Dynamic Thresholds for Self-Organizing Predictive Cells

Benjamin Cowley[1], John Thornton[1,2], Linda Main[1] and Abdul Sattar[1]

[1]Institute for Integrated and Intelligent Systems, School of ICT, Griffith University, QLD 4111, Australia
[2]Department of Informatics, University of Sussex, Brighton BN1 9RH, UK
benjamin.cowley@griffithuni.edu.au

Abstract

It has become increasingly popular to view the brain as a prediction machine. This view has informed a number of theories of brain function, the most prominent being predictive processing, where generative hypotheses are iteratively updated by error signals. In this treatment we take a lower level approach by examining the hierarchical temporal memory framework, which views individual pyramidal cells as the primary predictive unit of a self-organizing networked sequence learning system. Within this computational framework, the cell behaviour is constrained by a number of parameters which are static and shared across all cells. To further increase the adaptability of the cells, we shift away from this paradigm by introducing the concept of dynamic thresholds. This allows for the activation threshold (the amount of activity on a distal dendrite needed to form a prediction) to be adjusted continuously and individually for each cell. As a metric we use the prior, or unconditional, probability of activity on the proximal dendrites. Our experiments show that using this metric for dynamic thresholds can improve the predictive capabilities of the system in a number of domains, including anomaly detection, where we achieve state-of-the-results on the Numenta Anomaly Benchmark.

Introduction

In recent years, the search for a unified theory of brain function has begun to converge on the idea that the brain is a prediction machine. At a high-level this is expressed in the Bayesian brain theory which envisions the brain as constantly performing probabilistic inference in an attempt to explain stimuli and trigger optimal reactions to it (Knill and Pouget, 2004). An "intermediate-level" model (Clark, 2015) is the increasingly popular theory of predictive processing (PP), which is also often referred to as hierarchical predictive coding. Under this model, top-down and lateral connections between neurons communicate hypotheses (predictions) regarding the current and future state of the system, while bottom-up connections communicate prediction errors which update the hypothesis of a higher layer (Rao and Ballard, 1999). PP can also be subsumed under the free-energy principle, which proposes that the behaviour of natural adaptive systems can be modelled by applying a policy that reduces the free-energy of their internal states (Friston et al., 2006, 2009). In this treatment we provide a lower level approach by examining prediction forming in a biologically plausible self-organizing system of cells.

The existing treatments of PP are largely theoretical and it is unclear how such models could be implemented in a neuronal architecture that is plausibly similar to actual biology. Some work in this area (Friston et al., 2006) makes useful suggestions concerning the functioning of candidate feedback and error-unit neurons, but does not integrate this account into a theory of both high-level and low-level neural architecture in specific regions of the central nervous system. Such a region of specific interest is the neocortex, which is implicated in perception, motor action, and high-level thought. A theory needs to incorporate what is already known concerning the architecture of the neocortex, such as its sparse encoding scheme (Olshausen and Field, 2004), its lateral organisation into layers, and its division into functional mini-columns (Mountcastle, 1997). Such a theory should also explain the behaviour of cells, which are known to form a large number of synapses by applying a Hebbian learning method (Hebb, 1949), and to produce complex non-linear responses to activity on these synapses (Häusser et al., 2000; Major et al., 2013).

One such theory that attempts to account for these factors, and which exhibits significant parallels with PP, is hierarchical temporal memory (HTM) (Hawkins and Blakeslee, 2007). Here, the function of the neocortex is again understood in terms of generating predictions of future states. These predictions are generated by self-organizing pyramidal cells that apply Hebbian learning to predict sequences of mini-column activity, both using and explaining the columnar structure of the neocortex. In effect, each column is a node in a hierarchically structured sequence learning algorithm, organised in such a way that each node can be incorporated into multiple learned sequences, depending on the context of operation. HTM theory posits that the *distal dendrites* of pyramidal cells act to recognise individual patterns, and learn the individual *causes* of activity on *proximal dendrites*. Essentially, activity on a pyramidal cell's distal dendrites predicts activity on its proximal dendrites,

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

and networks of such pyramidal cells can learn and predict sequences (Hawkins and Ahmad, 2016).

