## Temporal Memory with 3 Stages

Peter Overmann, 21 Jul 2022

## Algorithm

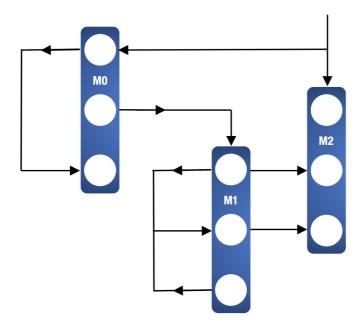
A temporal memory composed of three triadic memory units, arranged in three stages:

M0 is a bigram encoder, creating a unique code for each unique pair of consecutive inputs. M1 creates a random context vector for a consecutive pair of bigrams, and feeds it back to the delayed input.

M2 learns the association of the current input and the output of M1.

The subcircuit consisting of M1 and M2 is equivalent to the elementary temporal memory algorithm described earlier.

The prediction step, not explicitly shown in the following circuit diagram, is a query on M2 performed at the moment its two bottom positions are filled with new values propagated from M1.



```
TemporalMemory[t_Symbol, {n_Integer, p_Integer}] :=
  Module[{M0, M1, M2, overlap, i, y, c, u, v, prediction},
   TriadicMemory[M0, {n, p}]; (* encodes bigrams *)
   TriadicMemory[M1, {n, p}]; (* encodes context *)
   TriadicMemory[M2, {n, p}]; (* stores predictions *)
   overlap[a_SparseArray, b_SparseArray] := Total[BitAnd[a, b]];
   (*initialize state variables with null vectors*)
   i = j = bi = y = c = u = v = prediction = M1[0];
   t[inp_] := Module[{x, j, bigram},
     j = i;
     bigram = M0[i = inp, j, _];
     If [ overlap[ M0[i, _, bigram], j] < p, M0[i, j, bigram = M0[]]];</pre>
      (* bundle previous input with previous context *)
     x = BitOr[y, c];
     y = bigram;
      (* store new prediction if necessary *)
     If[prediction != i, M2[u, v, i]];
      (*create new random context if necessary*)
     If [overlap[M1[_, y, c = M1[x, y, _]], x] < p, M1[x, y, c = M1[]]];
     prediction = M2[u = x, v = y, _]
    ]
  ];
```

# Configuration

```
Get[ $UserBaseDirectory <> "/TriadicMemory/triadicmemoryC.m"]
n = 500; p = 5;
TemporalMemory[T, {n, p}];
```

## Encoder / Decoder

```
Encoder[e_Symbol, {n_Integer, p_Integer}] := Module[{code},
     e[Null] = SparseArray[{0}, {n}];
     e[{}] = Null;
     e[x_SparseArray] := Module[{s},
        s = e[Flatten[x["NonzeroPositions"]] // Sort];
        If[Head[s] === SparseArray,
         ToString[Sort[Flatten[x["NonzeroPositions"]]]], s]];
     e[s_] := e[s] = Module[{r},
         r = SparseArray[ RandomSample[ Range[n], p] \rightarrow Table[1, {p}], {n}];
         e[Sort[Flatten[r["NonzeroPositions"]]]] = s; r];
    ];
  Encoder[e, {n, p}]
Test function
  temporalmemorytest[s_String, repetitions_Integer] := Module[{b, symb},
     a = Flatten[Join[Table[Characters[s], repetitions]]];
     b = e / @ T / @ e / @ a;
     Row[Style[#[[1]], If[#[[1]] === #[[2]], Black, Red]] & /@
        Transpose[{a, Most[Prepend[b, Null]]}]]
    ];
Timing
  timing[s_String, repetitions_Integer] := Module[{ch, b, symb},
     a = Flatten[Join[Table[Characters[s], repetitions]]];
```

#### **Tests**

11.9422

AbsoluteTiming[T /@ e /@ a;][[1]]

timing["!@#\$%^&", 1000]

The following tests are run in a single session. The temporal memory processes a stream of characters with repeating patterns, at each step making a prediction for the next character. Correct

predictions are shown in black, mispredictions in red.

This method performs better than the two-stage elementary temporal memory.

Note that all characters are test input, we're not using the temporal memory to auto-continue a sequence here.

```
temporalmemorytest["ABC", 8]
ABCABCABCABCABCABCABC
temporalmemorytest [ "kiwi", 8]
kiwikiwikiwikiwikiwikiwikiwi
temporalmemorytest [ "apple", 8]
appleappleappleappleappleapple
temporalmemorytest [ "pepper", 8]
pepperpepperpepperpepperpepperpepperpepper
temporalmemorytest [ "tomato", 8]
{\color{blue}\mathsf{tomatotom}} a {\color{blue}\mathsf{tot}} o {\color{blue}\mathsf{matotom}} a {\color{blue}\mathsf{matotom}} a {\color{blue}\mathsf{tot}} o {\color{blue}\mathsf{matotom}} a {\color{blue}\mathsf{matotom}} o {\color{blue}\mathsf{matotom}} a {\color{blue}\mathsf{ma
temporalmemorytest [ "banana", 8]
bananabananabananabananabananabananabanana
temporalmemorytest [ "wiriwirichili", 8]
wiriwirichiliwiriwirichiliwiriwirichiliwiriwiric
   hiliwiriwirichiliwiriwirichiliwiriwirichili
temporalmemorytest [ "alfalfa", 20]
temporalmemorytest ["A quick brown fox jumps over the lazy dog. ", 4]
A quick brown fox jumps over the lazy dog. A quick bro
   wn fox jumps over the lazy dog. A quick brown fox jumps ove
   r the lazy dog. A quick brown fox jumps over the lazy dog.
temporalmemorytest [ StringDrop[ ToString[N[Pi, 100]], {2}], 5]
3141592653589793238462643383279502884197169399375105820974944592307816406286
   20899862803482534211706831415926535897932384626433832795028841971693993
   75105820974944592307816406286208998628034825342117068314159265358979323
   84626433832795028841971693993751058209749445923078164062862089986280348
   25342117068314159265358979323846264338327950288419716939937510582097494
   4592307816406286208998628034825342117068314159265358979323846264338327
   9502884197169399375105820974944592307816406286208998628034825342117068
```