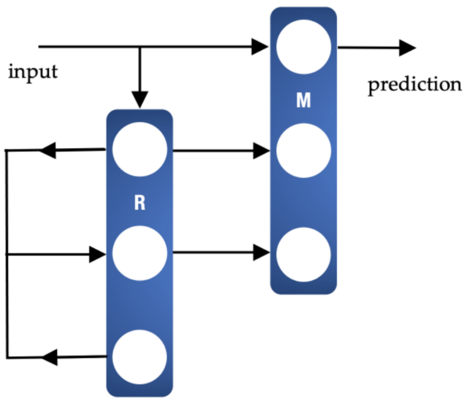
Deep Temporal Memory - Introduction

Peter Overmann 08 Aug 2022

Algorithm

A simple temporal memory algorithm can be composed of two triadic memory instances, wired together in the following circuit:



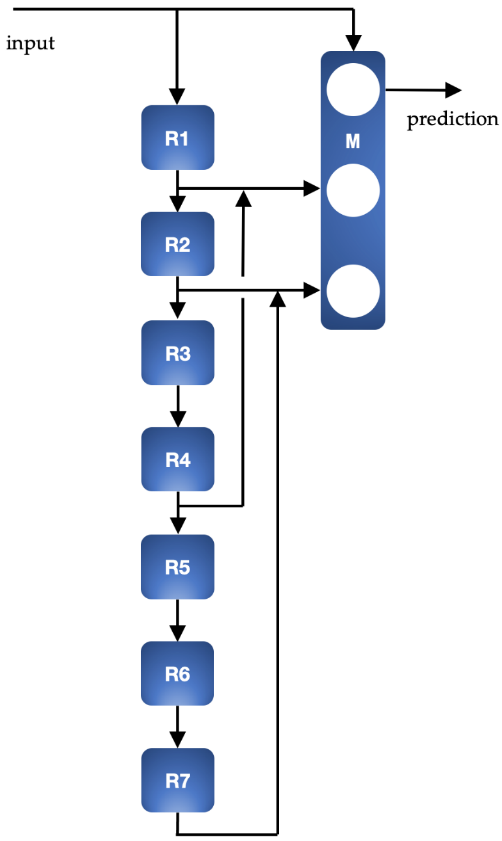
In this circuit, triadic memory R creates a random context vector for a consecutive pair of inputs, and feeds it back to the delayed input. This element effectively creates context-dependent bigrams from consecutive inputs.
Triadic memory M learns the association of the current input, and the readout from triadic memory R.

The elementary temporal memory circuit is capable of learning simple repeating patterns, such as 60 digits of pi.

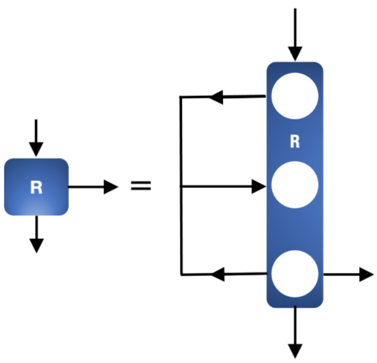
A circuit able to learn more complex patterns requires multiple feedback lines. There are infinitely many ways to design large circuits from triadic memory components. The following circuit diagram shows an example built from eight triadic memory units. The units R1 to R7 encode inputs to bigrams, bigrams to 3grams, and so on. At each step, feedback from the previous step is added to the inputs.

A readout memory M aggregates the state of several encoding states to learn the predicted value at each step.

This circuit design resembles an architecture known as reservoir computing, where a network involving multiple feedback loops propagates a series of inputs, and a readout component attaches predictions to the state of the reservoir.