An algorithm that implements the sequence learning procedure of HTM is called the HTM *temporal pooler* (TP), an online learning connectionist system (Hawkins et al., 2010). TP comprises a number of self-organizing cells (artificial neurons) which possess distal dendrite *segments* that form synapses with other cells according to plausible Hebbian learning principles. The activity of proximal dendrites is governed by a separate learning system, the *spatial pooler* (SP). When the activity on a distal dendrite segment is greater than an *activation threshold*, the cell is considered to be predicting activity on its proximal dendrites and enters a predictive (depolarized) state. The overall system has already proved itself highly effective for temporal sequence prediction (Cui et al., 2016) and on-line anomaly detection (Lavin and Ahmad, 2015), while also providing a computational explanation for the density of connectivity found in actual neocortex (Hawkins and Ahmad, 2016).

Each cell's behaviour in the TP is constrained by a number of pre-set parameters which remain static throughout the lifetime of the system and are identical across all cells. Here, we shift away from this paradigm and propose that, as well as its connectivity, the internal dynamics of a cell should adapt to the cell's environment. We experiment with this idea by dynamically adjusting the threshold of synaptic activity required for a cell to enter a predictive state. The metric we use is the cell's prior, or unconditional, probability of becoming active. The basis for this metric is simple and lies at the heart of Bayesian inference: the higher the prior probability, the higher the posterior probability should be.

We refer to our modified algorithm as temporal pooler with dynamic thresholds (TPDT) and compare it to the standard TP on two sequence prediction tasks on which the TP has previously produced strong results (Cui et al., 2016), and show that TPDT can produce improved performance on these tasks. We further apply TPDT to the Numenta Anomaly Benchmark (NAB) (Ahmad and Hawkins, 2015), where we achieve state-of-the-art results. Our application of dynamic thresholds to TP, then, has produced two major contributions: 1) we have shown that incorporating the prior probability of a cell's activation into the predictive mechanisms can improve the performance of TP; 2) introduced a new spur of HTM research in which parameters are dynamic and adaptive.

Previous work

TP and a sparse encoder, the spatial pooler (SP), both form part of the HTM framework, which was first outlined in the Cortical Learning Algorithms (CLA) white paper by Numenta (Hawkins et al., 2010). TP and, more generally, the HTM framework has been shown to perform strongly on sequence prediction tasks, where it produces results comparable or better than leading sequence learning algorithms, such as TDNN and LSTM (Cui et al., 2016). The HTM framework has also been applied to the task of online anomaly detection, it has produced strong results in this field and is state-of-the-art on NAB (Lavin and Ahmad, 2015).

Other treatments of the HTM computational framework have looked at both formalizing (Mnatzaganian et al., 2016), and expanding SP (Thornton and Srbic, 2013). As well as investigating the best method for communicating between hierarchical regions (Kneller and Thornton, 2015; Skrynnik et al., 2016). There has also been work on implementing a PP style hierarchy using CLA (McCall and Franklin, 2013).

Temporal pooler with dynamic thresholds

HTM models pyramidal neurons as being the primary encoding and predicting unit within the neocortex. The proximal dendrites learn and encode bottom-up input, distal dendrites learn causes of and forms predictions on the bottom-up input, and the apical dendrite learns and encodes top-down predictions. Within the computational framework, the SP governs the proximal dendrites, while the TP governs the distal dendrites. In this treatment we implement dynamic thresholds to TP, and will focus our discussion on TP and the modified algorithm (TPDT). We apply this system as a single layer, non-hierarchical system, so are not concerned with the apical dendrites. We begin this section with a discussion of the overarching theories of TP and TPDT, followed by a description of the implementation of these theories.

Temporal pooler

The TP is a connectionist system in which the individual artificial neurons form temporal predictions on their future state; when the system is taken in its totality, it forms predictions on sparsely encoded sequences. In contrast to more typical artificial neural networks, where the states and output of artificial neurons are represented by real numbers, the activity of a TP system is represented by discrete states of the various structures. This paradigm allows us to investigate biologically inspired encoding from the perspective of dynamically shifting states, where action potentials are discrete events, and, more practically, it allows for a number of optimization techniques in implementation.

