Showing posts with label **Cortical Learning Algorithm**. Show all posts

Sunday, 9 November 2014

Cortical Learning Algorithms with Predictive Coding for a Systems-Level Cognitive Architecture

This is a quick post to link a poster paper by Ryan McCall, who has experimented with a Predictive-Coding / Cortical Learning Algorithm (PC-CLA) hybrid approach. We found the paper via Ryan writing to the NUPIC theory mailing list.

What's great about the paper is it links to some of the PC papers we mentioned in a previous post and covers all the relevant literature, with clear and detailed descriptions of key features of each method.

So we have Lee & Mumford, Rao and Ballard, Friston (Generalized Filtering)... It's also nice to see Baar's Global Workspace Theory and LIDA (a model of consciousness or, at least, attention).

Ryan has added a PC-CLA module to LIDA and tested robustness to varying levels of input noise. So, early days with the experiments but great start.

http://www.cogsys.org/papers/2013poster7.pdf

Posted by Dave Rawlinson at Sunday, November 09, 2014 1 comment :

Labels: Baar , CLA , Cortical Learning Algorithm , Friston , Global Workspace , Lee & Mumford , Predictive Coding , Rao & Ballard , Ryan McCall

Wednesday, 2 July 2014

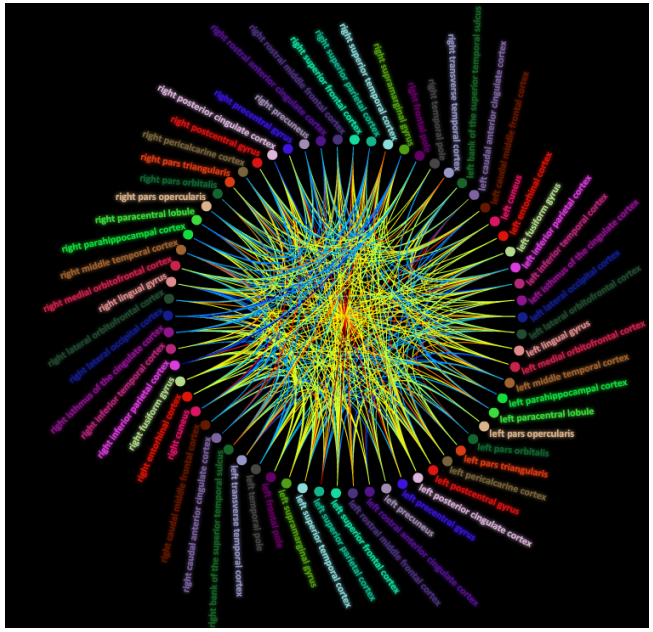
Connectome browser - hierarchy viewer?

Hierarchy Viewer?

Ongoing research to map out the patterns of connectivity in the human brain continues to produce stunning visuals. Are we looking at the hierarchy as proposed in the Memory-Prediction framework?

In any case, it's fun to explore, which you can do here:

http://www.humanconnectomeproject.org/data/relationship-viewer/



Posted by Dave Rawlinson at Wednesday, July 02, 2014 No comments :

Labels: Connectome , Cortical Learning Algorithm , Hierarchical Generative Models

Wednesday, 25 June 2014

TP 3/3: Proposed Temporal Pooler

submit

by David Rawlinson and Gideon Kowadlo

A Temporal Pooler Proposal

We will describe a Temporal Pooler design below. We have tested this design and it gives satisfactory output given the criteria and expectations described below and in the previous articles in this series (post #1, post #2). It is also relatively simple. You could imagine this design being implemented in biology, but we have no biological evidence that this is the design used in the cortex. Nevertheless, it is useful to have a template for an "ideal" design (from a certain

perspective), against which to compare known biology.

[EDIT: We have now rejected this design and prefer temporal pooling via on Predictive Coding]

This article will first list our design criteria, then give pseudocode for the TP implementation. Finally we will work through some examples to show the output of our temporal pooler.

Assumptions

1. This is a design based on engineering requirements, not based on biology

2. For simplicity of explanation, we will assume a "Region" is a piece of cortex comprising several layers of cells with distinct functions. There is mutual inhibition and high internal connectivity within the region. The region has a finite 2-d extent within all layers.

3. For a given set of active feed-forward (FF) input cells (let's call this an input "pattern"), each sub-region will have a single "winning" spatial-pooler (SP) cell that most closely matches the input pattern. This differs from the CLA/HTM definition of a region, where there may be multiple "winners". However, for this example let's keep things simple.

4. The finite set of cells local to the region mean that all configurations of world and agent are represented by states in a small, loopy graph embodied by the region's SP cells. "Loopy" means that there is at least one cycle and that there are no terminal states in this digraph. We understand that the local representation of agent + world state will be missing a lot of detail, that that's OK.

5. The temporal pooler will only update on a change to the FF input pattern.

6. Temporal Pooling should guarantee to increase output stability for any given sequence of SP cell activation. This means that sequences of length at least 3 must be replaced with constant output from the Temporal Pooler.

7. Uncertainty should not be hidden by the Temporal Pooler. Where forks occur in the graph, TP output should change, unless the fork is predictable. If an unpredicted event occurs, TP output should change to allow the transition to be modelled at a higher hierarchy level.

8. We assume that state-splitting creates redundant SP-cells that respond to the same FF input, but with different prior (previous) cell activations. e.g. SP cell #1 responds to uniquely to "green" after "blue", and SP cell #2 responds to "green" after "red". See the earlier post for more on this. A combination of state-splitting and simultaneous multiple TP-cell activation is necessary to guarantee pooling. It also allows variable-order modelling of FF input in different sequential contexts.

Outline

We require 2 sets of cells for our design. First, a layer of SP cells. A single SP cell fires for a given FF input, and represents the observation of that input when active.

Second, a layer of TP cells. We require 1 TP cell per SP cell. While SP cells fire individually, and only one at any given time, a set of TP cells will be simultaneously active. The set of active TP cells collectively represent a sequence of FF input, and the set must collectively fire only on observation of a specific and unique sequence of active SP cells. In a hierarchy, the next layer's SP cells will replace the active TP cells with individual cells that represent these combinations.

At all times at least 1 TP cell will be active.

The design has 2 key elements:
- Propagation of TP cell activity based on first-order modelling of historical transitions between SP cells.
- Assumes state splitting has created redundant SP cells representing the same FF input patterns, thereby allowing variable-order modelling using only first-order relations between SP cells.
- Inhibition of TP cell activity after a prediction failure.

Pseudocode
Variables:
`nCells`: The number of SP and TP cells
`spCells`: An array of SP cell activation values, length nCells
`inCells`: An array of TP cell inhibition values, length nCells
`tpCells`: An array of TP cell activation values, length nCells
`spBest`: The winning SP cell that matches the current FF input
`spPrev`: The previous winning SP cell.
`spPred`: The SP cell predicted to be the next best cell.
`w( i, j )`: Probability of transition from SP cell i to SP cell j in local graph

Note that weights w(i,j) represent conditional probabilities w(i,j) = P(S'=j | S=i) where S is the currently active SP cell and S' is the next active SP cell. The weights can be approximated using historical observations of transition frequencies between active SP cells. Learning of sequence memory transitions is not covered in the pseudocode below. Similarly, the process for learning and defining the set of SP cells is not described.

```
// clear tp cell activity after bad prediction
if( spPred != spBest ) { // last prediction wrong
  inCells[ spPrev ] = 1;
  tpCells[ all ] = 0;
```

```
}

// clear tp cell activity when we reach inhibited cell
if( inCells( spBest ) ) {
  tpCells[ all ] = 0;
}

// clear inhibition of winning cell and activate it
inCells[ spBest ] = 0
tpCells[ spBest ] = 1

// now propagate this activity forwards
sp1 = spBest;

int i = 0;
while( i < nCells ) { // or stop when stable

  sp2Max = null;
  wMax = 0;

  // find the most likely transition starting at sp1
  for( sp2 = 0; sp2 < nCells; ++sp2 ) {
    if( sp1 == sp2 ) continue;

    if( w( sp1, sp2 ) > wMax ) {
      wMax = w( sp1, sp2 )
      sp2Max = sp2;
    }
  }

  // check inhibition, and consider replacing
  // sp1 with sp2Max to continue propagation.
  if( sp2Max != null ) {
```

```
    if( inCell[ sp2Max ] > 0 ) {
      break; // don't propagate to this cell, it is inhibited
    }
    else { // not inhibited
      tpCells[ sp2Max ] = 1; // activate
      sp1 = sp2Max; // now project forward from here
    }
  }

  ++i;
}
```

So what does it do?

Let's work through a simple example that includes deterministic sequences and a non-deterministic fork. Obviously, the latter cannot always be predicted.

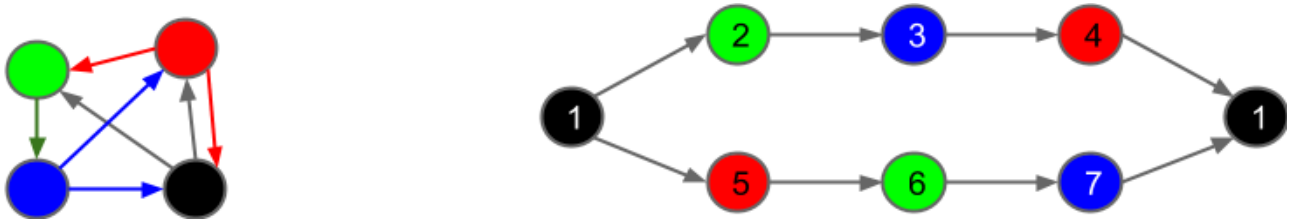


Figure 1: First order (left) and variable order via state-splitting (right) representations of a problem. The problem is characterized by observation of a colour in each state. There are 7 states (some colours occur in more than 1 state). The 7 states are organised into 2 sequences (R,G,B and G,B,R). After state 1 (black), a random sequence is chosen.

Figure 1 shows a test problem where each state has a corresponding colour observation. Four different colours are observed (red, green, blue and black). Arrows indicate transitions between states. The "true" graph of the problem is shown to the right. On the left is a first-order graph of observations. State-splitting is necessary to build a more useful model of the transitions between states, creating redundant copies of states with red, green and blue observations. Let's just assume that the SP cells have formed the "correct" representation of the world shown (figure 1, right) because that learning process is too complex to explain here. The fork at state 1 is random, with a slight bias making transition from 1 to 2 more likely than from 1 to 5. State 1 is shown twice for improved presentation.

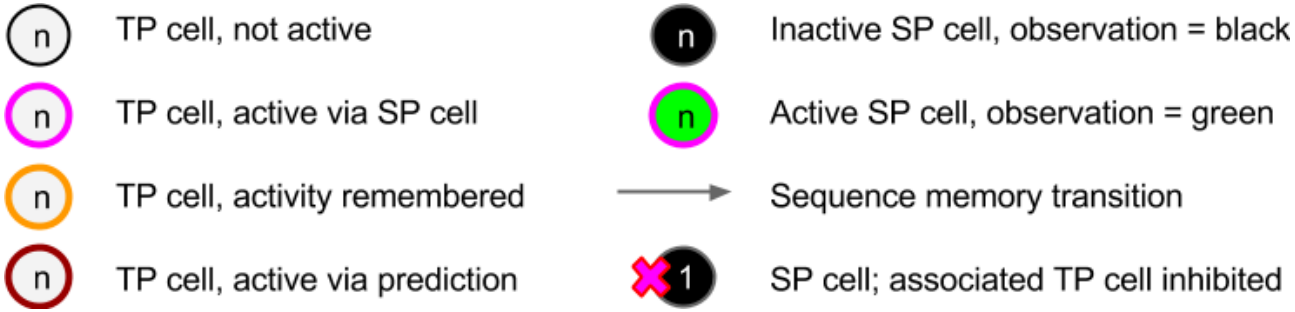


Figure 2: Key for diagrams used in the rest of this post. We have 2 sets of cells: TP cells (left column styles) and SP cells (right column styles).

Figure 2, above, shows the annotations and styling we use to present the state of SP and TP cells in the following walkthrough of the proposed temporal pooling algorithm.

