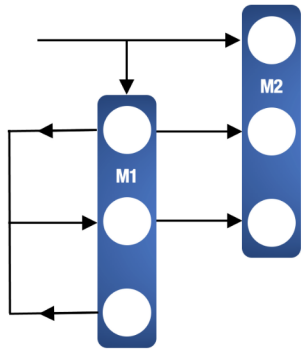
Temporal Memory - Elementary Algorithm

Peter Overmann, 21 Jul 2022

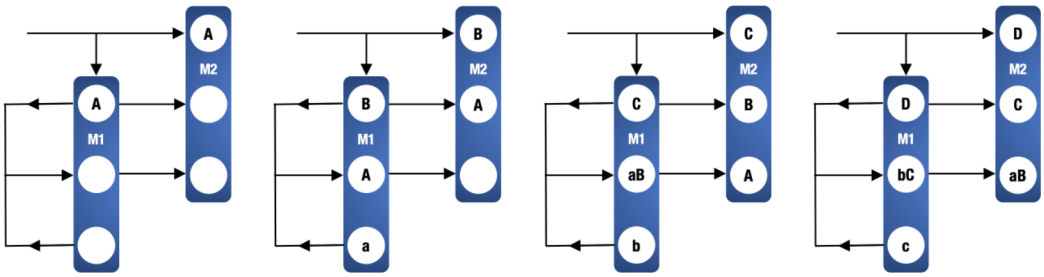
The simplest possible temporal memory algorithm composed of two triadic memory instances M1 and M2.

M1 creates a random context vector for a consecutive pair of inputs, and feeds it back to the delayed input.
M2 learns the association of the current input, the previous input, and the delayed previous input plus feedback.

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.



The following series of diagrams shows how a temporal sequence ABCD flows through this initially blank circuit.



```
TemporalMemory[t_Symbol, {n_Integer, p_Integer}] :=

  Module[{M1, M2, overlap, y, c, u, v, prediction},

   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 *)
   y = c = u = v = prediction = M1[0];

   t[inp_] := Module[{x},

     (*flush state if input is zero -needed when used as a sequence memory*)
     If[Total[inp] == 0, Return[y = c = u = v = prediction = M1[0]]];

     (* bundle previous input with previous context *)
     x = BitOr[y, c];

     (* store new prediction if necessary *)
     If[prediction ≠ (y = inp), M2[u, v, y]];

     (* 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

Timing

```
timing[ s_String, repetitions_Integer] := Module[{ch, b, symb},
   a = Flatten[Join[ Table[ Characters[s], repetitions]]];
   AbsoluteTiming[ T /@ e /@ a;][[1]]
  ];
```

```
timing[ "!@#$%^&", 1000]
```
```
6.72417
```

Test function

Tests

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.

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

```
temporalmemorytest [ "ABC", 8 ]
```
```
ABCABCABCABCABCABCABC
```

```
temporalmemorytest [ "kiwi", 8]
```
```
kiwikiwikiwikiwikiwikiwikiwikiwi
```

```
temporalmemorytest [ "apple", 8]
```
```
appleappleappleappleappleappleappleapple
```

```
temporalmemorytest [ "pepper", 8]
```
```
pepperpepperpepperpepperpepperpepperpepperpepper
```

```
temporalmemorytest [ "tomato", 8]
```
```
tomatotomatotomatotomatotomatotomatotomatotomato
```

```
temporalmemorytest [ "banana", 8]
```
```
bananabananabananabananabananabananabananabanana
```

```
temporalmemorytest [ "wiriwirichili", 8]
```
```
wiriwirichiliwiriwirichiliwiriwirichiliwiriwiric
 hiliwiriwirichiliwiriwirichiliwiriwirichiliwiriwirichili
```

```
temporalmemorytest [ "alfalfa", 20]
```
```
alfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaal
 falfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfa
```

```
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}], 10]
```

```
3141592653589793238462643383279502884197169399375105820974944592307816406286
 2089986280348253421170683141592653589793238462643383279502884197169399375
 1058209749445923078164062862089986280348253421170683141592653589793238
 4626433832795028841971693993751058209749445923078164062862089986280348
 5342117068314159265358979323846264338327950288419716939937510582097494
 5923078164062862089986280348253421170683141592653589793238462643383279
 5028841971693993751058209749445923078164062862089986280348253421170683
 1415926535897932384626433832795028841971693993751058209749445923078164
 0628620899862803482534211706831415926535897932384626433832795028841971
 6939937510582097494459230781640628620899862803482534211706831415926535
 8979323846264338327950288419716939937510582097494459230781640628620899
 8628034825342117068314159265358979323846264338327950288419716939937510
 5820974945923078164062862089986280348253421170683141592653589793238462
 6433832795028841971693993751058209749445923078164062862089986280348253
 4211706831415926535897932384626433832795028841971693993751058209749445
 9230781640628620899862803482534211706831415926535897932384626433832795
 0288419716939937510582097494459230781640628620899862803482534211706831
```