In TP the pyramidal cells are referred to simply as *cells*, and are grouped into *columns* (based on minicolumns of the neocortex). For simplicity the activity on the proximal dendrites is the same for all cells in a single column. As such, we refer to the proximal dendrite activity as column activity, and the cell's role is to predict the future activity of the column. The use of multiple cells in a column allows for the representation of different causes for the column's activation. For example, a column which encodes an edge appearing at a specific spot in the visual field could have been predicted to appear there through multiple causes, such as moving left to right, right to left, or being stationary. With different causes learned by different cells, the system is then

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

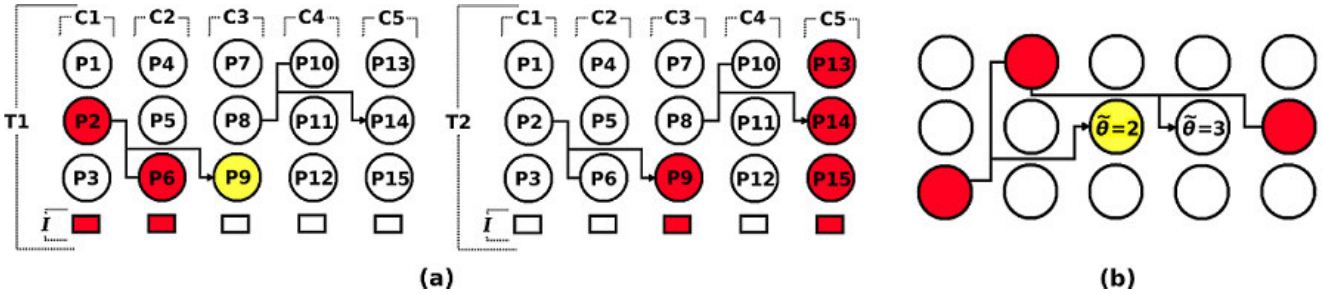

Figure 1: **a)** Diagram of TP activity over two time-steps (T1,T2). This simple TP has 5 columns (C1-C5) each with three pyramidal cells (P1-P15). Columns are activated by the input (I). Synapses and segment structures are shown by a single solid line, where the arrowhead points to the cell that the segment belongs to. In T1, P9 is in a predictive state (yellow) due to its connections with P2 and P6, which are active (red). In T2, both C3 and C5 are activated, in C3 only P9 is active as it predicted the column activation, while all cells are active in C5, as no prediction-cell predicted the column activation. **b)** Two cell's are shown with activation thresholds, $\tilde{\theta}$, of 2 and 3. Both have segments with two active cell's, but only the one with $\tilde{\theta} = 2$ has enough activity to enter a predictive state.

able to learn high order sequences, while still operating under the Markov assumption.

A cell learns causes by forming *synapses* along its dendritic *segments* to other cells using a Hebbian learning inspired method: when a column becomes active one of its cells will form synapses, from a single dendrite, to other cells which were active the previous time-step. On future time-steps if the number of active synapses on a dendrite is above an activation threshold then the cell will enter a predictive state. If the cell's column becomes active on the next time-step than that cell will become active. If no cells were in a predictive state when a column became active than all cells will become active. This is referred to as *bursting* a column and represents that the activation may have any number of causes. We display an example of this inference procedure in Figure 1.

Dynamic thresholds

In TPDT the activation threshold is dynamic, such that it can be continually modified for each cell on an individual basis. The effect of modifying the activation threshold is that when the threshold is higher more synapses on a segment are required to be active to put the cell in a predictive state than when the threshold is lower. So the higher the threshold the more 'evidence' the cell requires to enter a predictive state.

As a metric for TPDT we use the prior, or unconditional, probability of a cell's column being in an active state. We modify the threshold such that, the greater the prior probability the lower the activation threshold. In effect if the column has a high prior probability its cells require less evidence, and are more likely, to predict its activation. This approach has obvious roots in Bayesian inference, where the greater the prior probability, the greater the upper bound of the posterior probability.

A column's activity is a Boolean state variable, either active or non-active, and we measure this value over a time series. As such, the state is of a Bernoulli probability distribution and we calculate the prior probability of the variable being in an active state, $\tilde{p}$, as:

$$\tilde{p} = \frac{\tilde{a}}{t} \tag{1}$$

Where $\tilde{a}$ is the total number of column activations, and t is the total number of time steps. We use the value of $\tilde{p}$ as the metric to adjust the activation threshold of each cell as described applied below.

Implementation

In this section we detail the implementation of the above theories. We take a more algorithmic approach than Cui et al. (2016), but use the same notation where appropriate. As the TPDT alterations to TP primarily affect the inference phase we focus our discussion on inference and the dynamic thresholds.