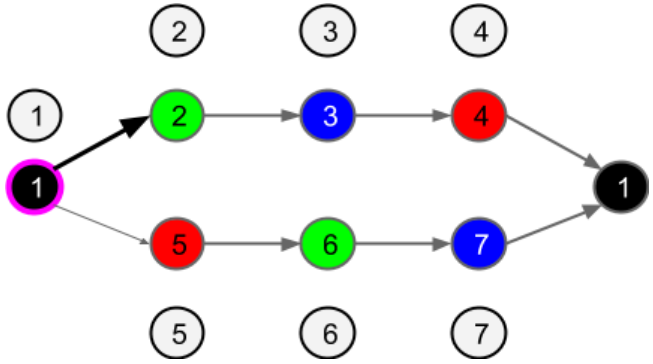


Figure 3: SP cell 1 is active. Each SP cell has an associated TP cell (none active yet).

Figure 3: Say that via state-splitting, we have constructed a set of 7 SP cells (centre graph, coloured and numbered circles). This means we also need to create 7 TP cells (we specified earlier that there would be 1 TP cell per SP cell). Each TP cell is associated with one SP cell. The magenta outline of SP cell 1 indicates that this cell has been selected as best matching the current FF input. None of the TP cells are active. The heavier arrow from SP cell 1 to 2 represents the higher probability of this outcome.

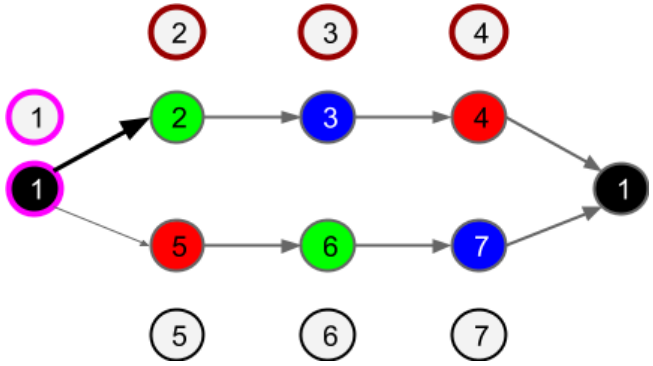


Figure 4: TP cells active via prediction.

Figure 4: According to the algorithm given above, we will first activate TP cell 1 due to activity of SP cell 1 (magenta). Then, we propagate forwards the activation through TP cells 2,3,4 because this is the most likely path (brown outline highlights). Activation propagates back to TP cell 1 and then has no further effect, since TP cell 1 is already active. TP cells 5,6,7 are not activated because this is not the most likely path from TP cell 1. As a result, of our 7 TP cells, 4 are active {1,2,3,4} of {1,2,3,4,5,6,7}.

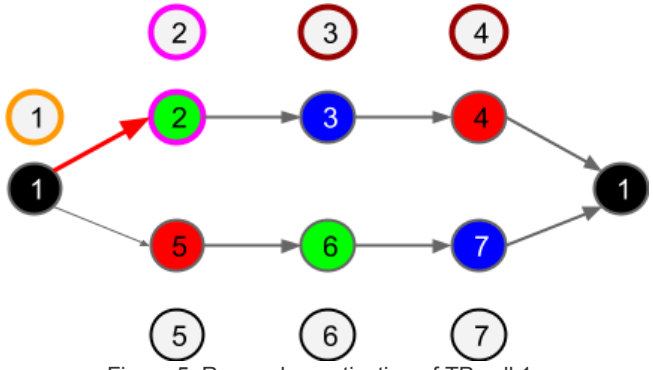


Figure 5: Remember activation of TP cell 1.

Figure 5: The external world moves from a state generating the black observation, to green. SP cell 2 is correspondingly made active. The temporal pooler cells are now updated, starting by activating TP cell 2 (although it is already active). TP cell 1 remains active for two reasons: First, activity is not cleared unless there is a prediction failure. Second, activity propagates from TP cell 2 through TP cells 3,4, and 1, back to 2. Again, TP cells 5,6,7 are not active. There has been no change in the set of active TP cells as they are following a predictable sequence. In fact, the set of active TP cells will not change while visiting any of the active states {1,2,3,4}.

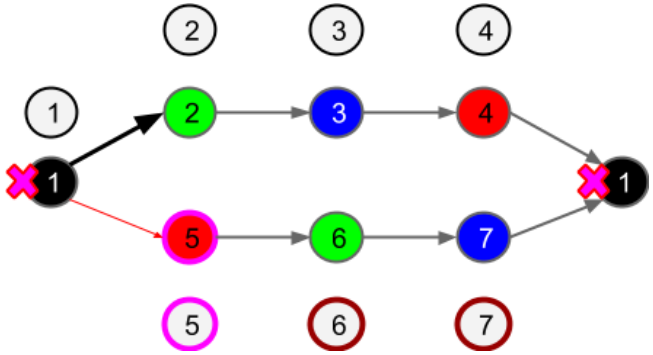


Figure 6: Prediction failure. We predicted a transition from SP cell 1 to 2, but observed from 1 to 5. Inhibit activation of TP cells associated with SP cell 1.

Figure 6: If, randomly, a transition from SP cell 1 to SP cell 5 occurs, the pattern of TP cell activation will change. TP cell 1 will be inhibited (magenta X) due to the prediction failure. TP cell 5 will be activated and the activity will propagate through TP cells 6 and 7. Crucially, it cannot propagate further due to inhibition of TP cell 1. This prevents the set of active cells extending to TP cells 2,3,4.

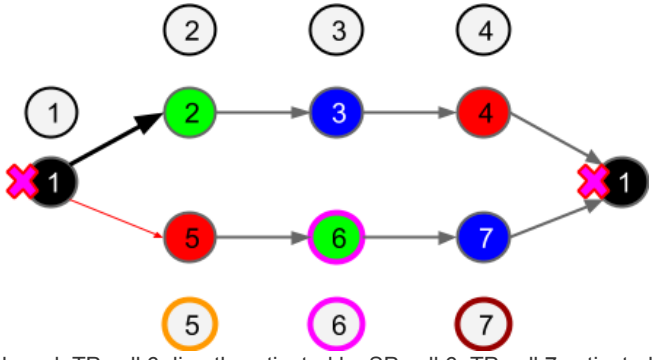


Figure 7: TP cell 5 activity remembered. TP cell 6 directly activated by SP cell 6. TP cell 7 activated by prediction. TP cell 1 inhibited.

Figure 7: For the remainder of the sequence R-G-B, TP cells 5,6,7 remain active. While Green is observed, as shown above, TP cell 5 remains active not due to forward prediction but due to the fact that TP cell activity is retained until there is a prediction failure.

Discussion

In this example, TP output is one of two subsets of cells, either {1,2,3,4} or {5,6,7}. In the next hierarchy layer, the active subsets would be associated with specific SP cells that would represent these sequences of observations in their entirety.

This is a relatively simple scheme that only requires inhibition and prediction. A prediction failure is **always** followed by a change in TP output, due to the inhibition mechanism, even if the most likely path leads back to the original set of active cells prior to the failed prediction. Recall that the local graph being

modelled in a region of cortex will always feature cycles and have no terminal states. Without inhibition, we would end up with scenarios where all TP cells were constantly active regardless of which fork was taken (if a lower-probability outcome occurs). This would obscure the unpredictable transitions and make them unavailable for modelling higher in the hierarchy.

At least 1 TP cell is always active (the cell that is associated with the currently active SP cell).

The output of the temporal pooler is maximally stable when sequences of observations are predictable. This is ideal if we assume the predicted outcome is the most common outcome. The propagation mechanism as proposed makes best use of predictable transitions - sequences of arbitrary length can be replaced with a constant output, if the system is sufficiently predictable.

Note that this method can only guarantee to simplify the observed problem in combination with state-splitting. The latter is required to build more and longer deterministic sequences of SP cell activation, even if these sequences are not predictable. It is assumed that given increasingly higher-order interpretation, predictability will be achieved.

In a future post we will look in more detail about the process of state-splitting.

Posted by Dave Rawlinson at Wednesday, June 25, 2014 No comments :

Labels: Cortical Learning Algorithm , Temporal Pooling

Saturday, 31 May 2014

TP 2/3: Jeff's new Temporal Pooler

By David Rawlinson and Gideon Kowadlo

Jeff's new Temporal Pooler

This is article 2 in a 3 part series about Temporal Pooling (TP) in MPF/CLA-like algorithms. You can read part 1 here. For the rest of this article we will assume you've read part 1.

This article is about the new TP proposed by Jeff Hawkins. The original TP was described in the CLA white paper. We will also assume you've at least had a quick read of the linked articles. Despite our best efforts this article is only an interpretation of those methods, and it may not be entirely correct or as Numenta intended.

Separation of internal and external causes

The first topic in Hawkins' proposal covers the possible roles of specific cortical layers and the separation of internal and external causes.

Hawkins suggests that cortical layers 3,4,5 and 6 are all implementing variants of the same algorithm, with minor differences. He also suggests that each layer is performing all functions (Spatial Pooling, Sequence Memory and Temporal Pooling. In higher hierarchy levels, Spatial Pooling may be absent). In CLA, these 3 components are implemented as a matrix of sequence-memory cells. Rules concerning how and when the cells activate each other in different input contexts implement the pooling and prediction features.

Hawkins also states that one distinction between layers 3 and 4 may be that cells in layer 4 are using copies of motor actions (internal causes), to predict better. In consequence, cells in layer 3 are left trying to predict the actions that layer 4 could not predict, i.e. relying more on historically-observed sequential patterns of activation. Although both layers will learn sequential patterns of activation, layer 3 will rely more heavily on history. External causes will more often generate input that can't be explained by motor actions, so we might expect layer 3 to more often respond to external events.

This article will not discuss these ideas any further. We note them for clarity and to distinguish our focus. Instead, we will talk about how the new CLA TP could be used to construct a hierarchical representation of changes in input patterns over time.

Temporal Slowness

Both old and new temporal poolers exploit a principle known as "Temporal Slowness", which means that output activity varies more slowly than input activity. You can read more about this general principle here.

Another, related feature of the HTM and CLA temporal poolers is that they emit a constant pattern of cell activity to generate stability. This is achieved by marking cells as "active" regardless of whether they are active via prediction or via feedfoward input. Although active-by-prediction and active-by-FF-input are distinguished within the region for learning purposes, this distinction is not visible to the next higher region in the hierarchy.

Old Temporal Pooler Cell Activity

For reference, this is an outline of the TP functionality in the "old" TP from the Numenta CLA white paper.

