DMC(Dynamic Markov Compression)

(Read Sayood, pp.174-176, Bell et. Al., Managing Gigabytes,pp.69-72)

- **Uses a finite state model**
- **Works at the bit level**
- **Adaptive**
- **Slow in performance**
- **Compression ratio comparable to PPM**

Initial model : A one state fsa with transitions 0/1 and 1/1. In general, the transitions are marked by 0/*p* or 1/*q* where *p* and *q* denote non-zero counts of number of transitions from that state (not including the next bit) from a given state with input 0 or 1, respectively. The initial count is set to 1 to take care of the "zero frequency" problem.

The estimate of probability of a 0 being the next input is $p/(p+q)$ and 1 being the next state is $q/(p+q)$. These probability values are used by a background entropy coder(such as an arithmetic coder) to estimate the interval in the cumulative probability space for the next symbol in the input. If the next symbol is zero, it will change then the count to $p+1$, make the appropriate state change and continue. Similarly, for a 1 input which will change the value of q to $q+1$.

Adaptation(Cloning): If a transition to a particular state *t* from some previous state *u* is heavy, that is more frequent, in order to capture this skewed context, the state *t* is cloned. A threshold value of the count value for cloning must be agreed upon by both encoder and the decoder. The cloned state is designated as *t'*. This state *t'* records independent probabilities for symbols occurring after a transition from state *u* to state *t.* Previously, they were treated with no distinctions from states other than *u*. The rules for cloning is as follows:

1. All transitions from state u are directed to state *t'*. All other transitions from other states to state *t* remain unchanged (unless those transitions also satisfy the threshold and separate cloned states have to be created for the state t for these transitions).
2. The counts on the transitions out of *t'* are in the same ratio (between 1 and 0) as the counts out of *t.*
3. The counts of *t* and *t'* are adjusted so that the sum is equal to the sum of counts in transitions coming in. This is logical equivalent of Kirchoff's law.

The DMC Algorithm

Two data structure *T* and *C* are defined. Each is a table with rows equal to number of states *s* and two columns corresponding to input 0 or 1. The first table is just the 'next' state transition table and is denoted as *T*[1:*s*][0,1] and *C* is the count table giving the number of transitions for a given state for input 0 or 1 and is designated as *C*[1:*s*][0,1].

To encode:

```
1. $s$:=1 /* The current number of states */
2. $t$:=1 /* the current state */
3. $T$[1][0]=$T$[1][1]:=1 /* initial model*/
4. $C$[1][0]=$C$[1][1[=1 /* initial count to avoid zero frequency problem*/
5. For each input bit $e$ do
    a) $u$:=t
    b) $t$:=$T$[$u$][$e$]  /* Follow the transition*/

    c) Code $e$ with probability $C$[$u$][$e$] / ($C$[$u$][0] + $C$[$u$][1])

    d) $C$[$u$][$e$] = $C$[$u$][$e$]+1

    e) If cloning threshold are exceeded then,

            $s$:=$s$+1 /* new state $t$'/

            $T$[$u$][$e$]= $s$

            $T$[$u$][$e$] := $s$

            $T$[$s$][$0$] := $T$[$t$][$0$]

            $T$[$s$][1] := $T$[$t$][$1$], and

            Move some of the counts from $C$[$t$] to $C$[$s$]
```

Discussion on DMC

The paper by Cormack and Horspool presented the original DMC algorithm. The paper first describes an algorithm by Guazzo. This algorithm takes a Markov Model and interprets the output of the source to be a binary fraction. So, if the output encoding begins with 0 1 1 0 1 …, this is considered to be a fraction 0.01101… The job of the encoder is to choose a fractional number between 0 and 1 which encodes the entire message. With reference to Fig.1 of the paper, initially the algorithm has the freedom to choose a sequence corresponding to 0.0000… and 0.1111.. and it therefore divides the space into a closed interval [0.000… , 0.0111..] to represent all messages that begin with '0' and all fractions in the interval [0.1000…, 0.111..] to represent messages that begin with '1'. The example source message begins with 0 so it must choose the first interval. At this point we are in state B in the Markov model. In this state, getting a 0 is twice as likely than getting a 1. So , the subinterval [0.000…, 0.0101..] represents source messages that begin with 00 and the interval [0.0101…, 0.0111..] represent source interval thar begin with 01. The first interval is is twice as likely as the second. We have to pick up the second subinterval since our source message begin with 01…. Continuing in this fashion our message '01101 must be contained in a the subinterval [011100110101.., 0.01110100101111..] etc.

The paper then goes into the details of how this algorithm can be adapted to a dynamic Markov model. It assigns the probabilities as

Prob{0/A}= (p+c)/(p+q+2c)   and Prob{1/A}= (q+c)/(p+q+2c) , where c= constant.

Note how it avoids the "zero frequency" problem.

Then , the paper describes how to build the state transition table and how the set of states change in the model. In this context, it explains the "cloning" operation (See Fig.2) and it has already been explained in the class.

Note, the rationale for cloning is explained in term of revealing the correlation of a transition in terms of preceding set of states rather than only the previous state and it defines two counts MIN_CNT1 and MIN_CNT2. (see p.545).

Section 4.3 describes how to start and stop the cloning operation. The initial state is considered to be a single state (or a binary tree-structured states of size 16; see Fig.3). But, it does not say how to split the initial single state. I think it should be like the following: To begin it has only one state with transitions with respect to 0 and 1 coming to the same state. When the conditions are satisfied, state 1 clones into state 1 and 1'. The transitions with 0's come to state 1. If a 1 appears, the state transition takes place to state 1' (cloned state). All transitions to the cloned state with respect to 1 end up in state 1'. If it receives a 0, the transition takes place back to state 1.