Inference A single TP layer comprises $m \times n$ number of cells, where n is the number of columns, and m is the number of cells per column. Each cell has an activity state variable, which represents whether a cell is active (1) or inactive (0). We track this variable in a 2-dimensional array, $\mathbf{A}$, with dimensions $m \times n$. The input into TP is a binary array, $\mathbf{x}$, of n length, the source of the input is either the SP or some other sparse binary encoder. Each entry of $\mathbf{x}$ corresponds to a column activation, which is the proximal dendrite activity of each cell in the column. So, a value of 1 in the jth entry of the array will activate the jth column. To calculate the activity of individual cells, $\mathbf{A}$, we use the function:

$$\text{cellAct}(\mathbf{\Pi}, \mathbf{x}, i, j, m) = \begin{cases} 1 & \text{if } \mathbf{x}_j = 1 \wedge \big(\mathbf{\Pi}_{(i,j)} = 1 \vee \\ & \quad \forall \pi \in \mathbf{\Pi}_{(0\ldots m,j)} (\pi = 0)\big) \\ 0 & \text{otherwise} \end{cases}$$

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

where $\mathbf{\Pi}$ is a 2-dimensional array of size $m \times n$ which contains the predictive state variables of cells; j is the column index ($0 \leq j \leq n$) and i is the index of a cell within a column ($0 \leq i \leq m$). So if $\mathbf{x}_j = 1$ then at least one cell within the jth column will be active; if a cell predicted the activity than it will be active, otherwise the column will burst and all will be active.

When cells have been activated we can perform inference, starting with calculating the activity of segments. We represent segments using an array, $\mathbf{s}$, of binary 2-tuples; each 2-tuple represents a cell with which the segment has formed an active synapse. Both segments and synapses have activity state variables. A synapse's activity state variable is set to active if the pre-synaptic cell it is connected to is in an active state. As a result we can obtain the activity state of a synapse by accessing $\mathbf{A}$ with a 2-tuple, for example: $A_{(s_1,s_2)}$ where $s \in \mathbf{s}$. We store each $\mathbf{s}$ array in a 3-dimensional array $\mathbf{S}$ of size $m \times n \times k$, where k is the maximum number of segments on a cell. Using this we can calculate each segment's activity as:

$$\text{segAct}(\mathbf{s}, \mathbf{A}, \alpha) = \begin{cases} 1 & \text{if } \left(\sum_{s \in \mathbf{s}} [\mathbf{A}_{(s_1,s_2)} = 1] \right) \geq \theta \\ 0 & \text{otherwise} \end{cases}$$

where θ is the activation threshold. As can be seen, if the total number of active synapses on a segment is above θ then that segment is active.

After computing the activity states of cells and segments we now set the predictive state variables for cells. A cell's π is set to predictive (1) if any segment is active:

$$\text{cellPred}(\mathbf{D}, i, j, k) = \begin{cases} 1 & \text{if } \exists d \in \mathbf{D}_{(i,j,0\ldots k)} (d = 1) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathbf{D}$ is a 3-dimensional array of size $m \times n \times k$ containing the activity states of segments. The value of each cell's π will then be used for the activation of cells in the next time-step. The system progressives iteratively through this process for each time-step. We outline the procedure with pseudo-code in Algorithm 1.

Learning As our additions to TP primarily concern the inference phase of the algorithm we only provide a high-level description of this learning method (for a more in depth treatment we recommend Cui et al. (2016) or the CLA white paper (Hawkins et al., 2010).

Each synapse has a *connected* state variable, this can either be *connected* or *unconnected*. Synapses that are connected have an entry in their segment's $\mathbf{s}$ array, whereas unconnected synapses do not and, therefore, play no role in inference. Each synapse has a *permanence* value, when this value is above a permanence threshold value the synapse is connected. Synapse permanence values, both connected and unconnected, are altered on the time step following their segment being active. If the cell enters an active state (i.e. the segment correctly predicted the activation of the column) each synapse on the segment that was active on the preceding time-step has its permanence value incremented by a set amount (as per parameter); synapses that were not active have their permanence decremented. If the cell does not become active following a segment being active then all active synapses on that segment have their permanence values decremented.

Segments and synapses are formed when a column becomes active but none of its cells were in a predictive state. We search all cells in the column for the segment which had the most active synapses, including synapses that are unconnected. If the total number of active segments on this segment is above a set number, we add a number of synapses to the segment that connect to cells that were active. Otherwise we search for the cell with the least number of segments and create a new segment on it with synapses to cells that were active.