The diagrams in this article each have 3 parts. At the top is a graph showing a fragment (in graph terminology, a component) of the Sequence Memory encoded by the cells in the region. Arrows show the learnt transitions between cells. Below the graph is a series of observations (marked "FF input") and the corresponding pattern of cell activity when each Feed-Forward (FF) input is observed. Each column represents a single cell from the sequence memory and its activity over time. Each row in the lower part of the diagrams shows all cells' activity at one moment. Cells are filled white when they are active and black when not active:

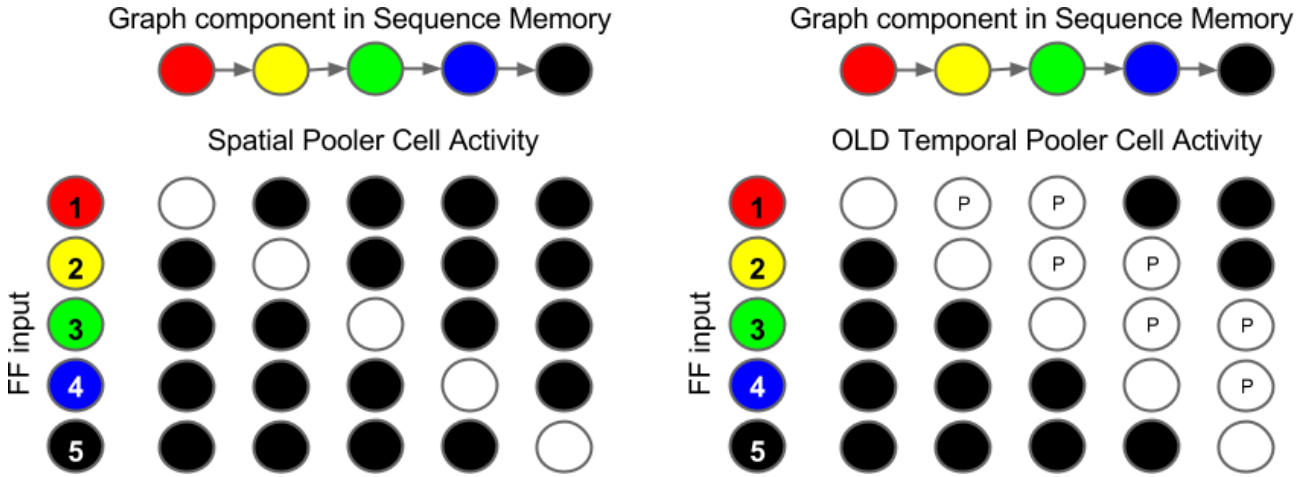


Figure 1: Spatial pooler and temporal pooler cell activity in the original CLA white paper. This image compares two patterns of cell activity over time, shown left and right. The left subfigure shows cell activity in spatial pooler cells, where the active cell[s] are the ones whose input bits most closely match current FF input. The right subfigure shows the original CLA temporal pooling method, where cells become active when predicted far in advance of their FF input being observed, and remain active until after the associated FF input is observed. In this example, ⅔ of the active cells are identical after each FF input change. A ‘P’ denotes cells activated due to prediction. Each subfigure has 3 parts. The main part is a matrix showing sequence memory cell activity over time. Each row is one time-step, numbered 1 to 5. The FF input observed at each time step is shown in a column to the left. The top row shows a fragment of Sequence Memory formed by the cells, and the colours each cell responds to. In this simple example, the Sequence Memory graph is simply a sequence of states that are always observed in the same order.

Spatial Pooler cell activity is easiest to explain (figure 1, left). In figure 1, we see that the Sequence Memory has learnt that the colours Red, Yellow, Green, Blue and Black occur in order. One cell responds uniquely to each of these colours, creating the diagonal line of active cells over time. Each cell is only active for the duration of its FF input being observed.

The original temporal pooler premise was to drive cells to an active state via prediction (cells active via prediction are marked with a 'P') as early as possible. The cells would then remain active until either the prediction was no longer made (e.g. due to observation of an unexpected FF input pattern) or the cell becomes active via its FF input.

Time and Stability

You can see in the figure above that in a sequence of predictable inputs each cell is active over a period of 3 input changes (the only meaningful way to measure time in these examples). So let's assume each input change is one time step.

The cells are shown to be active for an arbitrary period of time - to keep the diagrams simple the minimum period is shown. In reality a fixed activation period is unlikely; it will depend on the activation or prior cells in the sparse distributed representation. However, it is still possible to make the point that at any time during a predictable sequence, a set of cells is active. Most of those cells are not changing between time steps and will be active after the next FF input change. This is the temporal pooler in action.

In the example above, each cell is predicted up to 2 steps before the corresponding input is observed. Therefore, in a predictable sequence 3 cells are simultaneously active, 2 of them due to prediction and one due to FF input.

Although the output of the temporal pooler is continuously changing, most of the active cells are not changed between inputs. In this case, 2/3 of the output is stable. With longer activation of TP cells, a larger fraction of the output becomes stable.

Noisy recognition and resource constraint assumptions

To simplify the problem observed by the next level in the hierarchy it is necessary to have cell activity changes in the lower level without corresponding cell activity changes in the higher level. Given that the TP output is continuously changing, how do we sometimes avoid cell activity changes in the higher level?

There are two parts to the answer. First, all cells' FF input are Sparse Distributed Representations (SDRs). These are large sets of input bits, of which only a few are active at any given time. The Spatial Pooler in CLA recognises FF inputs when only a fraction of the expected (synapsed) input bits are active. For example, a cell may become active from FF input when 80% of its FF input bits are active - any 80%. The set of active input bits can change while the cell is active. This means that cells' recognition is tolerant to noisy FF input.

Noisy recognition of TP output from lower hierarchy levels is one assumption necessary for increasing stability in higher levels. But this assumption is actually a useful feature, allowing classification to be tolerant to noise.

The other necessary assumption is a resource-constraint. If an infinite supply of cells were available, then after much slow learning every FF input pattern would have a dedicated cell (due to inhibition between cells). Cell activity changes would occur throughout the hierarchy after every input change, no matter how tiny. Obviously, resource constraints are physically necessary.

The finite size of a CLA region ensures that there aren't enough cells to represent each FF input pattern perfectly. Instead, some similar (probably successive)

FF inputs will be represented by the same set of active cells (quantization error).

These are both very reasonable assumptions, but worth stating and understanding.

New Temporal Pooler Cell Activity

The new TP proposes that cells are only counted as active when confirmed by observation of the corresponding FF input, and that they stay active for a period of time after this:

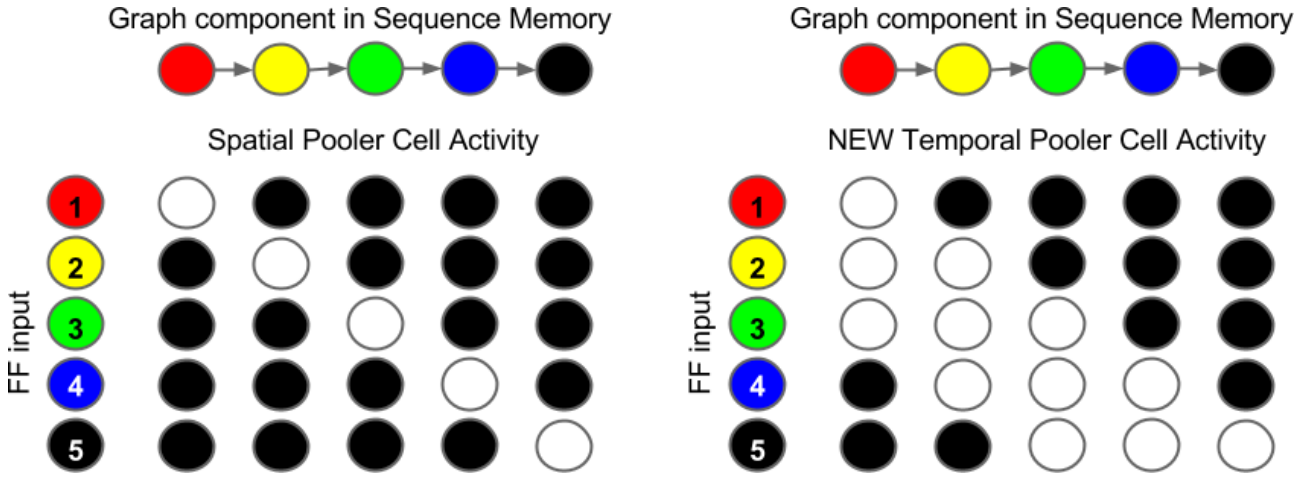


Figure 2: Spatial pooler and temporal pooler cell activity as described by the new TP method. Each TP cell is active for a period of time after its corresponding FF input is observed. There may be no distinct SP or TP cells, but we show them separately to illustrate differences in activation behaviour.

This in itself doesn't change much, but because we are now building patterns forwards we can represent unpredicted events more accurately. Cells are no longer active until the corresponding FF input is observed or prediction is cancelled; instead they are never fully activated prior to the corresponding FF input. When a prediction error occurs, the results are immediate and lasting. Given noisy recognition of FF input, the old method would be more likely to have hidden prediction failures.

Temporal pooling “replaces” graph components (specifically sequences of vertices) with a single vertex that represents the component by being constantly active for the duration of those inputs. It is also worth noting that to simplify any graph, the minimum number of vertices in each replaced sequence is 3. In a temporal pooler, this means that the minimum number of FF input changes for which a predicted cell must remain active is 3. If temporal pooler output is constant for sequences of length 2, the next hierarchy level will encode transitions between cells instead of sequences of cells (i.e. no simplification or effective

pooling has occurred).

Activity after a Successful Prediction

The new TP proposes that there are two cortical layers of cells. One layer of cells embodies the Spatial Pooler. The other layer forms the TP. In the TP layer, cells remain active for a long time after being successfully predicted in the SP layer, but for only a short time when not predicted in the SP layer.

Prediction failures will occur regularly, whenever there are multiple future states and the available data does not allow the correct future to be determined. This looks like a fork in the Sequence Memory graph:

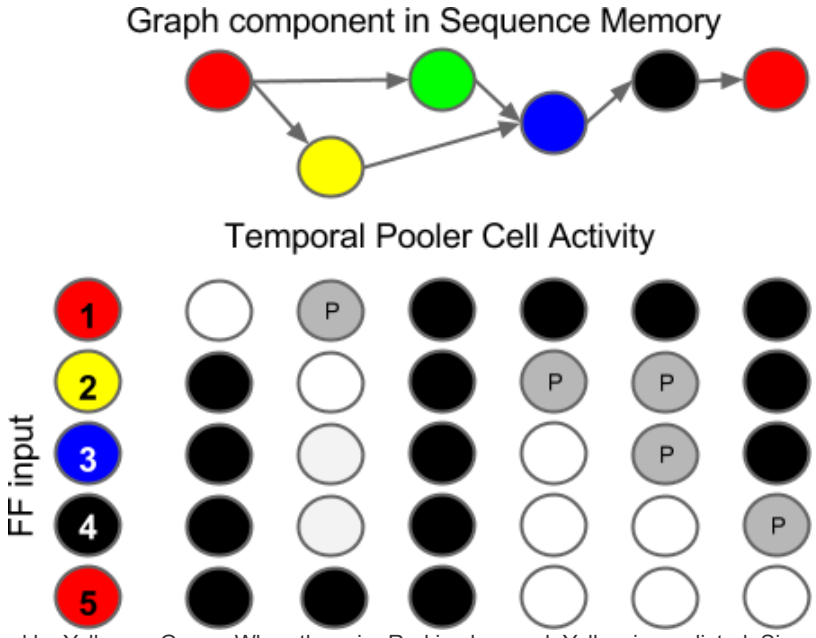


Figure 3: In this non-deterministic sequence, Red is followed by Yellow or Green. When the prior Red is observed, Yellow is predicted. Since it was predicted, the Yellow cell stays active for 3 steps in total. Since Blue always follows Yellow and Green, and Black follows Blue, the other cells are all active for the full 3 steps.

In the example above, the Yellow cell is successfully predicted leading to a long activation of the sequence memory cell that responds to Yellow after Red.

Activity after a Failed Prediction

The new TP proposes that in the event of a failed prediction, cells only remain active briefly. This is shown in the example below, where Yellow was predicted but Green was observed:

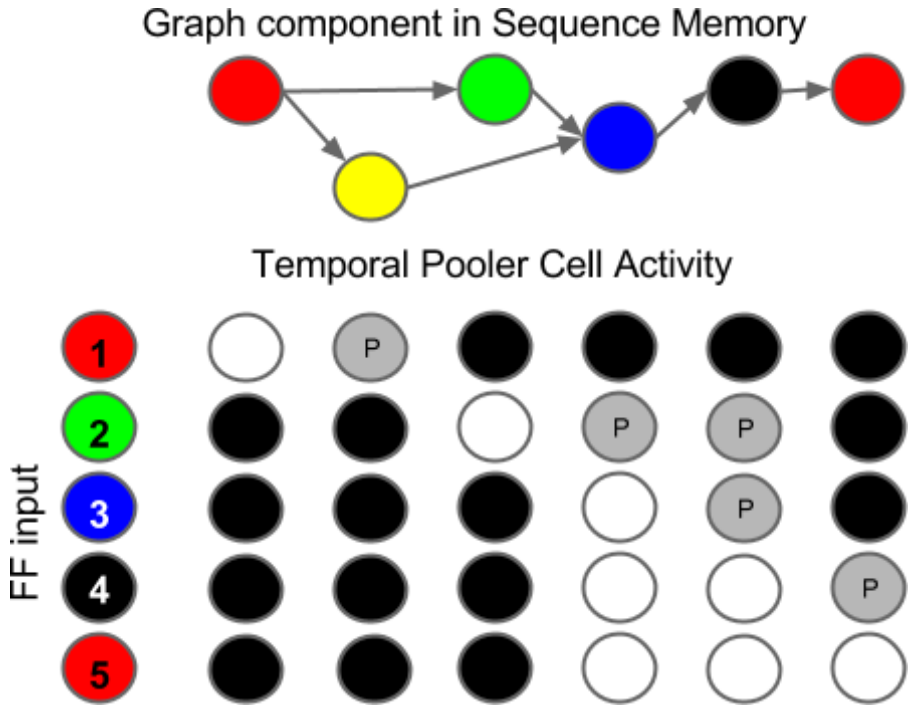


Figure 4: New TP cell activity after a failed prediction. Yellow was predicted but Green was observed.

