SCOT Approximation, Training and Asymptotic Inference

Mikhail Malyutov M.MALIOUTOV@NEU.EDU
Mathematics Department
Northeastern University
360 Huntington Avenue, Boston, MA 02115, USA
Paul Grosu PGROSU@GMAIL.COM
College of Computer and Information Science
Northeastern University
360 Huntington Avenue, Boston, MA 02115, USA

Editor: Alex Gammerman, Vladimir Vovk, Zhiyuan Luo, and Harris Papadopoulos

Abstract

Approximation of stationary strongly mixing processes by SCOT models and the Le Cam-Hajek-Ibragimov-Khasminsky locally minimax theory of statistical inference for them is outlined. SCOT is an m-Markov model with sparse memory structure. In our previous papers we proved SCOT equivalence to 1-MC with state space—alphabet consisting of the SCOT contexts. For the fixed alphabet size and growing sample size, the Local Asymptotic Normality is proved and applied for establishing asymptotically optimal inference. We outline what obstacles arise for a large SCOT alphabet size and not necessarily vast sample size. Training SCOT on a large string using clusters of computers and statistical applications are described.

Keywords: Strong mixing, strongly stationary sequences, Local Asymptotic Normality, Local Asymptotic Minimaxity, SCOT models, Edgeworth expansion.

Appendix. Simulation of an Asymmetric Cyclic Random Walk (RW)

The simulation program — written using the R statistical language — determines the number of visits of k for large numbers of sample size N, with an alphabet A. The algorithms have been augmented with pseudo-code in order to make the syntax easier to read. For the first algorithm we will determine the visits of k for a specific value of A and N, assuming the probability of moving left and remaining on the node in the ring is the same ($P(move\_left) = P(remain) = 0.5$):

Algorithm 1: GET_K_VISITS_FOR_N_JUMPS: Get visits of k after N Jumps given A

Inputs:
A: Number of nodes, where $k = A/2$. ($A = \{10, 20, \ldots, 100\}$)
N: Number of jumps. ($N = \{20, 40, 60, 80\}$)
$P(move\_left) = P(remain) = 0.5$

Outputs:
Returns the number visits at k ($A/2$) after N jumps.

```
begin
    k ← A/2
    visits ← 0
    A ← 1
    // Randomize the random number generator with
    // a seed based on the timer
    t = t ← as.numeric(Sys.time())
    seed ← 1e8 * (t − floor(t))
    set.seed(seed)
    while N > 0 do
        move_or_stay ← sample(c(−1, 0), prob =
        c(P_move_left, P_remain), size = 1, replace = TRUE)
        A_previous ← A
        N ← N − 1
        if move_or_stay == −1 then
            A ← A − 1
        if A < 1 then
            A ← k * 2
        if A_previous ≠ A AND A == k then
            visits ← visits + 1
    return (visits)
```

Next we had to build the table of $N \times A$ for one round, which is used in performing multi-round simulations for a table of $N \times A$ containing distributions of individual rounds. Both algorithms are presented in subsequent pages.

Algorithm 2: BUILD_N_BY_A: Build the $N \times A$ table for one round.

Inputs:
A_vals: List of number of nodes. ($A = \{10, 20, \ldots, 100\}$)
N_vals: List of number of jumps. ($N = \{20, 40, 60, 80\}$)
$P(move\_left) = P(remain) = 0.5$

Outputs:
Returns the table of the number visits at k ($A/2$) after N jumps for different values of A and N.

```
1 begin
2     N_by_A_Table ← matrix(|N_vals|, |A_vals)|)
3     for A ∈ A_vals do
4         for N ∈ N_vals ∗ A do
5             N_by_A_Table[N/A, A] ←
              c(GET_K_VISITS_FOR_N_JUMPS(A, N, P_move_left,
6                                                  P_remain))
7     return (N_by_A_Table)
```

Finally we run multi-round simulations for table of $N \times A$ using the following algorithm, which gives us a distribution for each element of the $N \times A$ table:

Algorithm 3: MULTI_ROUND_N_BY_A: Populate the $N \times A$ table for multi-round simulations .

Inputs:
A_vals: List of number of nodes. ($A = \{10, 20, \ldots, 100\}$)
N_vals: List of number of jumps. ($N = \{20, 40, 60, 80\}$)
$Rounds = 100$ is the rounds of repetition.
$P(move\_left) = P(remain) = 0.5$

Outputs:
Returns the table of the number visits at k ($A/2$) after N jumps for different values of A and N.

begin
  # Construct the initial table to which more values of the distributions will be appended $NA\_Table \leftarrow BUILD\_N\_BY\_A(A\_vals, N\_vals, P\_move\_left, P\_remain)$
  for $r \in \{1, \ldots, Rounds - 1\}$ **do**
    # Get the values of one round to append to each cell's distribution $N\_by\_A\_Table\_Sample \leftarrow BUILD\_N\_BY\_A(A\_vals, N\_vals, P\_move\_left, P\_remain)$
    for $A \in A\_vals$ **do**
      for $N \in N\_vals$ **do**
        $value\_list \leftarrow N\_by\_A\_Table\_Sample[N, A]$
        $cell\_list \leftarrow N\_by\_A\_Table[N, A]$
        $cell\_list \leftarrow c(cell\_list, value\_list)$
        $N\_by\_A\_Table[N, A] \leftarrow cell\_list$
  return $(N\_by\_A\_Table)$