Dynamic thresholds

To the above algorithm we add dynamic thresholds which modify the activation threshold, θ, for individual cells. We denote a modified activation threshold as $\tilde{\theta}$, where $\tilde{\theta} \in \mathbb{Z}^+$. When applying dynamic thresholds during inference we pass the target cell's $\tilde{\theta}$ instead of the global θ variable when calling the function actSeg; we show this in Algorithm 1.

In this treatment we calculate each cell's $\tilde{\theta}$ by incorporating the prior, or unconditional, probability of each column activity. In TP each cell's proximal dendrite activity is encoded by columns, so a cell's prior probability of proximal dendrite activity is the column's prior probability of being active. We calculate the prior probability of a column being in an active state, $\tilde{p}$, using:

$$\text{calcPrior}(\tilde{\mathbf{x}}, t, j) = \frac{\tilde{\mathbf{x}}_j}{t}$$

where $\tilde{x}$ is an array containing the activity count of each column and t is the total number of time-steps the system has been exposed to. Before calculating $\tilde{\theta}$ we perform a few steps. Firstly we constrain the value of $\tilde{p}$ to be within a maximum value, $\tilde{p}^{\leq}$, we then divide by this value:

$$\text{constrainPrior}(\tilde{p}, \tilde{p}^{\leq}) = \begin{cases} 1.0 & \text{if } \tilde{p} \geq \tilde{p}^{\leq} \\ \tilde{p} \frac{1}{\tilde{p}^{\leq}} & \text{otherwise} \end{cases}$$

This is similar to a normalization process. We use this method as it prevents outliers from influencing the final value and is more biologically plausible, as it does not require the cell to be aware of other cell's prior probabilities.

Before calculating $\tilde{\theta}$ we multiply the 'constrained' prior, $\breve{p}$, by a value $\breve{p}^{\times}$, which modifies the effective range of $\tilde{\theta}$. For example if $\breve{p}^{\times} = 2$ then there would be three possible

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

values for $\tilde{\theta}$ (as $\tilde{\theta}$ is an integer value). As $\breve{p}^{\times}$ increases, so does the range of possible $\tilde{\theta}$ values. Finally we calculate $\tilde{\theta}$ using:

$$\text{calcDynThresh}(\breve{p}, \breve{p}^{\times}, \tilde{\theta}^{\leq}) = \lfloor\ \tilde{\theta}^{\leq} - \breve{p}\breve{p}^{\times}\ \rfloor$$

where $\tilde{\theta}^{\leq}$ is the maximum value for $\tilde{\theta}$. Pseudo-code for inference with TPDT is provided in Algorithm 1.

Algorithm 1 Inference and Dynamic Thresholds

Require: f {TPDT flag: 1 if using TPDT, 0 if TP}
Require: $\mathbf{\Pi}, \mathbf{x}, \mathbf{S}, \theta, m, n, k$ {TP parameters}
Require: $\tilde{\mathbf{x}}, t, \tilde{p}^{\leq}, \breve{p}^{\times}, \tilde{\theta}^{\leq}$ {TPDT parameters}

$t \leftarrow t + 1$
for $j \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** m **do**
    $\mathbf{A}_{(j,i)} \leftarrow \text{cellActive}(\mathbf{\Pi}, \mathbf{x}, j, i, m)$
  if $f = 1$ **then**
    if $\mathbf{x}_j = 1$ **then**
      $\tilde{\mathbf{x}}_j \leftarrow \tilde{\mathbf{x}}_j + 1$
for $j \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** m **do**
    if $f = 1$ **then**
      $\tilde{p} \leftarrow \text{calcPrior}(\tilde{\mathbf{x}}, t, j)$
      $\breve{p} \leftarrow \text{constrainPrior}(\tilde{p}, \tilde{p}^{\leq})$
      $\tilde{\theta} \leftarrow \text{calcDynThresh}(\breve{p}, \breve{p}^{\times}, \tilde{\theta}^{\leq})$
    else
      $\tilde{\theta} \leftarrow \theta$
    for $k \leftarrow 1$ **to** p **do**
      $\mathbf{D}_{(i,j,k)} \leftarrow \text{segActive}(\mathbf{S}_{(i,j,k)}, \mathbf{A}, \tilde{\theta})$
    $\mathbf{\Pi}_{(i,j)} \leftarrow \text{cellPredictive}(\mathbf{D}, i, j, k)$

HTM Computational Framework

The TP is typically deployed as part of an HTM computational framework (Cui et al., 2016). To adequately compare TP with and without dynamic thresholds we use this framework for a number of experiments, following the methodology used in the literature. As well as the TP, this framework includes a number of other computation steps, which we will outline below.