The pattern of activity after a failed prediction is initially different to the pattern after a correct prediction, with only a short activation of the Green cell and no full activation of the predicted cell at all. This means that now, cells are only active when their corresponding FF input is actually observed.

The FF output of the TP after a prediction failure is quite different to the FF output during predictable sequences before and after. This helps to ensure that the unpredictable transition is modelled in higher hierarchy levels, passing the problem up the hierarchy rather than obscuring it. We anticipate that higher levels of the hierarchy will have the ability to understand and hence predict the problematic transition.

Analysis

Prediction with/without motor output distinguishes Cortex layers 3,4

Copying motor actions back to the cortex to help with prediction makes sense, especially in lower hierarchy levels. However, recent motor signals become increasingly irrelevant when trying to predict more abstract, longer term events. For example, getting sacked from your job is less likely due to the way you just now sipped your coffee, and more likely to do with some events that happened days or weeks ago. These older events will be hierarchically represented as more abstract causes.

At higher hierarchy levels, with greater abstraction, the "motor actions" that are necessary to explain & predict events are not simple muscle contractions, but complex sequences of decisions and behaviour with specific intents and expectations. The predictive data encoded in the Feed-Forward Indirect and Feed-Back (FB) pathways contains this data in a form that is appropriate and meaningful at each level of the hierarchy. If predictions and decisions are synonymous, then we can treat selected predictions as if they were actions.

For these reasons we are skeptical about the idea that the use of motor actions to aid prediction is enough to distinguish the functionality of different cortical layers. However, in support of the idea, layer 4 does disappear in higher hierarchy levels where motor actions would be less relevant.

Propagation of uncertainty to higher regions

The way that uncertainty (as prediction failure) is propagated up the hierarchy is vital to being able to reliably assemble a useful hierarchical representation of FF input. In fact, we believe that unpredictable events should be propagated until they reach a level of abstraction and invariance where they become predictable (see the Newtonian world assumption in the previous post). Therefore, we believe TP output should be highly orthogonal to prior and subsequent output in the event of a prediction failure. In the case of an SDR, highly orthogonal means that many bits should have dissimilar activity (a small intersection between the sets of active bits before and after).

Only a fraction of synapsed input bits are needed to activate a cell, and therefore CLA features “noise-tolerant” recognition of FF input. Only a few output bits would be dissimilar between the outcomes of prediction success and failure. This seems to raise the risk that unpredictable events could be “hidden” by noise-tolerance, and not passed up the hierarchy for higher levels to solve. From the perspective of a higher level, the set of active cells has not been significantly affected by their failure to predict.

Some loss of uncertainty in propagation may be acceptable in a sufficiently complex system. These are toy examples with only a few cells, whereas real CLA regions have hundreds or thousands of cells. However, we are working through some simple examples to try to better understand the behaviour and limits of the CLA.

Another detail that is not fully described in Hawkins’ current TP proposal is how long cells should be active when predicted. Maximum stability is achieved when cells are active for long periods, but we are limited by the conflicting objective to not hide uncertainty. Should we truncate activity when other prediction failures occur? In the next article we will propose explicitly making TP cells active unless uncertainty is too high, thereby implementing an auto-tuning of activation period.

Representations of random sequences

There is one comment in Hawkins' proposal that we disagree with. He says: "One of the key requirements of temporal pooling is that we only want to do it when a sequence is being correctly predicted. For example, we don't want to form a stable representation of a sequence of random transitions." In fact, it may be necessary to build a framework of some random sequences, in order to build sufficiently complex representations to explain any of the simpler events. Although the random sequences may not be the right ones, we need to have a mechanism of assembling more complex hierarchical representations even when there is no incremental explanatory power in doing so (this was discussed in the previous article). This would mean looking for structure in randomness, on the assumption that it would eventually be worthwhile due to explanatory models in higher levels of the hierarchy.

Summary

To wrap up:
- Feed-Back or Feed-Forward indirect pathway data may be a more appropriate source of data than motor actions in the FF direct pathway, for predicting events based on internal causes at higher levels of abstraction

- Reliable propagation of uncertainty (prediction failure) up the hierarchy is critical to move unexplained events to a level of abstraction where they can be understood

- We would like to extend activity period for maximum stability, balanced against the desire to avoid hiding prediction errors. How this is done is not detailed

- It may be necessary to perform temporal pooling even when there are no predictable patterns, in order to construct higher-order representations that may be able to predict the simpler events.

The next and final article in our 3 part series will present some specific alternative temporal pooler ideas.

Posted by Dave Rawlinson at Saturday, May 31, 2014 2 comments :

Labels: Cortical Learning Algorithm , HTM , Sparse Distributed Representations , Temporal Pooling

Tuesday, 27 May 2014

Thalamocortical architecture

by Gideon Kowadlo and David Rawlinson

Introduction

One of the keys to understanding the neocortex as a whole, and the emergence of intelligence, is to understand how the cortical hierarchical levels interconnect. This includes:

- the physical connections,
- the meaning of the signals being transmitted,
- and possibly also the way the signal is encoded.

Physical connections: Physical connections refer to gross patterns of neuron routing throughout the brain. This is known as the connectome. Below is an image from the Human Connectome Project, that beautifully illustrates many connections including thalamocortical ones.

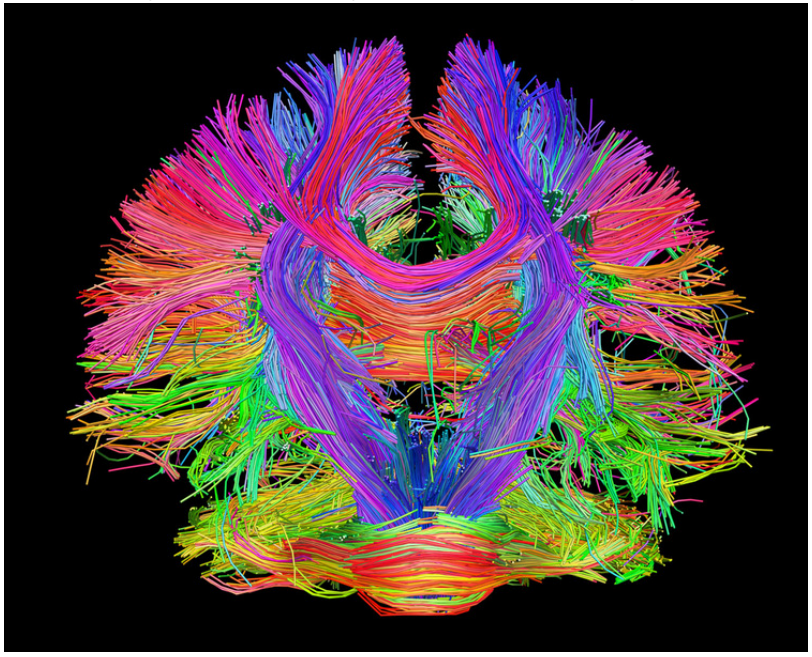


Figure 1. Courtesy of the Laboratory of Neuro Imaging and Martinos Center for Biomedical Imaging, Consortium of the Human Connectome Project - www.humanconnectomeproject.org

Meaning of signals: One classification that can be applied to thalamocortical neurons is drivers versus modulators. A driver can be thought of as a neuron that carries information, whereas a modulator modulates or alters the transmission of information in a driver. They have different functional and anatomical properties, as nicely described in (Sherman and Guillery 2011). If a neuron is a driver, what information does it encode, and if it is a modulator, is it inhibiting or excitatory and what effect does this have?

Signal Encoding: Signal encoding refers to the details of how the information is represented. This includes timing and amplitude information. The way the signal is encoded in the neurons may have a bearing on the properties of the system. Specific information has been added to the diagram where this looks relevant.

Our aim is to build AI with general intelligence characteristic of biological organisms such as primates. Therefore, we draw inspiration and insight from these working examples. Understanding the biology obviously gives us the best insight into how to do that. However, what level of abstraction do we need to capture the essential qualities?

- at the lowest level: molecular structure, interactions and neurotransmitters,
- above that, firing patterns and newly discovered molecular machinery (that excitingly shows this is more complex and interesting than previously thought - see paper and work by Seth Grant),
- higher still, the brain as a set of modules that interact with each other,
- or multi scale simulation of the whole brain (see the Human Brain Project).

For simplicity, we want to understand it at the highest level that is still capable of capturing the essential qualities, and drill down where necessary. Therefore, are factors such as the way that the signal is encoded important? Not in and of themselves, but they may have a bearing on emergent qualities, that *are* significant.

In order to understand the above, including drawing conclusions about the appropriate level of abstraction, we've elaborated on a figure first published in the CLA White Paper that was included in a previous post (in the section 'Regions'). In that article, we started to explore these topics in the context of Numenta's work. The figure shows the thalamo-cortical connections to specific cortical layers and is very useful for exploring the concepts described above. Here, we will expand on that figure, shown below. We will go over a first version, and we plan to make further posts in the future, as we develop it further. Each of the initial annotations are explained in the sub sections below.

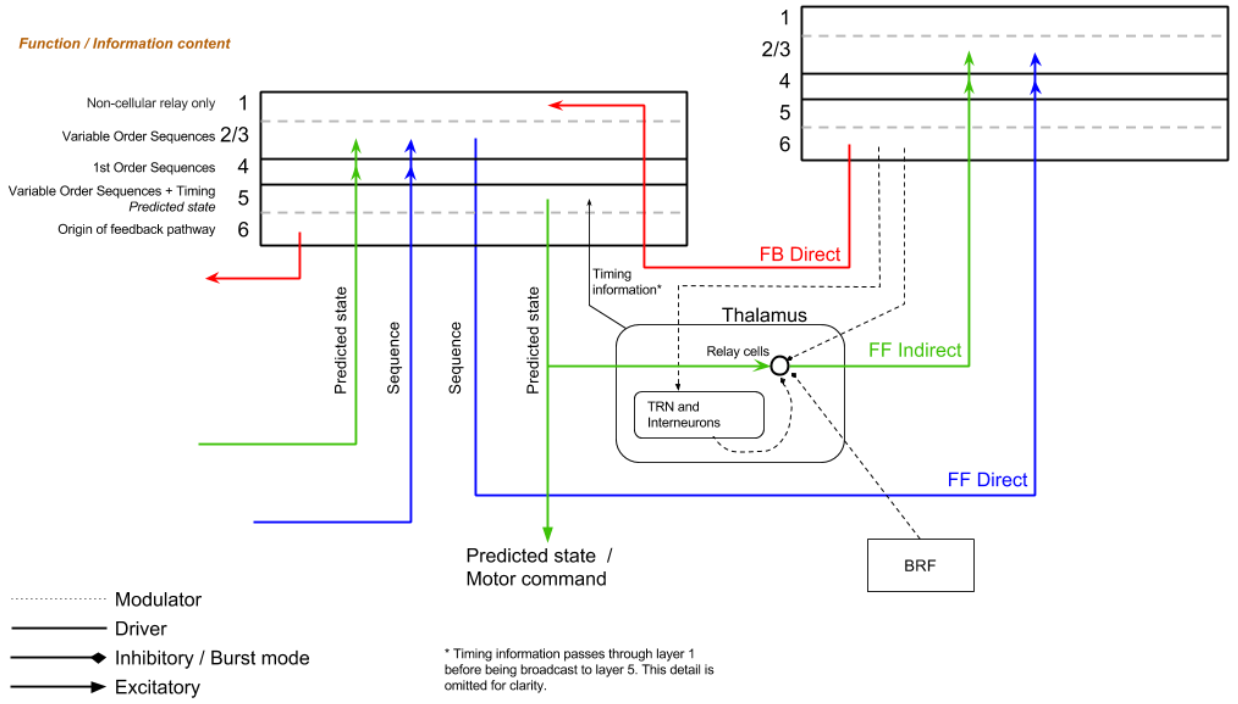


Figure 2. Thalamocortical architecture including cortical layers and connections between hierarchy levels. This figure is an annotated version of a figure from the 'CLA White Paper'. Some information is added from the text of that document. Other sources used are Sherman and Guillery 2011, Grossberg 2007, Sherman 2006 and Sherman 2007.