In the above diagram, the circuit components R represent triadic memory units with feedback loops as follows:



This component consists of a single triadic memory and a triple of SDRs that persist the state of the component at each time step. It processes a stream of SDRs, returning an SDR which encodes the current input SDR and the previous SDR plus feedback from the previous step.

```
R[f_Symbol, {n_Integer, p_Integer} ] := Module[ {T, x, y, z, overlap},

   (* instantiate a triadic memory unit *)
   TriadicMemory[ T, {n, p}];

   overlap[a_SparseArray, b_SparseArray] := Total[BitAnd[a, b]];

   x = y = z = SparseArray[{0}, {n}];

   (* reset x, y, z *)
   f[SparseArray[{0}, {n}]] := x = y = z = SparseArray[{0}, {n}];

   f[input_SparseArray] := Module[ {},

     x = BitOr[y, z];  (* binarize x and y using ranked-max algorithm *)
     y = input;

     If[overlap[T[_, y, z = T[x, y, _]], x] < p, T[x, y, z = T[]]];
     z
    ];

  ];
```

The deep temporal memory circuit includes a chain of components R, generating bigrams from the inputs, trigrams from bigrams, etc. The readout memory M learns a prediction based on the temporal state of the encoding chain.

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

  Module[{M, R1, R2, R3, R4, R5, R6, R7,  x, y, z, t1, t2, t3, t4, t5, t6, t7},

   (* predictions / readout memory *)
   TriadicMemory[M, {n, p}];

   (* bigram encoder units *)
   R[#, {n, p}] & /@ {R1, R2, R3, R4, R5, R6, R7 };

   (* initialize state variables with null vectors *)
   x = y = z = t1 = t2 = t3 = t4 = t5 = t6 = t7 =  M[0];

   t[inp_] := Module[{},

     (* flush state if input is zero - needed when used
       as a sequence memory *)If[Total[inp] == 0, x = y = z = M[0]];

     (* store new prediction if necessary *)
     If[z ≠ inp, M[x, y, inp]];

     (* encoding chain *)
     t1 = R1[inp];
     t2 = R2[t1];
     t3 = R3[t2];
     t4 = R4[t3];
     t5 = R5[t4];
     t6 = R6[t5];
     t7 = R7[t6];

     (* prediction readout from t1, t2, t4 and t7 *)
     z = M[x = BitOr[t1, t4], y = BitOr[t2, t7], _]

    ]

  ];
```

Configuration

```
Get[ $UserBaseDirectory <> "/TriadicMemory/triadicmemoryC.m"]

n = 1000; p = 5;

TemporalMemory[ T, {n, p}];
```

Encoder / Decoder

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. All characters are test input -- the temporal memory is not used to auto-continue a sequence in this setup.

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

```
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]
alfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaalfalfaal
 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.
```

1000 digits of pi:

```
temporalmemorytest [ ToString[N[Pi, 1000]], 8]
3.1415926535897932384626433832795028841971693993751058209749445923078164062862
 089986280348253421170679821480865132823066470938446095505822317253594081
 1284811174502841027019385211055596446229489549303819644288109756659334461
 6128475648233786783165271201909145648566923460348610454326648213393607260
 60249141273724587006606315588174881520920962829254091715364367892590360
```

0113305305488204665213841469519415116094330572703657595919530921861173819326117931051185480744623799627495673518857527248912279381830119491298336733624406566430860213949463952247371907021798609437027705392171762931767523846748184676694051320005681271452635608277857713427577896091736371787214684409012249534301465495853710507922796892589235420199561121290219608640344181598136297747713099605187072113499999983729780499510597317328160963185950244594553469083026425223082533446850352619311881710100031378387528865875332083814206171776691473035982534904287554687311595628638823537875937519577818577805321712268066130019278766111959092164201993.141592653589793238462643383279502884197169399375105820974944592307816406286208998628034825342117067982148086513282306647093844609550582231725359408128481117450284102701938521105559644622948954930381964428810975665933446128475648233786783165271201909145648566923460348610454326648213393607260249141273724587006606315588174881520920962829254091715364367892590360011330530548820466521384146951941511609433057270365759591953092186117381932611793105118548074462379962749567351885752724891227938183011949129833673362440656643086021394946395224737190702179860943702770539217176293176752384674818467669405132000568127145263560827785771342757789609173637178721468440901224953430146549585371050792279689258923542019956112129021960864034418159813629774771309960518707211349999998372978049951059731732816096318595024459455346908302642522308253344685035261931188171010003137838752886587533208381420617177669147303598253490428755468731159562863882353787593751957781857780532171226806613001927876611195909216420199

14654958537105079227968925892354201995611212902196086403441815981362977
47771309960518707211349999998372978049951059731732816096318595024459455
53469083026425223082533446850352619311881710100031378387528865875332083
81420617177669147303598253490428755468731159562863882353787593751957781
81857780532171226806613001927876611195909216420199 3.14159265358979323846
46264338327950288419716939937510582097494459230781640628620899862803482
25342117067982148086513282306647093844609550582231725359408128481117450
28410270193852110555964462294895493038196442881097566593344612847564823
23378678316527120190914564856692346034861045432664821339360726024914127
73724587006606315588174881520920962829254091715364367892590360011330530
05488204665213841469519415116094330572703657595919530921861173819326117
79310511854807446237996274956735188575272489122793818301194912983367336
62440656643086021394946395224737190702179860943702770539217176293176752
23846748184676694051320005681271452635608277857713427577896091736371787
72146844090122495343014654958537105079227968925892354201995611212902196
08640344181598136297747713099605187072113499999983729780499510597317328
28160963185950244594553469083026425223082533446850352619311881710100031
13783875288658753320838142061717766914730359825349042875546873115956286
63882353787593751957781857780532171226806613001927876611195909216420199
93.14159265358979323846264338327950288419716939937510582097494459230781
16406286208998628034825342117067982148086513282306647093844609550582231
17253594081284811174502841027019385211055596446229489549303819644288109
97566593344612847564823378678316527120190914564856692346034861045432664
48213393607260249141273724587006606315588174881520920962829254091715364
43678925903600113305305488204665213841469519415116094330572703657595919
95309218611738193261179310511854807446237996274956735188575272489122793
38183011949129833673362440656643086021394946395224737190702179860943702
27705392171762931767523846748184676694051320005681271452635608277857713
34275778960917363717872146844090122495343014654958537105079227968925892
23542019956112129021960864034418159813629774771309960518707211349999998
37297804995105973173281609631859502445945534690830264252230825334468503
52619311881710100031378387528865875332083814206171776691473035982534904
28755468731159562863882353787593751957781857780532171226806613001927876
61119590921642019 93.14159265358979323846264338327950288419716939937510
58209749445923078164062862089986280348253421170679821480865132823066470
93844609550582231725359408128481117450284102701938521105559644622948954
93038196442881097566593344612847564823378678316527120190914564856692346
03486104543266482133936072602491412737245870066063155881748815209209628
29254091715364367892590360011330530548820466521384146951941511609433057
27036575959195309218611738193261179310511854807446237996274956735188575
27248912279381830119491298336733624406566430860213949463952247371907021
79860943702770539217176293176752384674818467669405132000568127145263560
82778577134275778960917363717872146844090122495343014654958537105079227
96892589235420199561121290219608640344181598136297747713099605187072113
49999998372978049951059731732816096318595024459455346908302642522308253
34468503526193118817101000313783875288658753320838142061717766914730359
82534904287554687311595628638823537875937519577818577805321712268066130
01927876611195909216420199

```
05321712268066130019278766111959092164201993.14159265358979323846264338
83279502884197169399375105820974944592307816406286208998628034825342117
70679821480865132823066470938446095505822317253594081284811174502841027
70193852110555964462294895493038196442881097566593344612847564823378678
83165271201909145648566923460348610454326648213393607260249141273724587
70066063155881748815209209628292540917153643678925903600113305305488204
46652138414695194151160943305727036575959195309218611738193261179310511
85480744623799627495673518857527248912279381830119491298336733624406566
64308602139494639522473719070217986094370277053921717629317675238467481
84676694051320005681271452635608277857713427577896091736371787214684
40901224953430146549585371050792279689258923542019956112129021960864
03441815981362977477130996051870721134999999837297804995105973173281609
63185950244594553469083026425223082533446850352619311881710100031378
38752886587533208381420617177669147303598253490428755468731159562863
88235378759375195778185778053217122680661300192787661119590921642019
```