Framework

The first step in the HTM framework encodes input into binary representations, which is the required format for both SP and TP. For real valued input, a selection of scalar encoders can be used. We use the random distributed scalar encoder that preserves important properties around overlapping representations, e.g. scalars with a small difference in value will have a high degree of overlap between their binary patterns, whereas scalars with large differences in value will have little or no overlap in their bit patterns. This scheme removes the need for later stages of the framework having to learn that similar values are indeed similar.

Next the SP algorithm is used to group, or 'pool', highly correlated output bits from the scalar encoder into individual features that will be used to activate the columns in TP (Hawkins et al., 2010). In SP, each column has a single segment which forms synapses to input bits. This is achieved using a Hebbian learning method similar to TP, here synapse permanences are increased when input bits are 'on' and the column is active. Instead of an activation threshold (as in TP), columns compete to be active, where the columns with the highest *overlap* score becoming active. Overlap is simply a column's total number of active synapses multiplied by a *boost* modifier, which is used to help inactive columns become more active. This competitive encoding is how sparsity is enforced, with only a small proportion of columns becoming active on each time-step. The output of SP is a sparse distributed representation (SDR) of column activity, in the form of a binary vector, which is used as input into the TP. For a more in-depth look at SP we recommend the mathematical formulation by Mnatzaganian et al. (2016) or the white paper (Hawkins et al., 2010).

The TP then forms predictions on the SDR vector using the methods outlined above. However the TP used for anomaly detection has a number of divergences from these methods, which we discuss below. After TP has formed predictions, the final step of the framework decodes the state of the system, the method used for this is largely task dependent. For the artificial data experiments we use a simple method that compares the predicted SDR with those that have previously been used as input, and then assign the closest matching SDR based on the overlap of the two. For the taxi cab experiment the range of possible input values is separated into 22 buckets, and we use a maximum likelihood classifier to assign the state of the system to one of these buckets. The methodology for both of these experiments follows that of Cui et al. (2016). For the anomaly detection experiment we use the scheme used by Lavin and Ahmad (2015). On each time-step the current activity states of columns are compared to the predictive states of cells from the previous time-step. Based on this comparison the time-step is assigned an anomaly score between 0.0 and 1.0, where 0.0 represents the next time-step was perfectly predicted and 1.0 indicates significant prediction errors. We calculate an anomaly likelihood by comparing the anomaly score at any time-step to the distribution of recent anomaly scores. This value is then thresholded before we output whether the input data for that time-step is anomalous or not.

Temporal pooler for on-line anomaly detection

In TP for anomaly detection the learning phase has its own inference phase that only applies to learning. Here only a single cell in a column can be in a predictive state on a given time-step. The single predictive cell is the one with the highest activation count on a segment. Due to this method, and

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

because there is no column bursting in learning, only a single cell can be active in a column. This constrains TP to learning on only those cells most likely to implicated in the current input. In our TPDT implementation we use $\tilde{\theta}$ for the activation thresholds both in learning and inference.

In this version of TP each column has a single *start cell*, these are used to indicate the beginning of a sequence and are triggered under three circumstances. The first is on the very first time-step of anomaly detection and the other two are based on the TP's internal states. The second is when TP has been *out of sequence* for a pre-set number of time-steps (Lavin and Ahmad (2015) use three time-steps). The system is considered out of sequence when greater than 50% of the predicted column activations are incorrect. But before giving up on an out of sequence time-step the system performs a backtracking method: this is the third circumstance start cells are used. Backtracking replays time-steps from a number of starting points. On these starting points start-cells are the only active cells on that time-step, the time-steps are then replayed in an attempt to get TP to lock on to the current sequence.

Although these added features to TP detract somewhat from its biological plausibility they can improve the system's ability to learn and infer on complex input sequences. Consequently, in order to make a fair comparison between the state-of-the-art TP and TPDT, TPDT also uses these features for anomaly detection.