We invite the community to make use of and contribute to this annotated diagram. The diagram is publicly available in a universal vector graphics format called SVG. Being vector based, it is easily modifiable. SVG is a common format, which many graphics packages are capable of editing.

The file is available from a git repository hosted on github called cortico-thalamic-circuit. Anyone can download, clone, make a pull request or fork the repository.

Pull requests allow you to make modifications and then give them back to the shared repository so that they are available to everyone. This is the action to take if you share our purpose for the diagram - staying as high level as possible, filling in details where they contribute to a holistic view or emergent properties of the thalamocortical architecture. Forking allows you to create a new repository that diverges from the main one. Use this option if you'd like to use the diagram for a different purpose, such as documentation of all the neurotransmitters in the different pathways.

The first set of diagram additions are described below.

Diagram Additions

Cortico-Cortical Feedback

The illustrated feedback between levels from layer 6 in Level (n+1) to layer 1 in Level (n) is described briefly in the CLA white paper. We have included an additional illustration from Grossberg 2007 (see figure 3 below), that shows in more detail how internal neural circuitry completes the intra-cortical, inter-level, feedback loop from:

H[n+1]:C[6] → H[n]:d[1]C[5]
H[n]:d[1]C[5] → **H[n]:C[6]**
H[n]:C[6] → H[n]:C[4]

Note: The connections above are described in a notation we have adopted for succinctly describing cortical neural pathways. Refer to our post for more details.

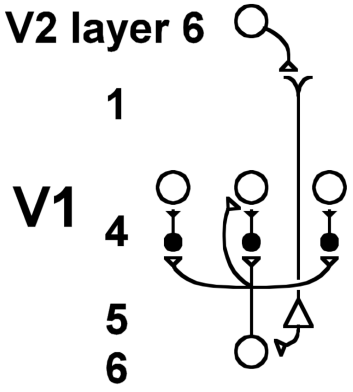


Figure 3. Inter-level feedback loop, reproduced from Grossberg 2007. The circles and triangles are neuron bodies, with varying shape depicting different neuron types. Two hierarchy levels are shown (V1,V2 from the visual cortex). Each hierarchy level has 6 cortical layers (numbered 1 to 6 where relevant). You can see that feedback from V2 affects activation of neurons in V1 layer 4.

The feedforward/feedback architecture gives rise to at least three important qualities, the first of which has been explored in the MPF literature. They are described below, reproduced from Grossberg 2007:

1. the developmental and learning processes whereby the cortex shapes its circuits to match environmental constraints in a stable way through time;
2. the binding process whereby cortex groups distributed data into coherent object representations that remain sensitive to analog properties of the environment; and
3. the attentional process whereby cortex selectively processes important events.

We may elaborate with a summary of Grossberg 2007 in a future post.

Gating by the Thalamus

Our main references for this section are Sherman 2006 and Sherman 2007.

We've seen that the thalamus acts as a relay for information passing up the hierarchy between cortical levels, which we're referring to as the feedforward indirect pathway (FF Indirect). It has been postulated that via this gating, the thalamus plays an important role in attention.

What inputs and computations determine that gating? This is one of the questions we are attempting to learn more about, and so have explored inputs to the gating.

Cortical feedback

One of the significant inputs is FB from Layer 6 in the level above. That is to say that the gating from Level (n) to (n+1), is modulated by FB from Layer 6 in Level (n+1).

Thalamic feedback and TRN

There is a substructure of the Thalamus called the Thalamic Reticular Nucleus (TRN) that receives cortical and thalamic excitatory input, and sends inhibitory inputs to the relay cells of the thalamus.

These gating cells also receive inhibitory input from other Thalamic cells, labelled interneurons. Thalamic interneurons receive input from the very same relay cells, layer 6 of the cortex and the brainstem.

These circuits between TRN, BRF and thalamus are complex. They are simplified in the figure below, which appears in Sherman 2006 (Scholarpedia on the Thalamus), a version of which is found in Sherman 2007.

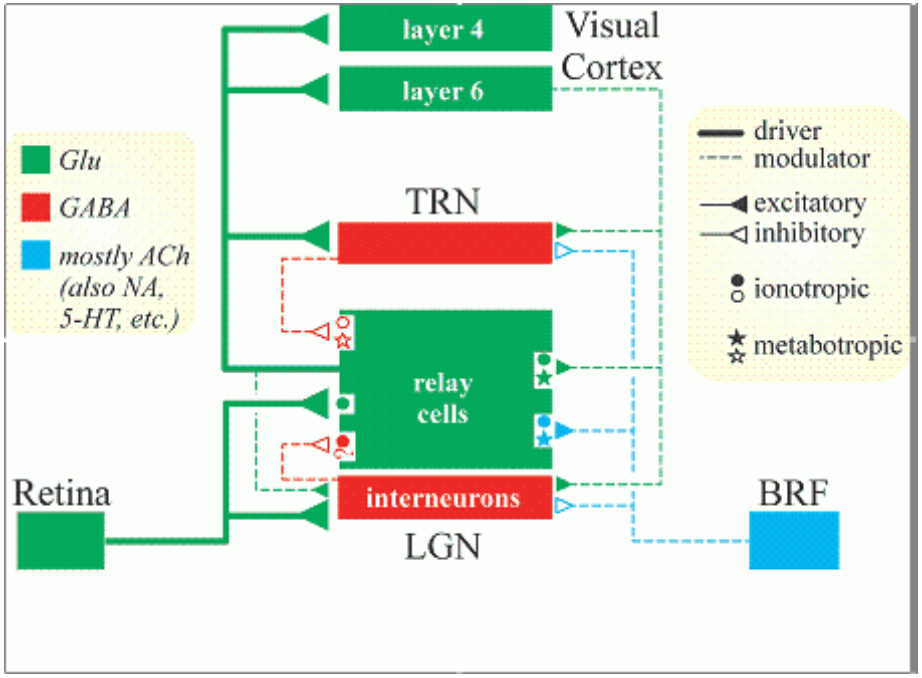


Figure 4. "Schematic diagram of circuitry for the lateral geniculate nucleus. The inputs to relay cells are shown along with the relevant neurotransmitters and postsynaptic receptors (ionotropic and metabotropic) Abbreviations: LGN, lateral geniculate nucleus; BRF, brainstem reticular formation; TRN, thalamic reticular nucleus." Caption and figure reproduced from Sherman 2006.

We are currently representing this complexity as a black box (as shown in the diagram) that receives input from the Thalamus, BRF and cortex, and inhibits the relay cells. The purpose and transfer function require analysis and exploration. It may be necessary to model the complexity explained above, or some simpler equivalent may provide the necessary functionality.

BRF

The BRF is the Brainstem Reticular Formation, which as the name suggests, is a part of the brainstem. It has a number of functions that could be very important for attention and general functioning of the cortex, and therefore, we have included it and it's connections to the Thalamus. Some of these functions include:

1. Somatic motor control
2. Cardiovascular control
3. Pain modulation
4. Sleep and consciousness
5. Habituation

The Wikipedia page for the BRF gives a very good summary.

Modulation Signal Characteristics

It is interesting to note that the firing mechanism for the BRF and Layer 6 modulation of the Thalamic relay is Burst Mode rather than the more common Tonic Mode. Tonic firing has a frequency that is proportional to the 'activation' of a neuron. The frequency can be interpreted as the "strength" of the signal. Some have interpreted it in the past as a probability or confidence value. For Burst Mode firing, after a 'silent' period, the initial firing pattern is a burst of activity. This "results in a very different message relayed to cortex, depending on the recent voltage history of the relay cell" (Sherman 2006). It is thought that this acts as a ‘wake up call’ to the cortex when there has been some external change. We plan to speculate and elaborate further on possible purposes of this in the future.

Timing Information

The CLA White Paper makes mention of timing information being fed back from the thalamus to layer 5 via layer 1. This has been added to the diagram for visibility. It is thought to be important for prediction of the next state at the appropriate time.

Other Factors

There are a number of other significant brain components that may substantially affect the operation of the neocortex. Based on the literature, the most significant of these is probably the Basal Ganglia, which forms circuits with the Thalamus and Cortex. Another interesting and possibly important component are Betz cells, which directly drive muscles from the cortex.

Conclusion

This post was a first attempt to create an enhanced diagram of cortical layers and thalamocortical connectivity in the context of MPF/HTM/CLA theory. We'll continue to elaborate on this in future posts.

Posted by Gideon Kowadlo at Tuesday, May 27, 2014 No comments :

Labels: Artificial General Intelligence , CLA , Cortical Learning Algorithm , Grossberg , Neocortex , Sherman , Thalamocortical , Thalamus

Monday, 28 April 2014

TP 1/3: Temporal Pooler background

by David Rawlinson and Gideon Kowadlo

Article Series

This is the first of 3 articles about temporal pooler design for Numenta's Cortical Learning Algorithm (CLA) and related methods (e.g. HTM/MPF). It has limited relevance to Deep Belief Networks.

This first part will describe some of the considerations, objectives and constraints on temporal pooler design. We will also introduce some useful terminology. Part 2 will examine Jeff Hawkins' new temporal pooler design. Part 3 will examine an alternative temporal pooler design we are successfully using in our experiments.

We will use the abbreviation TP for Temporal Pooler to save repeating the phrase.

Purpose of the Temporal Pooler

What is the purpose of the Temporal Pooler? The TP is a core process in construction of a hierarchy of increasingly abstract symbols from input data. In a general-purpose algorithm designed to accept any arbitrary stream of data, building symbolic representations, or associating a dictionary of existing symbols with raw data, is an exceedingly difficult problem. The MPF/CLA/HTM family of algorithms seeks to build a dictionary of symbols by looking for patterns in observed data, specifically:

a) inputs that occur together at the same time
b) sequences of inputs that reliably occur one after another

The Spatial Pooler's job is to find the first type of pattern: Inputs that occur together. The TP's job is to find the second type of pattern - inputs that occur in predictable sequences over time. MPF calls these patterns "invariances". Repeated experience allows discovery of invariances: reliable association in time and space binds inputs together as symbols.

MPF/CLA/HTM claims that *abstraction is equivalent to the accumulation of invariances*. For example, a symbol representing "dog" would be invariant to the pose, position, and time of experiencing the dog. This is an exceptionally powerful insight, because it opens the door to an algorithm for automatic symbol

definition.

What is a symbol? Symbols are the output of Classification. There must be consistent classification of the varying inputs that collectively represent a common concept, such as an entity or action. There must also be substitution of the input with a unique label for every possible classification outcome.

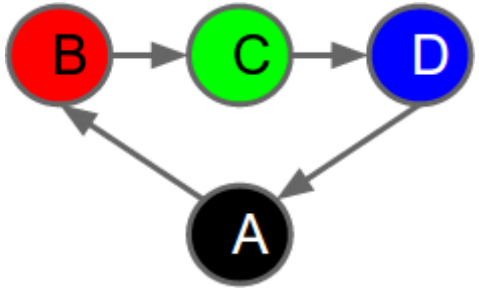
The symbol represents the plethora of experiences (i.e. input data) that cumulatively give the symbol its meaning. Embodiment of the symbol can be as simple as the firing of a single neuron. Symbols that represent a wider variety of input are more abstract; symbols that represent only a narrow set of input are more concrete.

