.298  0.737
  - - - -    0.298  0.105  0.105  0.895
  - - 2 1    0.737  0.895  --2--  --1--
```

Figure 2: Example of a situation in which probabilistic inference distinguishes between very safe squares—(2,2) and (3,2)—and very unsafe squares—(2,1) and (4.2). Essentially, sharing a mine between the two “2” numbers would mean forcing 3 mines into the three background squares (1,1), (4,3), and (4,4), which can happen in only one way and hence is unlikely.

When $X_{1,4} = false$, $L = 3$, so $\binom{B}{M-L} = \binom{3}{1} = 3$.
There are 6 fringe models, so

$$P(X_{1,4}|e) = \alpha' \langle 6 \times 1,\, 6 \times 3 \rangle = \langle 1/4, 3/4 \rangle$$

That is, each background square has a 1/4 probability of being a mine. So the safest move is in a background square.

We can check the theorem we proved in the linearity-of-expectation homework: that the sum of probabilities equals the number of mines left. The sum of probabilities is

$$\frac{1}{2}+\frac{1}{2}+\frac{1}{3}+\frac{1}{3}+\frac{1}{3}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4}+\frac{1}{4} = 3$$

We can also check the work by calculating the probability of a mine after the first move (before finding out the number in the square). In that case, all the remaining squares besides X_{ij} and the initial square are background squares ($B = N-2$), and $L = 1$ when $X_{ij} = true$ and 0 when $X_{ij} = false$. There are no variables in the fringe, so there is exactly one (empty) model. Hence

$$\begin{aligned}
P(X_{ij}|e) &= \alpha \left\langle \binom{N-2}{M-1}, \binom{N-2}{M} \right\rangle \\
&= \frac{\binom{N-2}{M-1}}{\binom{N-2}{M-1} + \binom{N-2}{M}} \\
&= \frac{\binom{N-2}{M-1}}{\binom{N-1}{M}} \quad \text{by Pascal's identity, } \binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k} \\
&= \frac{(N-2)!}{(M-1)!(N-M-1)!} \, \frac{M!(N-M-1)!}{(N-1)!} = \frac{M}{N-1}
\end{aligned}$$

which agrees with our earlier calculation.

Figure 2 shows a case where probability really helps in deciding what to do. Figure 3 shows the performance of all four methods for minesweeper. Each improvement in the algorithm gives an improvement in play across all mine densities, but improvement is most pronounced for the more difficult cases.

Figure 3: Performance of the brain-dead, Mark II, Mark III, and probailistic algorithms on a 4×4 board, averaged over 100 trials.