Experiments

To test whether precisions can improve the predictive performance of TP, we apply TPDT to three experiments of TP in the literature and compare the results. For all experiments we use the same framework as the initial study and we replicate their results with TP. The results are averaged over a number of runs, altering the random seed for each separate run.

Taxi Passenger Demand

This experiment tests a sequence learning algorithm's predictive capabilities in a real-world scenario: predicting the taxi demand in two and half hours, given the current demand and time. The HTM framework has been shown to produce state-of-the-art results on this problem (Cui et al., 2016).

Experimental Design Before running TP on this dataset, SP is applied over the first 5,000 time-steps for five iterations. TP is only trained for one iteration over the 5,000 and after this we begin scoring, which we detail below. The parameters we use are: $\breve{\theta}^{\times} = 5.0$, $\tilde{\theta}^{\leq} = 0.05$, and $\tilde{\theta}^{\leq} = 20$

The dataset contains the aggregate demand for taxis in New York City over 30 minute periods spanning July 1st 2014 through to June 30th 2015, there are a total of 17,520 time-steps. This dataset is derived from data made a available by the New York City Transit Authority. All possible values are separated into 22 buckets. We use maximum likelihood classifier to predict the bucket given the current state of the system. We use both the mean absolute percentage error (MAPE) and negative log likelihood (NegLL) as metrics for error; MAPE is the error given the most likely bucket, while NegLL incorporates the probabilities across all the buckets, allowing for the consideration of multiple predictions (Cui et al., 2016).

Table 1: Results for TP and TPDT on NAB

	MAPE		NegLL	
	Avg	StdDev	Avg	StdDev
TP	0.0869	0.0005	1.6563	0.0071
TPDT	**0.0858**	0.0006	**1.6439**	0.0052

Results The results for this experiment are given in Table 1, showing the average and standard deviation over the 25 trials. For both MAPE and NegLL lower values are better and TPDT shows improved performance on both these scores. Although the differences are small, the variance across the trials is quite low and the differences are significant using a one-tailed t-test ($p < 0.05$). These results therefore indicate that dynamic thresholds can improve the accuracy of TP for sequence prediction.

Online anomaly detection

To test the capabilities of TPDT for anomaly detection, we embed it in the HTM framework described previously and apply it to NAB.

Experimental Design On this experiment there is less pre-training for spatial pooler, so the input has a lower lifetime sparsity and consequently there are columns with much higher prior probabilities. Because of this we use a greater $\tilde{\theta}^{\leq}$. The parameters used for TPDT in this experiment is: $\breve{\theta}^{\times} = 2.5$, $\tilde{\theta}^{\leq} = 0.85$, and $\tilde{\theta}^{\leq} = 15$. All other parameters are the same as the TP.

NAB contains 58 datasets of time-series data, each with between 1,000 to 22,000 time-steps. These datasets are from a variety of different domains, such as server load, taxi bookings, machine temperature, and artificial data. All data has been labelled to include known anomalies within it. For scoring in NAB, each anomaly is assigned an *anomaly window*, if an anomaly detector detects an anomaly within this window it is assigned a score; the earlier in the window the anomaly is detected the higher the score. If no anomalies are detected within the anomaly window or anomalies are detected outside of an anomaly window a negative score is assigned. Final scores are given for three separate profiles: standard, reward low false-positives (FP) where negative scores from detecting an anomaly outside of a window are given greater weight, and reward low false-negatives (FN) where negative scores from not detecting an anomaly within

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

Table 2: Results for TP and TPDT on NAB

	Standard		Minimize FP		Minimize FN	
	Avg	Best	Avg	Best	Avg	Best
TP	69.97	71.88	62.84	65.90	74.50	76.36
DT	**70.22**	72.64	62.73	66.03	**74.85**	76.88

an anomaly window are given greater weight.

Results The results of the experiments on NAB are shown in Table 2, where again results in bold indicate the better results that have statistical significance on a one-tailed t-test ($p < 0.05$). TPDT shows gains on both the standard and minimize FN profiles; the minimize FP profile shows a minor (not statistically significant) drop over TP. The 'best' results indicate the best score achieved over the 50 trials, and they appear to follow the same pattern as the averages: TPDT gives improved results for both standard and minimize FN.