Markov Chains

We will evaluate and compare TP functions using some example problems. We will be using Markov Chains to define the example problems. The Markov property is very simple; each state only depends on the state that preceded it. All information about the system is given by the **identity** of the **current** state. The current state alone is enough to determine the probability of transitioning to any other state.

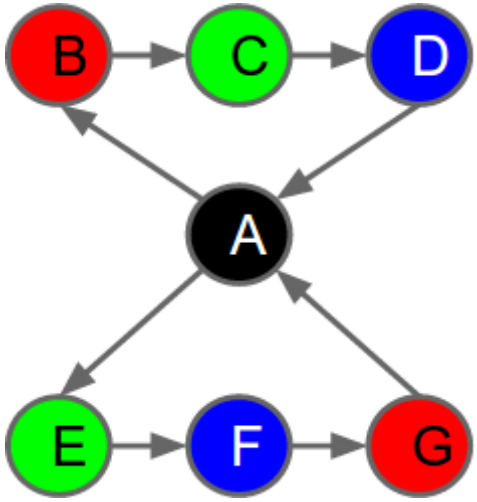
Markov chains are normally drawn as graphs, with states being represented as vertices (aka nodes, typically presented as circles or ellipses). Changes in system state are represented by transitions between vertices in the graph. Edges in the graph represent potential changes, usually annotated with transition probabilities; there may be more than one possible future state from the current state. When represented as a Markov chain, the history of a system is always a sequence of visited states without forks or joins.

Here is an example Markov Chain representing a cycle of 4 states (A,B,C,D). Each state has a colour. The colour of the current state is observable. This graph shows that Red comes after Black. Green comes after Red. Blue follows Green, and Black follows Blue:



A simple Markov Chain shown as a graph. Vertices (circles) represent states. There are 4 states. Edges represent possible transitions between states. The color of each circle indicates what is observed in the corresponding state.

Now let's look at a more complex example. What happens when one state can be followed by more than one subsequent state? The answer is that the next state is determined randomly according to the probability of each transition.



A Markov chain with 7 states. State A can be followed by B or E with equal probability. Both states D and G are always followed by state A. This system represents 2 sequences of colours, separated by black. Each sequence is equally likely to occur, but once the sequence starts it is always completed in full.

In our examples, when there is a fork, all subsequent states are equally likely. This means that in the example above, both states B and E are equally likely to follow state A.

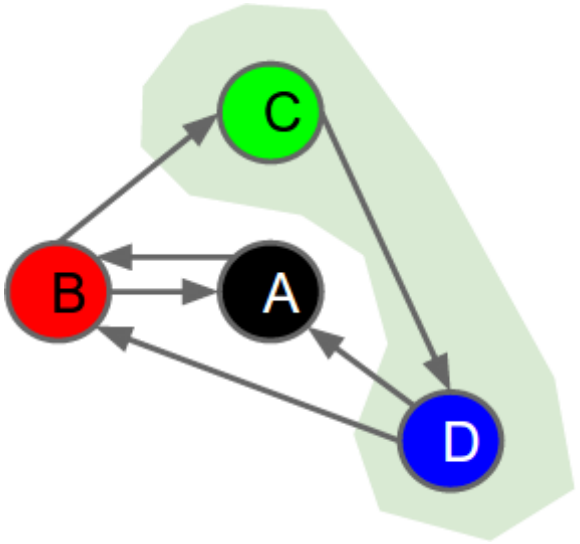
First-Order Sequence Memory Representation

If we allow an MPF hierarchy to observe the system described by the Markov Chain above, the hierarchy will construct a model of it. Spatial pooling tries to identify instantaneous patterns (which will cause it to learn the set of observed colours). Temporal pooling attempts to find sequential patterns (frequently-observed sequences of colours). More simply put, an MPF hierarchy will try to learn to predict the sequences of colours.

How accurately the hierarchy can predict colours will depend on the internal model it produces. We know that the "real" system has 2 sequences of colours (RGB and GBR). We know that there is only uncertainty when the current state is Black.

However, lets assume the MPF hierarchy consists of a single region. Let's say the region has 4 cells, that learn to fire uniquely on observation of each of the 4 colours. The Sequence Memory in the region learns the order of cell firing - i.e. it predicts which cell will fire next, given the current cell. It only uses the current active cell to predict the next active cell.

The situation we have described above can be drawn as a Markov Chain like this:



A Markov chain constructed from first-order prediction given observations from the 7-state system shown in the previous figure.

Note that the modelled system is less predictable than the "real world" system we were observing. We can still be sure that Blue always follows Green. But when we look at which colours follow Blue, we can only say that **either** Black or Red will follow blue. Similarly, we cannot predict whether Red will be followed by Green or Black, whereas in the "real" world this is predictable.

The reason for the lack of predictive ability is that we are only using the **current** colour of our model to predict the next colour. This is known as first-order prediction (i.e. Markov order=1). If we used a longer history of observations, we could predict correctly in every case except Black, where we could be right half the time. Using a "longer" history to predict is known as variable-order prediction (variable because we don't know, or limit, how much history is needed).

Uncertainty and the Newtonian experience

In today's physics, non-determinism (future not decided) or at least in-determinism (inability to predict) are widely and popularly accepted. But although these effects may dominate on very small and large scales, for much of human-scale experience the physical world is essentially a predictable, Newtonian system. Indeed, human perception encodes Newtonian laws so exactly that acceleration of objects induces an expectation of animate agency.

In a Newtonian world, every action is "explained" or caused by some other physical event; it is simply necessary to have a sufficiently comprehensive understanding & measurement to be able to discover the causes of all observed events.

This intuition is important because it motivates the construction of an arbitrarily large hierarchy of symbols and their relations in time and space, with the expectation that ***somewhere*** in that hierarchy all events can be understood. It doesn't matter that some events cannot be explained; we just need to get

enough practical value to motivate construction of the hierarchy. The Newtonian nature of our experiences at human scale means that most external events are comprehensible and predictable.

Finding Higher-Order Patterns in Markov Chains

The use of longer sequences of colours to better explain (predict) the future shows that confusing, unpredictable systems may become predictable when more complex causes are understood. Longer histories (higher-order representations) can reveal more complex sequences that **are** predictable, even when the simpler parts of these patterns are **not** predictable by themselves.

The "Newtonian world" assumption gives us good reason to expect that a lot of physical causes are potentially predictable, given a suitably complex model and sufficient data. Even human causes are often predictable. It is believed that people develop internal models of third party behaviour (e.g. "theory of mind"), which may help with prediction. This evidence motivates us to try to discover and model these causes as part of an artificial general intelligence algorithm.

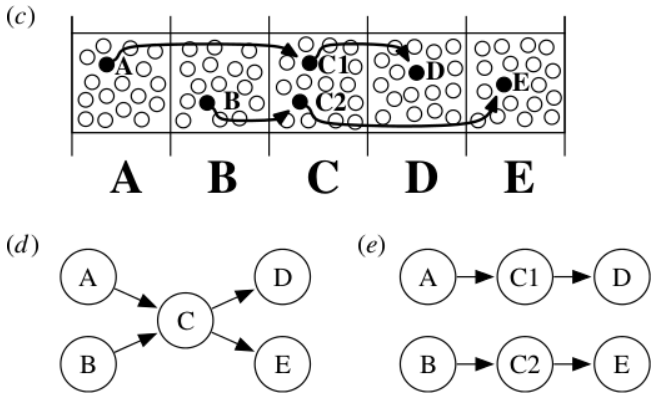
Therefore, one purpose of the MPF hierarchy is to construct higher-order representations of an observed world, in hope of being able to find predictable patterns that explain as many events as possible. Given this knowledge, an agent can use the hierarchy to make good, informed decisions.

Constructing Variable-Order Sequences using First-Order Predictions

There is one final concept to introduce before we can discuss some temporal pooler implementations. The final concept is ways of representing *variable-order* sequences of cell activation in a Sequence Memory. This is not trivial because the sequences can be of unlimited length and complexity, (depending on the data). However, for practical reasons, the resources used must be limited in some way. So, how can arbitrarily complex structures be represented using a structure of limited complexity?

Let's define a sequence memory as a set of cells, where each cell represents a particular state. Let's specify a fixed quantity of cells, and to encode **all** first-order transitions between these cells. All such pairwise transitions can be encoded in a matrix of dimension cells x cells. This means that cells only trigger each other individually; only pairs of cells can participate in each sequence fragment.

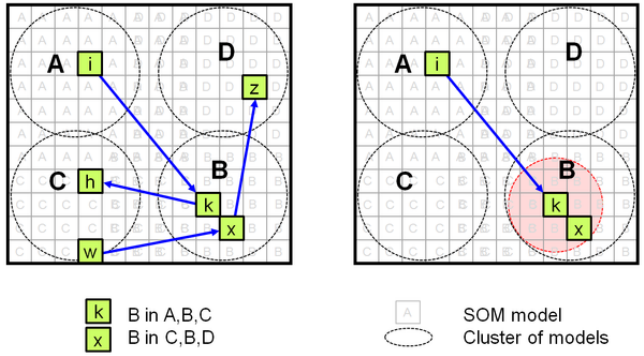
So, how can longer sequences be represented in the matrix? How can we differentiate between occurrences of the same observation in the context of different sequences?



The "state splitting" method of creating a variable-order memory using only first-order relationships between cells. This is part of a figure from Hawkins, George and Niemasik's "Sequence memory for prediction, inference and behaviour" paper. In this example there are 5 states A,B,C,D,E. We observe the sequences A,C,D and B,C,E. In subfigure (c), we see that for each state A,...,E there are a bunch of cells that respond to it. However, each cell only responds to specific *instances* of these states. Specifically, there are two cells (C1 and C2) that respond to state C. C1 responds to state C only after state A. C2 responds to an observation of state C only after state B. If we have only a single cell that responds to C, we lose the ability to predict D and E (see subfigure (d)). With two cells uniquely responding to specific instances of C (see subfigure (e)), we gain the ability to predict states D and E. Prediction is improved by splitting state C, giving us a variable-order memory.

An elegant answer is to have multiple cells that represent the same observation, but which only fire in unique sequences. This concept is nicely explained in Hawkins, George and Niemasik's "Sequence memory for prediction, inference and behaviour" paper. They call it "state-splitting", i.e. splitting cells that represent an observed state and having multiple cells each responding to specific instances of the state in different sequences.

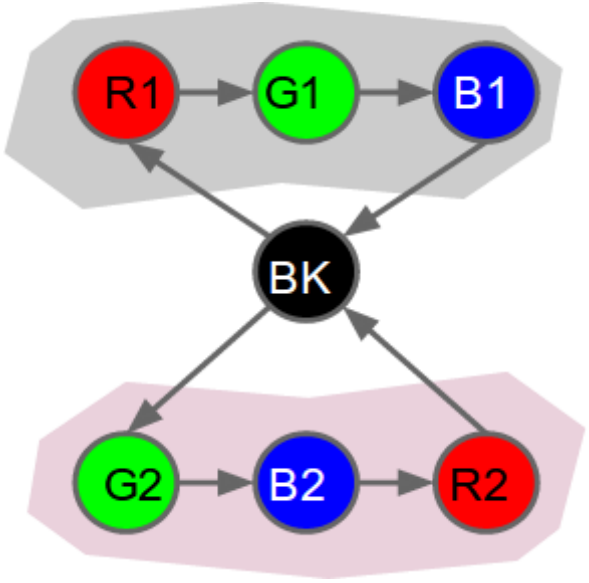
In the current CLA implementation, the same feature is achieved by having "columns" of cells that all respond to the same input in different sequence contexts (i.e. given a different set of prior cells). CLA says they share the same "proximal dendrite", which defines the set of input bits that activate the cell. In our paper, we showed how a radial inhibition function could induce sequence-based specialization of Self-Organising Map (SOM) cells into a variable-order sequence memory:



Creation of a variable order sequence memory in a Self-Organising Map (SOM) using inhibition. The circles represent cells that respond to the same input, in this case the letters A,B,C or D. We can use first-order sequence relationships to cause specialization of cells to specific instances of each letter. Blue lines represent strong first-order sequence relationships. The edge i-->k promotes k responding to "B" and inhibits x. Eventually k only responds to AB and x only responds to CB.

