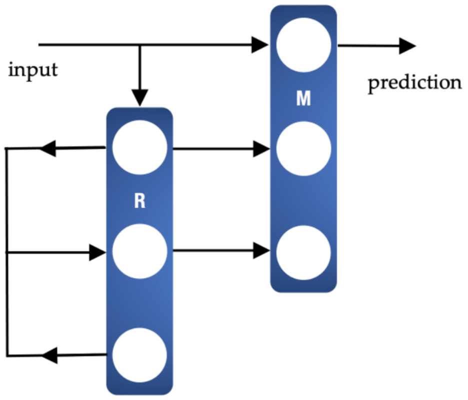
Deep Temporal Memory - Introduction

Peter Overmann, 08 Aug 2022

The most elementary temporal memory algorithm is composed of two triadic memory instances R and M, wired together in the following circuit:



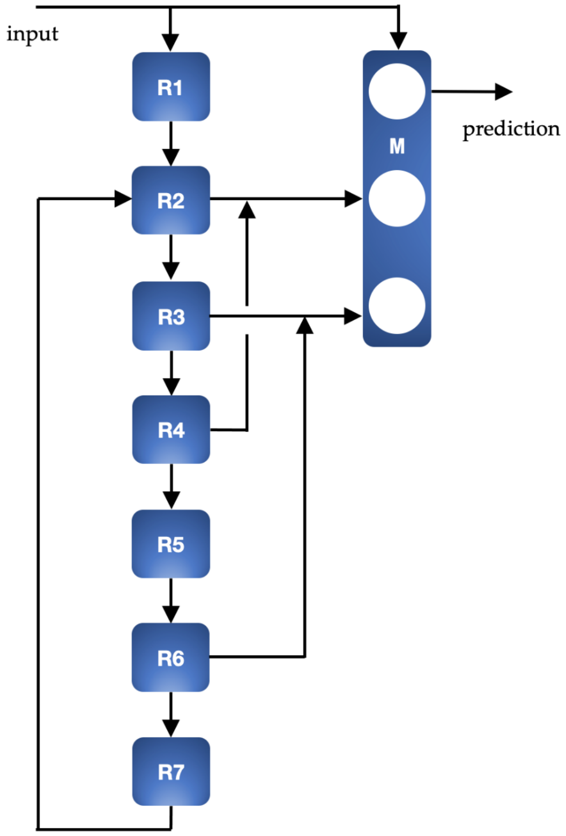
In this circuit, triadic memory R creates a random context vector for a consecutive pair of inputs, and feeds it back to the delayed input. This element effectively creates context-dependent bigrams from consecutive inputs.
Triadic memory M learns the association of the current input, and the readout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

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

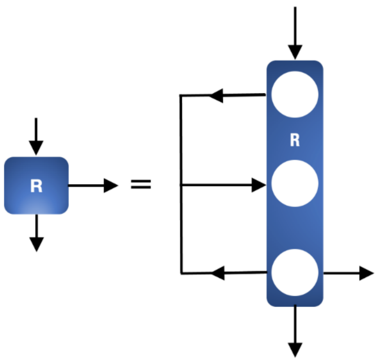
A circuit able to learn more complex patterns requires multiple feedback lines. There are infinitely many ways to design such circuits from triadic memory components. The following circuit diagram shows a simple example built from eight triadic memory units. The units R1 to R7 encode values to bigrams, bigrams to 3grams, and so on. At each step, feedback from the previous step is added to the inputs. There is an additional feedback loop from the output of R7 to the input of R2.

A readout memory M takes the state of selected bigram encoders (here chosen to be R2, R3, R4 and R6) 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 bigram encoders R represent triadic memory units with feedback loops.



Temporal Bigram Memory

TemporalBigramMemory is a circuit component consisting of a single triadic memory. 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.

Therefore in encodes two consecutive SDRs into bigrams (or higher-level representations).

