Bushfire Modelling with Uncertainty

Malcolm Ryan

3 Jun 2004

1 Modelling bushfire as a cellular automaton

I decided to create a very simple bushfire model to begin exploring issues about modelling our uncertainty in how it propagates. The model I am using is a cellular automaton - a grid of cells, each with a particular state, and a collection of rules which define how the state of a cell changes over time, based on the states of its neighbours. In this case, the state of the cell is whether it is unburnt (green), burning (red) or burnt (black). The rules for updating a cell are simple:

- if the cell is **green**, and it has n neighbours (among the surrounding 8 cells) that are **red**, then it becomes **red** with probability $1-(1-p_{catch})^n$, otherwise it stays **green**;
- if the cell is **red**, it becomes **black** with probability p_{die};
- if the cell is **black**, it remains **black**.

Typical values of p_{catch} and p_{die} are 0.15 and 0.6 respectively. In a more realistic simulation p_{catch} and p_{die} would depend on local environmental condtions: fuel load, wind, temperature.

In the simulation I have built, the world is represented as a 100x75 grid of cells. Given that each cell can take one of three different values, this yields $3^7500 \approx 10^3578$ different maps.

2 Modelling Uncertainty

We can simulate the above cellular automaton easily, but it would only give us one possible outcome from a given set of starting conditions. This is not so much what we are interested in. Rather, we want to know something about the *probability distribution* over possible outcomes. We will want to ask questions like: "What is the probability of fire in this particular cell (or group of cells)." and "Where is the fire most likely to be?".

Furthermore, we will want to be able to update this estimate based on later observations. If the fire is sighted in a particular cell, then we would like to update our map appropriately - and not just by making the corresponding cell red. Knowing about a fire in one cell tells us something about the likely state of the neighbouring cells also. We would like to include as much of this information as possible.

Plus our observations might not always be in the form of an finding the state of a particular cell. In a more elaborate model other things could be measured: wind, temperature etc. This information tells us something about the state of the system as a whole.

3 The probability math

To formalise things, let us consider how we might compute the joint probability distribution over all possible maps. Let

$$\mathbf{x} = (x^{1,1}, x^{1,2}, \cdots, x^{m,n})$$

be one state of the map, where the map is m by n. Each element $x^{i,j}$ of the vector represents the state of the cell at position (i, j), which is either unburnt (G), burning (R) or burnt (B).

The process tracking the fire now divides into two steps: 1) predicting how the state evolves due to transitions, and 2) Estimating what the real state is based on observations.

3.1 Prediction step

The transition equation is:

$$\begin{aligned} P(\mathbf{x}_t) &= \sum_{\mathbf{x}_{t-1}} P(\mathbf{x}_t|\mathbf{x}_{t-1})P(\mathbf{x}_{t-1}) \\ &= \sum_{x_{t-1}^{1,1}} \cdots \sum_{x_{t-1}^{m,n}} P(x_t^{1,1}, \cdots, x_t^{m,n}|x_{t-1}^{1,1}, \cdots, x_{t-1}^{m,n})P(x_{t-1}^{1,1}, \cdots, x_{t-1}^{m,n}) \end{aligned}$$

Now given that each cells state only depends on its previous states and those of its 8-neighbours, we define:

$$N_8(x^{i,j}) \equiv \left\{x^{i-1,j-1}, x^{i,j-1}, x^{i+1,j-1}, x^{i-1,j}, x^{i,j}, x^{i+1,j}, x^{i-1,j+1}, x^{i,j+1}, x^{i+1,j+1}\right\}$$

Now

$$\begin{aligned} P(x_t^{1,1}, \cdots, x_t^{m,n}|x_{t-1}^{1,1}, \cdots, x_{t-1}^{m,n}) &= \prod_{i,j} P(x_t^{i,j}|x_{t-1}^{1,1}, \cdots, x_{t-1}^{m,n}) \\ &= \prod_{i,j} P(x_t^{i,j}|N_8(x_{t-1}^{i,j})) \end{aligned}$$

because the subsequent state of each cell depends only on its neighbourhood (i.e. it's spatially markov), and given this information it is conditionally independent of all other cells. So:

$$P(\mathbf{x}_t) = \sum_{x_{t-1}^{1,1}} \cdots \sum_{x_{t-1}^{m,n}} P(x_{t-1}^{1,1}, \cdots, x_{t-1}^{m,n}) \prod_{i,j} P(x_t^{i,j}|N_8(x_{t-1}^{i,j}))$$

unfortunately, I can't see any particularly useful way to simplify this further. The computation is still very large. The string of sums requires 3^{mn} operations, each of which is the product of m*n terms. Storing the old joint distribution will also require 3^{mn} space. This is far too much time and space.

3.2 Estimation step

Let us suppose we make an observation z, drawn from a sensor model $P(z|\mathbf{x})$. We can compute the posterior probability over $\mathbf{x}$ by applyingBayes rule:

$$P(\mathbf{x}|Z) = \frac{P(z|\mathbf{x})P(\mathbf{x})}{P(z)}$$

In our current simulation, the obervations are localised to particular cells on the map, and depend only on the state of those cells. In general, it will probably be likely that observations will depend on some neighbourhood of cells about a particular location. Ie:

$$P(z|\mathbf{x}) = P(z|N(z))$$

for some neighbourhood $N(z)$ of z.