In all cases, the common element is having multiple cells that respond to the same input, **but only after specific sequences of prior inputs**.

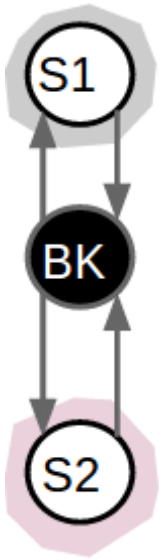
So, returning to our example problem above with 2 sequences of colours, RGB and GBR, what is the ideal sequence-memory representation using a finite set of cells, multiple cells for each input depending on sequence context, and only first-order relationships between cells? One good solution is shown below:



Example modelling of our test problem using a variable order sequence memory. We have 7 cells in total. Each cell responds to a single colour. Only one cell responds to black (BK). Having two cells responding to each colour R,G and B allows accurate prediction of all transitions, except where there is genuine uncertainty (i.e. edges originating at black). The temporal pooler component should then be able to identify the two sequences (grey and pink shading) by learning these predictable sequences of cell activations. The temporal pooler will replace each sequence with a single "label", which might be a cell that fires continuously for the duration of each sequence. Cells watching the temporal pooler output will notice fewer state changes, i.e. a more stable output.

Let's assume this graph of cells representing specific occurrences of each input colour (i.e. a Sequence Memory) provides the input to the Temporal Pooler. What is the ideal Temporal Pooler output?

Well, we know that there are in fact 2 sequences, and a "decision" state that switches between them. The ideal sequence-memory and temporal pooler implementation would track all predictable state changes, and replace these sequences with labels that persist for the duration of each sequence. In this way, the problem is simplified; other cells watching the temporal pooler output would observe fewer state changes - only switching between Black, Sequence #1 and Sequence #2.



Markov Chain that is experienced by an observer of the output from the ideal sequence-memory and temporal pooler modelling shown in the figure above. The problem has been reduced from 7 states to 3. State transitions are only observed when transitioning to or from the Black state (BK). Otherwise, a constant state is observed.

How can the ideal result be achieved? The next article will discuss how CLA simplifies sequences using the concepts described in this article, and the final article will discuss some alternative methods that we propose.

Posted by Dave Rawlinson at Monday, April 28, 2014 No comments :

Labels: CLA , Cortical Learning Algorithm , first order , HTM , invariances , markov chain , symbol grounding problem , Temporal Pooling , variable order

Tuesday, 22 April 2014

Architecture of the Memory Prediction Framework / Cortical Learning Algorithm / Hierarchical Temporal Memory

by David Rawlinson and Gideon Kowadlo

Introduction

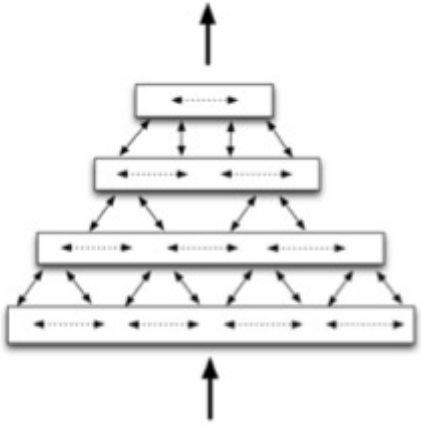
The Memory Prediction Framework (MPF) is a general description of a class of algorithms. Numenta's Cortical Learning Algorithm (CLA) is a specific instance of the framework. Numenta's Hierarchical Temporal Memory (HTM) was an earlier instance of the framework. HTM and CLA adopt different internal representations so it is not as simple as CLA supersedes HTM.

This post will describe structure of the framework that is common to MPF, CLA and HTM, specifically some features that cause confusion to many readers.

For a good introduction to MPF/CLA/HTM see the Numenta CLA white paper.

The Hierarchy

The framework is composed as a hierarchy of identical processing units. The units are known as "regions" in CLA. The hierarchy is a tree-like structure of regions:

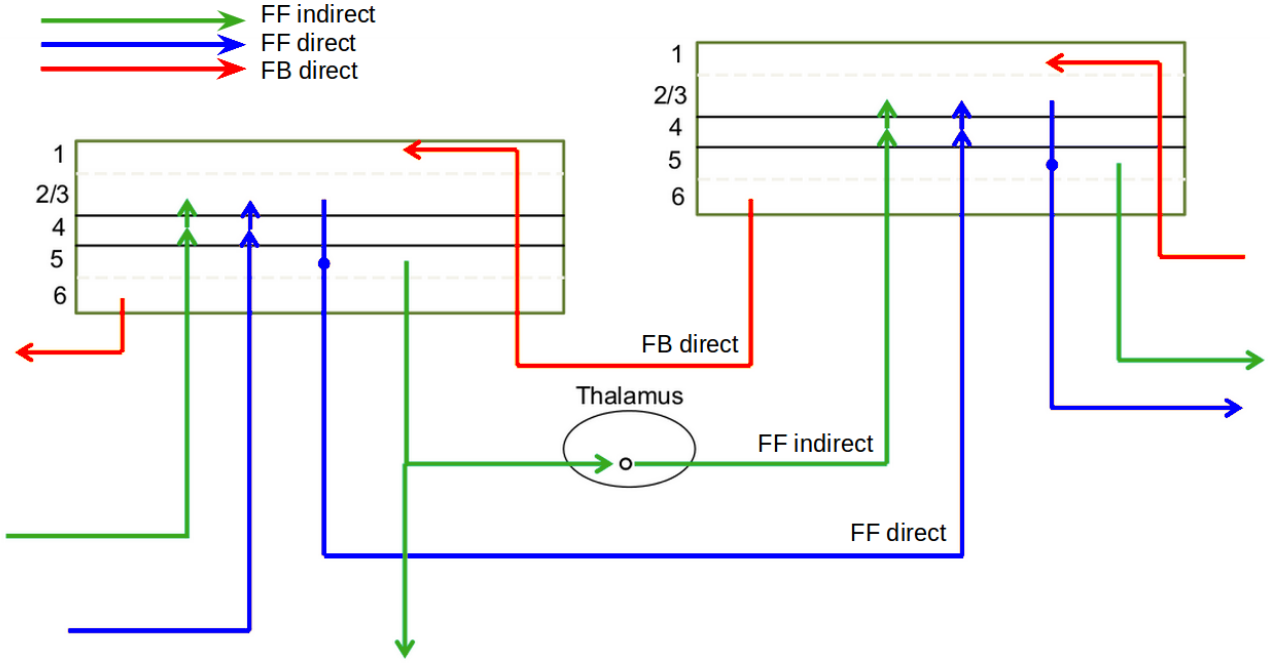


MPF/CLA/HTM hierarchy of Regions. The large arrows show the direction of increasing abstraction. Smaller arrows show the flow of data between nearby regions in a single level of the hierarchy, and between levels of the hierarchy. Figure originally from Numenta.

Regions communicate with other, nearby regions in the same level of the hierarchy. Regions also communicate with a few regions in a higher level of the hierarchy, and a few regions in a lower level of the hierarchy. Notionally, abstraction increases as you move towards higher levels in the hierarchy. Note that Hawkins and Blakeslee define abstraction as "the accumulation of invariances".

Regions

Biologically, each Region is a tiny patch of cortex. The hierarchy is constructed from lots of patches of cortex. Each piece of cortex has approximately 6 layers (there are small variations throughout the cortex, and the exact division between cortical layers in biology is a bit vague. Nature hates straight lines). Note that in addition to having only 6 layers, each cortical region is finite in extent within the cortex - i.e. it is only a tiny area on the surface of the cortex.



Cortical **layers** and connections between hierarchy **levels**. Each cortical region has about 6 structurally (i.e. also functionally) distinct ***layers***. The hierarchy is composed of a tree of cortical regions, with connections between regions in different ***levels*** of the hierarchy. 3 key pathways are illustrated here. Each pathway is a carrier of distinct information content. The Feed-Forward pathways carry information UP the hierarchy levels towards increasing abstraction/invariance. The Feed-Back pathway carries information DOWN through hierarchy levels towards regions that represent more concrete, raw, unclassified inputs. Some pathways connect cortical regions directly, others indirectly (via other brain structures). Note that this image is a modified copy of one from Numenta, with additional labels and colours standardised to match the rest of this document.

Levels and Layers

Newcomers to MPF/CLA/HTM theory sometimes confuse "cortical layers" and connections between regions placed in different "levels" of the hierarchy. We recommend everyone uses layers to talk about cortical layers and levels to talk about hierarchy levels, although the levels and layers are somewhat synonymous in English. I believe this confusion arises because readers expect to learn one new concept at a time, but in fact levels and layers are two separate things.

Pathways

There are several distinct routes that information takes through the hierarchy. Each route is called a "pathway". What is a pathway? In short, a pathway is a set of assumptions that allows us to make some broad statements about what components are connected, and how. We assume that the content of data in each pathway is qualitatively different. We also assume there is limited mixing of data between pathways, except where some function is performed to specifically combine the data.

Directions

There are two directions that have meaning within the MPF/CLA/HTM literature. These are feed-forward and feed-back. Feed-Forward (FF) means data travelling **UP** between hierarchy levels, towards increasing abstraction. Feed-Back (FB) means data travelling **DOWN** between hierarchy levels, with reducing abstraction and taking on more concrete forms closer to raw inputs.

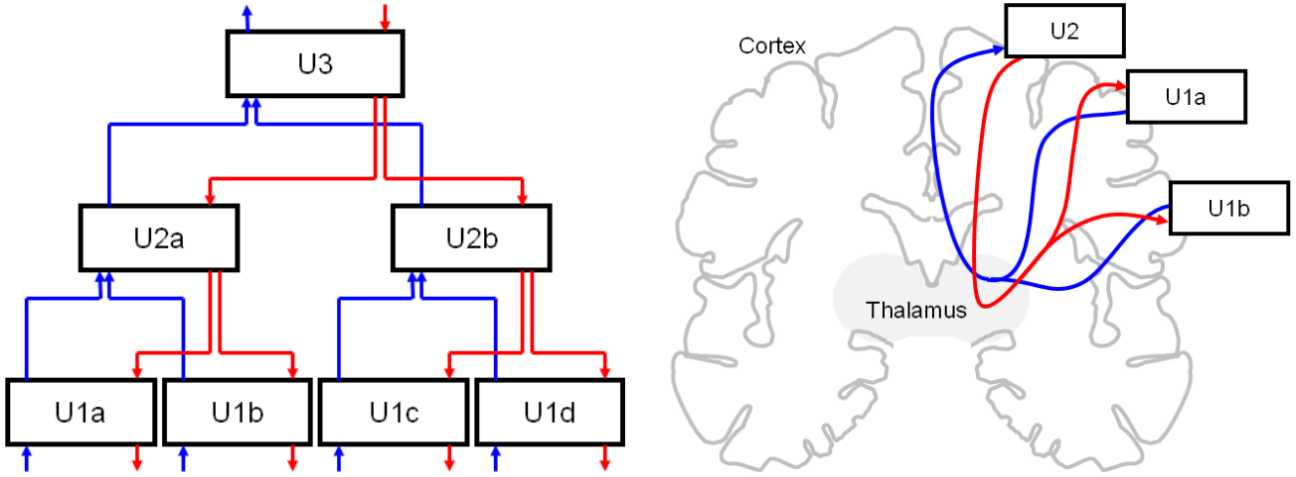
3 Pathways

The 3 pathways typically discussed in the MPF/CLA/HTM literature are:
- FF direct (BLUE)
- FF indirect (GREEN)
- FB direct (RED)

Direct means that data travels from one cortical region to another, without a stop along the way at an intermediate brain structure. Indirect means that the data is passed through another brain structure en-route, and possibly modified or gated (filtered).

This does not mean that other pathways do not exist. There is likely a FB-indirect pathway from Cortex to Cortex via the Basal Ganglia, and direct connections between nearby Regions at the same level in the hierarchy. However, current canonical MPF/CLA theory does not assign roles to these pathways.