Note: the input SDR need not be binary. The algorithm reduces it to binary using the ranked-max method from the triadic memory algorithm.

```
TemporalBigramMemory[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 = T[y + z];  (* binarize x and y using ranked-max algorithm *)
     y = T[input];

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

  ];
```

Deep Temporal Memory Algorithm

```
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}];

   (* reservoir units *)
   TemporalBigramMemory[#, {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]];

     t1 = R1[inp];
     t2 = R2[t1 + t7]; (* feedback from the output of R7 *)
     t3 = R3[t2];
     t4 = R4[t3];
     t5 = R5[t4];
     t6 = R6[t5];
     t7 = R7[t6];

     (* prediction readout from t1, t4, t2 and t6 *)
     z = M[x = M[t1 + t4], y = M[t2 + t6], _]

     (* Note: the function M[x+y]
       binarizes the sum of two binary vectors. BitOr[x,y] would
       give the same result if x and y don't overlap *)

    ]

  ];
```

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]
```

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.

1000 digits of pi:

```
temporalmemorytest [ ToString[N[Pi, 1000]], 8]
```

3.1415926535897932384626433832795028841971693993751058209749445923078164062862
08998628034825342117067982148086513282306647093844609550582231725359408
12848111745028410270193852110555964462294895493038196442881097566593344
61284756482337867831652712019091456485669234603486104543266482133936072
60249141273724587006606315588174881520920962829254091715364367892590360
01133053054882046652138414695194151160943305727036575959195309218611738
19326117931051185480744623799627495673518857527248912279381830119491298
33673362440656643086021394946395224737190702179860943702770539217176293
17675238467481846766940513200056812714526356082778577134275778960917363
71787214684409012249534301465495853710507922796892589235420199561121290
21960864034418159813629774771309960518707211349999998372978049951059731
73281609631859502445945534690830264252230825334468503526193118817101000
31378387528865875332083814206171776691473035982534904287554687311595628
63882353787593751957781857780532171226806613001927876611195909216420199
3.141592653589793238462643383279502884197169399375105820974944592307816
40628620899862803482534211706798214808651328230664709384460955058223172
53594081284811174502841027019385211055596446229489549303819644288109756
65933446128475648233786783165271201909145648566923460348610454326648213
39360726024914127372458700660631558817488152092096282925409171536436789
25903600113305305488204665213841469519415116094330572703657595919530921
86117381932611793105118548074462379962749567351885752724891227938183011
94912983367336244065664308602139494639522473719070217986094370277053921
71762931767523846748184676694051320005681271452635608277857713427577896
09173637178721468440901224953430146549585371050792279689258923542019956
11212902196086403441815981362977477130996051870721134999999837297804995
10597317328160963185950244594553469083026425223082533446850352619311881
71010003137838752886587533208381420617177669147303598253490428755468731
1595628638823537875937519577818577805321712268066130019278766111959092
164201993.141592653589793238462643383279502884197169399375105820974944
5923078164062862089986280348253421170679821480865132823066470938446095
5058223172535940812848111745028410270193852110555964462294895493038196
4428810975665933446128475648233786783165271201909145648566923460348610
4543266482133936072602491412737245870066063155881748815209209628292540
9171536436789259036001133053054882046652138414695194151160943305727036
5759591953092186117381932611793105118548074462379962749567351885752724
8912279381830119491298336733624406566430860213949463952247371907021798
6094370277053921717629317675238467481846766940513200056812714526356082
7785771342757789609173637178721468440901224953430146549585371050792279
6892589235420199561121290219608640344181598136297747713099605187072113
4999999837297804995105973173281609631859502445945534690830264252230825
3344685035261931188171010003137838752886587533208381420617177669147303
5982534904287554687311595628638823537875937519577818577805321712268066
130019278766111959092164201993.141592653589793238462643383279502884197
1693993751058209749445923078164062862089986280348253421170679821480865
1328230664709384460955058223172535940812848111745028410270193852110555
9644622948954930381964428810975665933446128475648233786783165271201909
1456485669234603486104543266482133936072602491412737245870066063155881

7488152092096282925409171536436789259036001133053054882046652138414695
