MML Notes

Patrick Robotham

May 1, 2013

This is a collection of notes about using MML to infer a single model in the general reinforcement learning problem set up. We can reduce this problem to that of predicting the next bit in an infinite stream of bits, taking in to account the bits observed previously.

1 Prediction Suffix Trees

We look at the class of D order binary Markov Models, where D is a fixed integer. These can be represented by prediction suffix trees.

Definition 1.1 A *prediction suffix tree* (M, θ) is a proper binary tree where each leaf node l is equipped with a probability distribution θ_l over $\{0, 1\}$.

Notation: Let M be a binary tree. Let s be a bitstring. M_s is the node in M reached by the following algorithm:

```
set M_s = M.root

for c in s:
    If M_s is a leaf node:
        return M_s
    If c == 0
        M_s = M_s.right
    If c == 1
        M_s = M_s.left
endfor
return M_s
```

To encode a prediction suffix tree, we must specify the tree structure M and encode the probabilities of the leaves θ.

1.1 Calculating Probabilities With Prediction Suffix Trees

Let (M, θ) be a prediction suffix tree. Let d be the depth of M. Let s be a bitstring. Assume $|s| > d$. Let n satisfy $d < n < s$.
Then $Pr_M(s_{n+1}|s_{1:n})$ is calculated as follows:

1. Let $s' = \text{reverse}(s_{1:n})$.
2. Return $\theta_{M_{s'}}$.

1.2 Coding Tree Structure

We assume we have an upper bound D for the proper binary tree M. We then code the tree by calling the function below with M.root. (The idea of this code is that we preform a pre-order traversal, writing down 1 for internal nodes, 0 for leaf nodes of length less than D, and nothing otherwise.)

```
function code(node,depth){
 if(node.leaf() == true){
   if(depth < D){
      message += 0;
   }
 } else {
   message += 1;
   code(node.left, depth+1);
   code(node.right, depth+1);
 }
}
message = [];
code(M.root, 0);
```

We denote the length of this code by $\Gamma_D(M)$.

1.3 Coding Leaf Probabilities

The leaf probabilities should only depend on the data observed so far.
Intuitively, each leaf of the tree is a bin to hold data. Every bit of data gets filed away in a leaf. To save space, we need only keep track of the number of 1s and 0s filed away in each leaf. We do not keep track of the order.
Suppose we have a 1s and b 0s filed away in a leaf l. Then the MML-Estimate (and the KT-Estimate) probability of the next bit being a 1 is

$$\theta(l) = P(1|a\ 1s, b\ 0s) = \frac{a + 1/2}{a + b + 1}$$

Once we have the leaf probabilities, we can regard the tree as a basis for a code which can convey the entire history.

The message length I_1 for a leaf l that has a 1s and b 0s is

$$\begin{aligned} I_1(l) =& \frac{1}{2}\log(a+b) - (a+\frac{1}{2})\log\theta_l - (b+\frac{1}{2})\log(1-\theta_l) \\ & - \log\binom{a+b}{a} + \frac{1}{2}\log\frac{1}{12} + \frac{1}{2} \end{aligned}$$

(See p. 246 of Wallace 2005.)