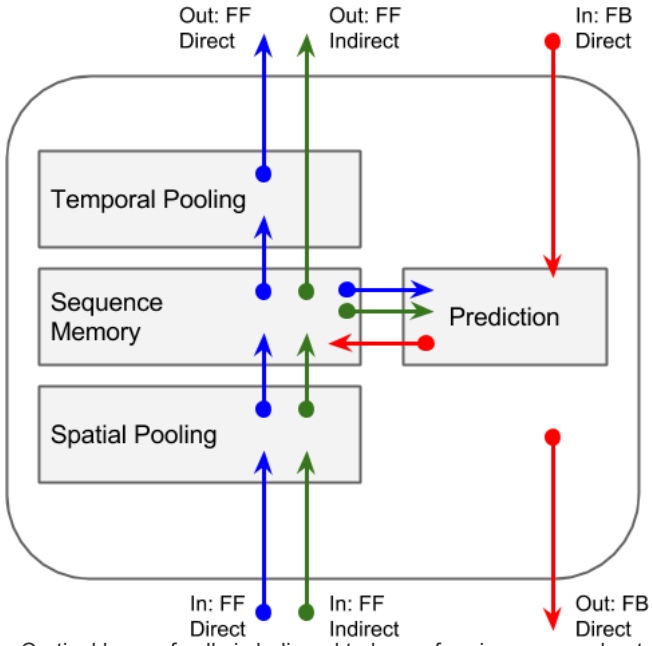
We will always use the same colours for these pathways.



The conceptual and biological arrangement of the MPF/CLA/HTM hierarchy. Left, the conceptual structure. Right, the physical arrangement of the hierarchy. Cortical processing occurs on the surface of the cerebrum, not inside it; the filling is mainly neuron axons connecting surface regions. FF (blue) and FB (red) pathways are shown. Moving between hierarchy levels involves routing data between different patches of cortex (surface). The processing Units - each, a separate region - here are labelled U*nc* where *n* is the hierarchy level and *r* is an identifier for each Region. Note that data from multiple regions is combined in higher levels of the hierarchy: For example, U2a receives FF data from U1a and U1b. Via the FB pathway, lower levels are able to exploit data from other subtrees. Some types of data relayed between hierarchy regions are relayed via deep brain structures, such as the Thalamus. We say these are "indirect" connections. The relays may modify / filter / gate the data en-route.

Conceptual Region Architecture

MPF/CLA/HTM broadly outlines the architecture of each Region as follows. Each region has a handful of distinct functional components, namely: Spatial Pooler, Sequence Memory, and Temporal Pooler. Prediction is also a core feature of each Region, though it may not be considered a separate component. I believe that Hawkins would not consider this to be a complete list, as the CLA algorithm is still being developed and does not yet cover all cortical functions. Note that the conceptual entities described here do not imply structural boundaries or say anything about how this might look as a neural network.



Key functional components of each Region. Note that ***every*** cellular Cortical layer of cells is believed to be performing some subset of these functions. It is not intended that each layer perform ***one*** of the functions. Where specifically described, the inputs and outputs of each pathway are shown. The CLA white paper does not specifically define how FB output is generated. It is possible that FB output contains predicted cells. Prediction is an integral function of the posited sequence memory cells, so whether it can be a separate component is debatable. However, conceptually, a sequence memory cell cannot be activated by prediction alone; FF ("bottom up") input is always needed to activate a cell. Prediction puts sequence memory cells into a receptive state for future activation by FF input. Regions receive additional data (e.g. from regions at higher hierarchy levels) when making their predictions. Prediction allows regions to better recognise FF input and predict future sequence cell activation. Note, from the existing CLA white paper it is not clear whether the FF indirect pathway involves Temporal Pooling. The white paper says that FF-indirect output originates in Layer 5 which is not fully described.

The **Spatial Pooler** identifies common patterns in the FF direct input and replaces them with activation of a single cell (or, variable, or state, or label, depending on your preferred terminology). The spatial pooler is functioning as an unsupervised classifier to transform input patterns into abstract labels that represent specific patterns.

The **Sequence Memory** models changes in the state of the spatial pooler over time. In other words, which cells or states follow which other cells/states? The Sequence Memory can be thought of as a Markov Chain of the states defined by the spatial pooler. Sequence Memory encodes information that enables predictions of future spatial pooler state.

The FF direct pathway cannot be driven by feedback from higher levels alone: FF input is always needed to fully activate cells in the Sequence Memory. As a hierarchy of unsupervised classifiers, the FF pathways are similar to the Deep Learning hierarchy.

Prediction is specifically a process of activating Sequence Memory cells that represent FF input patterns that are likely to occur in the near future. Prediction

changes Sequence Memory cells to a receptive state where they are more easily activated by future FF input. In this way, prediction makes classification of FF input more accurate. Improvement is due to the extra information provided by prediction, using both the history of Sequence Cell activation within the region and the history of activation of Sequence Memory cells within *higher* regions, the latter via the FB pathway.

It is probable that the FB pathway contains prediction data, possibly in addition to Sequence Memory cell state. This is described in MPF/HTM literature, but is not specifically encoded in existing CLA documentation.

Personally, I believe that prediction is synonymous with the generation of behaviour and that it has dual purposes; firstly, to enable regions to better understand future FF input, and secondly, to produce useful actions. A future article will discuss the topic of whether prediction and planning actions could be the same thing in the brain's internal representation. An indirect FB pathway is not shown in this diagram because it is not described in MPF/CLA literature.

While Spatial Pooling tries to replace instantaneous input patterns with labels, **Temporal pooling** attempts to simplify ***changes over time*** by replacing common sequences with labels. This is a function not explicitly handled in Deep Learning methods, which are typically applied to static data. MPF/CLA/HTM is explicitly designed to handle a continuous stream of varying input.

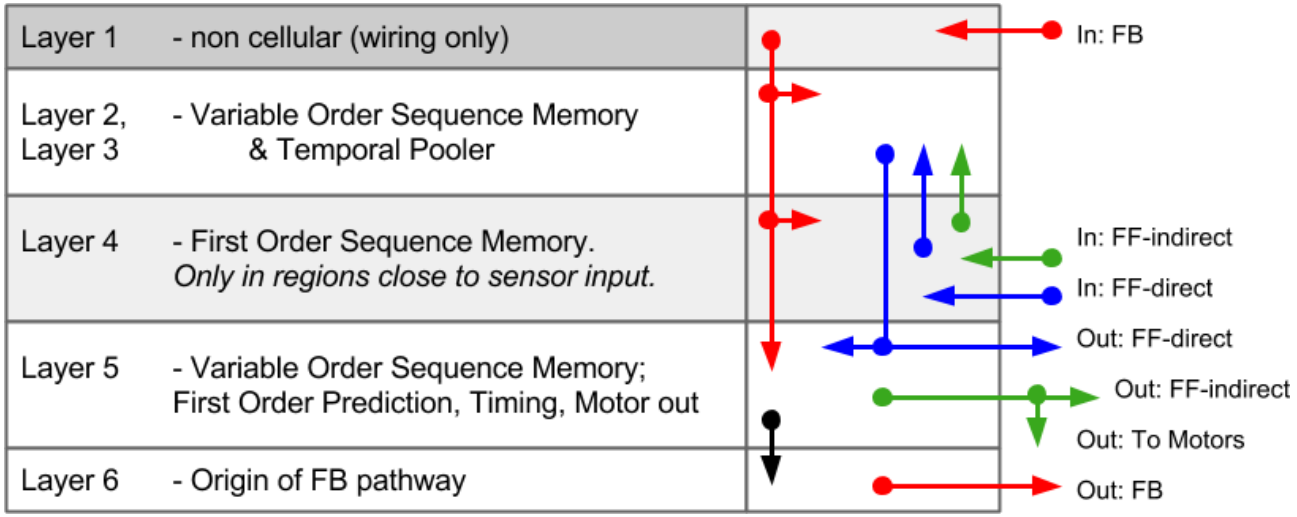
Temporal pooling ensures that regions at higher levels in the hierarchy encode longer sequences of patterns, allowing the hierarchy to recognise long-term causes and effects. The input data for every region is different, ensuring that each region produces unique representations of different sub-problems. Spatial and Temporal pooling, plus the merging of multiple lower regions in a tree-like structure, all contribute to the uniqueness of each region's Sequence Memory representation.

Numenta also claim that there is a timing function in cortical prediction, that enables the region to know *when* specific cells will be driven active by FF input. Since this function is speculative, it is not shown in the diagram above. The timing function is reportedly due to cortical layer 5.

Mapping Region Architecture to Cortical Layers

As it stands CLA claims to explain (most of) cortex layers 2, 3 and 4. Hawkins et al are more cautious about their understanding of other cortical layers.

To try to present a clear picture of their stance, I have included a graphic (below) showing the functions of each biological cortex layer as defined by CLA. The graphic also shows the flows of data both between layers and between regions. Note that the flows given here are **only** those as described in the CLA white paper and Hawkins' new ideas on temporal pooling. Other sources do describe additional/alternative connections between cortical levels and regions. The exact interactions of each layer of neurons are somewhat messy and difficult to interpret.



Data flow between cortical layers as described in the CLA white paper. Every arrow in this diagram is the result of a specific comment or diagram in the white paper. This figure is mostly a repeat of the same information as in the second figure, using a different presentation format. I have speculatively coloured each arrow by content (i.e. pathway) , but don't rely on this interpretation. Inputs to L2/3,L4 and L5 from L1 are red because there are no cells in L1 to transform the FB input signal, therefore this must be FB data. The black arrow is black because I have no idea what data or pathway it is associated with!

Summary

I hope this review of the terminology and architecture is helpful. Although the MPF/CLA/HTM framework is thoroughly and consistently documented, some of the details and concepts can be hard to picture, especially in the first encounter. The CLA White Paper does a good job of explaining Sparse Distributed Representations and spatial, temporal pooler implementations as biologically-inspired Sequence Memory cells. However, the grosser features of the posited hierarchy are not so thoroughly described.

It is worth noting that according to recent discussions on the NUPIC mailing list, the current NUPIC implementation of CLA does not correctly support multi-level hierarchies correctly. This problem is expected to be addressed in 2014, permitting multi-level hierarchies.

Posted by Dave Rawlinson at Tuesday, April 22, 2014 3 comments :

Labels: CLA , Cortical Learning Algorithm , Hierarchical Generative Models , HTM , MPF , Neocortex , Sequence Memory , Spatial Pooling , Temporal Pooling

Home Older Posts

Subscribe to: Posts (Atom)

Recent activity

Tweets by @ProjectAGI

Translate

Select Language

Powered by Google Translate

Blog Archive

Labels

- AlphaGo
- arcade
- Architecture
- Artificial General Intelligence
- atari
- awareness
- Baar
- choice
- CLA
- columns
- Competitive Learning
- Computational
- computer games
- Connectome
- Consciousness
- cortex
- Cortical Learning Algorithm
- deep belief networks
- deep convolutional networks
- Deep Learning
- DeepMind
- demo
- dendrite
- directed acyclic graph
- Emotion
- Eric Laukien
- Experimental Framework
- Felix Andrews
- Fergal Byrne

- first order
- free will
- Friston
- frontal cortex
- Generalized LSTM
- Generative Models
- Global Workspace
- Go
- Graves
- Grossberg
- Hierarchical Generative Models
- hierarchy
- HQSOM
- HTM
- ICML
- Intuition
- invariances
- Jeff Hawkins
- Lee & Mumford
- Long Short Term Memory
- LSTM
- machine learning
- markov chain
- Memes
- Memory-Prediction Framework
- Michael Ferrier
- Microsoft
- missing data
- MNIST

- Model
- Monner
- morphology
- MPF
- Natural Selection
- Neocortex
- neurobiology
- Numenta
- NUPIC
- paper
- Predictive Coding
- pyramidal cell
- quantum computing
- Rao & Ballard
- Recurrent Neural Networks
- Reinforcement Learning
- Research
- retina
- review
- Rinkus
- Risks
- Ryan McCall
- self determination
- Sequence Memory
- sequence unfolding
- Sherman
- Singularity
- Software
- sparse coding

- Sparse Distributed Representations
- Spatial Pooling
- stationary problem
- symbol grounding problem
- Temporal Pooling
- Thalamocortical
- Thalamus
- unsupervised learning
- variable order
- visual object recognition