1941511609433057270365759591953092186117381932611793105118548074462379
9627495673518857527248912279381830119491298336733624406566430860213949
4639522473719070217986094370277053921717629317675238467481846766940513
2000568127145263560827785771342757789609173637178721468440901224953430
1465495853710507922796892589235420199561121290219608640344181598136297
7477130996051870721134999999837297804995105973173281609631859502445945
5346908302642522308253344685035261931188171010003137838752886587533208
3814206171776691473035982534904287554687311595628638823537875937519577
818577805321712268066130019278766111959092164201993.141592653589793238
4626433832795028841971693993751058209749445923078164062862089986280348
2534211706798214808651328230664709384460955058223172535940812848111745
0284102701938521105559644622948954930381964428810975665933446128475648
2337867831652712019091456485669234603486104543266482133936072602491412
7372458700660631558817488152092096282925409171536436789259036001133053
0548820466521384146951941511609433057270365759591953092186117381932611
7931051185480744623799627495673518857527248912279381830119491298336733
6244065664308602139494639522473719070217986094370277053921717629317675
2384674818467669405132000568127145263560827785771342757789609173637178
7214684409012249534301465495853710507922796892589235420199561121290219
6086403441815981362977477130996051870721134999999837297804995105973173
2816096318595024459455346908302642522308253344685035261931188171010003
1378387528865875332083814206171776691473035982534904287554687311595628
6388235378759375195778185778053217122680661300192787661119590921642019
93.1415926535897932384626433832795028841971693993751058209749445923078
1640628620899862803482534211706798214808651328230664709384460955058223
1725359408128481117450284102701938521105559644622948954930381964428810
9756659334461284756482337867831652712019091456485669234603486104543266
4821339360726024914127372458700660631558817488152092096282925409171536
4367892590360011330530548820466521384146951941511609433057270365759591
9530921861173819326117931051185480744623799627495673518857527248912279
3818301194912983367336244065664308602139494639522473719070217986094370
2770539217176293176752384674818467669405132000568127145263560827785771
3427577896091736371787214684409012249534301465495853710507922796892589
2354201995611212902196086403441815981362977477130996051870721134999999
8372978049951059731732816096318595024459455346908302642522308253344685
0352619311881710100031378387528865875332083814206171776691473035982534
9042875546873115956286388235378759375195778185778053217122680661300192
7876611195909216420199 3.1415926535897932384626433832795028841971693993
7510582097494459230781640628620899862803482534211706798214808651328230
6647093844609550582231725359408128481117450284102701938521105559644622
9489549303819644288109756659334461284756482337867831652712019091456485
6692346034861045432664821339360726024914127372458700660631558817488152
0920962829254091715364367892590360011330530548820466521384146951941511
6094330572703657595919530921861173819326117931051185480744623799627495
6735188575272489122793818301194912983367336244065664308602139494639522
4737190702179860943702770539217176293176752384674818467669405132000568

```
1271452635608277857713427577896091736371787214684409012249534301465495
8537105079227968925892354201995611212902196086403441815981362977477130
9960518707211349999998372978049951059731732816096318595024459455346908
3026425223082533446850352619311881710100031378387528865875332083814206
1717766914730359825349042875546873115956286388235378759375195778185778
0532171226806613001927876611195909216420199 3.14159265358979323846264 33
8327950288419716939937510582097494459230781640628620899862803482534211
7067982148086513282306647093844609550582231725359408128481117450284102
7019385211055596446229489549303819644288109756659334461284756482337867
8316527120190914564856692346034861045432664821339360726024914127372458
7006606315588174881520920962829254091715364367892590360011330530548820
4665213841469519415116094330572703657595919530921861173819326117931051
1854807446237996274956735188575272489122793818301194912983367336244065
6643086021394946395224737190702179860943702770539217176293176752384674
818467669405132000568127145263560827785771342757789609173637178721468
440901224953430146549585371050792279689258923542019956112129021960864
034418159813629774771309960518707211349999998372978049951059731732816
096318595024459455346908302642522308253344685035261931188171010003137
838752886587533208381420617177669147303598253490428755468731159562863
882353787593751957781857780532171226806613001927876611195909216420199
```