4 Estimating the joint probability

Now as we pointed out above, the number of possible maps is exponential in the size of the grid. So the joint probability distribution over these maps has exponentially many terms. We cannot compute this or store this in any reasonable manner. So we will need to approximate it. But first we should ask ourselves the question: why do we need the full joint anyway? Will not the individual cell-by-cell marginal distributions do the job? The answer, unfortunately, is no.

The state of cell at time t depends on the previous state of its neighbours at $t-1$, which is turn depends on *their* neighbours at time $t-2$ and so on. Over time the state of every cell becomes correlated with the state of every other. The strength of this correllation depends on the parameters p_{catch} and p_{die}, and on the distance between the cells. If p_{catch} is low, then two distant cells will only be weakly coupled, and it may be reasonable to treat them as independent.

5 Particle Filtering

We can approximate the joint distribution using a particle filter. Each particle represents one potential state of the map - i.e. one sample vector $\mathbf{x}$. Prediction is done by evolving each particle according to the transition function. Estimation is done by reweighting each particle in proportion to it's likelihood, given the observation.

The problem here is that we cannot have nearly enough particles to accurately represent the joint probability distribution. Therefore correlations between distant states are artificially high. In particular, when p_{catch} is low, the probability of any particular cell being on fire at a particular time is small. So if fire is observed at a particular point, only a very small number of particles will match. These particles will be weighted highly, which will disproportionately affect the estimates for all cells in the map. Resampling exacerbates this problem even further.

A related issue is the absense of any fire at all in the initial state. If all the particles are free of fire, then a fire observed at a particular location will not find any strongly matching particles, so the observation will effectively be ignored. For this to work in the standard particle filter model, there must be a particle to represent every possible new fire. With 7500 cells, this means 7500 particles, just to handle the initial conditions. If there are other unknown factors, then the number multiplies. In essence, the problem is that low-probability events will be ignored, even if observed, unless there is a particle that represents them.

In this case, spontaneous fire at any particular location is a very low probability event. But given that there are many cells, spontaneous fire at *some* location has significant probability (this, after all, is the reason why we're interested in this problem).

5.1 Tiled particle filtering

The first of these problems can be addressed in a naive fashion by deliberately dividing the grid up into regions which are deemed to have independent probability distributions, meaning that the joint probability distribution can be decomposed into the product of probability distributions over individual regions.

I have implemented what I call a "tiled particle filter". The grid is divided up into a tessellation of tiles, each a rectangle of size q by r cells. For each tile we then essentially run a particle filter separate. Because the number of cells in a tile is assumed to be small, the probability distribution can be reasonably represented with a much smaller number of particles.

The tiles are not totally independent, however. Each tile-particle ("subparticle") is associated with a collection of other sub-particles from different tiles, which together cover the entire map. When the prediction step is performed, the transition function is applied to the map formed by this set of particles. So cells in one sub-particle with be affected by cells in neighbouring sub-particles.

When resampling is done, some sub-particles are deleted, leaving holes in these maps. However for each tile an equal number of sub-particles are duplicated. So the duplicated sub-particles are stitched into the holes left by those discarded. Currently this is done in a completely random way.

Alone, this kind of tiled estimator produces unusual effects. An observation only affects the tile in which it is made. This leads to strange discontinuities near the boundaries of tiles. This has been overcome fairly effectively by using several such tiled estimators with tilings offset from one another. Probabilities are then computed by a weighted average of all the estimators.

The outcome is that cells very close to an observation will share many tiles with that observation and will be strongly affected. As you move further away, there will be fewer shared tiles and the effect will diminish, until at a certain distance there will be no effect at all.

This is a somewhat hand-wavy description of the process. The algorithm has been coded up and appears to work well, under manual inspection, but I have not yet proven anything about its effectiveness or thoroughly evaluated it.

6 Related ideas in SLAM

It seems to me that there are some ideas in SLAM that are strongly related. The issues dealt with in coupling landmarks in submaps (Bailey, 2002) seem to be very similar. The general solution there is interesting: each sub-group of landmarks is correlated with a particular origin. Then at the higher level the origins of different submaps are correlated with one another. This leads to an interesting generalisation: a group of state-variables could be correlated with a particular abstract description of that group. These abstract features could then be correlated between groups, in order to transfer information from one group to another.

In the bushfire case we are interested in what the presense of fire in one tile can tell you about the presence of fire in other tiles. Potentially we could ignore all position information and just represent each tile with 3 values equal to the proportion of green, red and black cells in that tile. There may be some significant correlation in these values between different tiles in the map.

References

Bailey, T. (2002). *Mobile robot localisation and mapping in extensive outdoor environments.* Unpublished doctoral dissertation, Australian Centre for Field Robotics, University of Sydney.