We have replicated the results for TP on the NAB scoreboard (as of writing): with a standard profile of 70.10, FP of 70.10, and FN of 69.9 (Lavin, 2017). This result is achieved with a seed value of 1960, which is the default for the NuPIC framework (Numenta, 2017). However, as shown in Table 2 we find a greater best score for TP, although the average across 50 trials is slightly lower. Regardless, TPDT's average is greater than the reported score, statistically significantly better than TP's average, and TPDT's best score is greater than TP's. To the best of our knowledge, TPDT's is therefore state-of-the-art on NAB for both the standard and the minimize FN profiles.

Discussion

The idea of the brain as a Bayesian prediction engine has its roots in the 19th century, with Helmholtz describing perception as a form of inference (von Helmholtz, 1867). More modern theories have expanded greatly on this insight, arguing that cognitive tasks such as attention and motor control can be explained through predictive techniques such as expected precision weighting and active inference, respectively (Feldman and Friston, 2010; Friston et al., 2009). These theories draw an understanding of brain function from a deep well of scientific knowledge, encompassing Bayesian statistics, thermodynamics, information theory, and cybernetics (Friston et al., 2006; Seth, 2014). Within this line of research we find HTM, a theory of particular interest as it makes specific predictions regarding the functionality of both pyramidal cells and the general architecture of the neocortex, positing that the distal dendrites learn causes of proximal dendrite activity and that when embedded within a network the system as a whole learns and predicts on temporal sequences (Hawkins and Ahmad, 2016).

To the HTM framework and, more generally, the model itself we add the concept of dynamic thresholds. This adds a further level of adaptation to the system, where not only does each cell's connectivity adapt to its environment, but the internal dynamics of each cell adapts as well. We implemented this idea in TP for the activation threshold parameter, which governs the amount of distal dendrite segment activity required for the cell to enter a predictive state. As a metric we use the prior probability of a cell's proximal dendrites becoming active. The higher the probability of proximal dendrite activity, the lower the threshold and the more likely a distal dendrite will predict future proximal dendrite activity. In considering the prior probability in the encoding scheme, we add a further element of basic Bayesian inference into the HTM framework.

Our experiments indicate that TPDT can improve the performance of the HTM framework for sequence learning. We achieved state-of-the-art results on NAB, a benchmark that is geared at evaluating online anomaly detecting algorithms across a number of domains, using a scoring method that prioritises early detection. This shows that, despite the theoretical underpinnings of this work, TPDT has obvious practical applications and could be deployed in industry settings.

Future work on TPDT could look at other parameters to add dynamism to, such as connected permanence, and the increment and decrement values used in learning. Dynamically adjusting values in this way could be key to introducing the concept of expected precisions (inverse variance) to HTM. Expected precision is major facet of PP, measuring the level of confidence in error signals, and is conceptually similar to Kalman weights used in Kalman filters. A successful scheme could improve the resilience of TP to temporal noise (Cui et al., 2016).

Conclusions

We have presented a modification to TP, an algorithm in which individual cells learn to predict their own activity and when networked form a system that can learn and predict temporal sequences. Our modified algorithm (TPDT) adds the concept of dynamic thresholds, where the activation threshold (level of activity needed to form a prediction) is continuously and individually updated for each cell. As a metric for this adjustment we use the prior probability of the cell becoming active. Our experiments indicate that dynamic thresholds can improve the performance of TP for both sequence prediction and anomaly detection, where we achieve state-of-the-art results on NAB. The major contributions of this work are to have improved the performance of TP and to have introduced a new concept to the algorithm, in which internal dynamics of the cells (not only the connectivity) can adapt to the environment in which the system is embedded.

References

Ahmad, S. and Hawkins, J. (2015). Properties of sparse distributed representations and their applica-

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).

tion to hierarchical temporal memory. *arXiv preprint arXiv:1503.07469*.

Clark, A. (2015). *Surfing Uncertainty: Prediction, Action, and the Embodied Mind*. Oxford University Press.

Cui, Y., Ahmad, S., and Hawkins, J. (2016). Continuous online sequence learning with an unsupervised neural network model. *Neural computation*, 28(11):2474–2504.

Feldman, H. and Friston, K. J. (2010). Attention, uncertainty, and free-energy. *Frontiers in human neuroscience*, 4.

Friston, K., Kilner, J., and Harrison, L. (2006). A free energy principle for the brain. *Journal of Physiology-Paris*, 100(1):70–87.

Friston, K. J., Daunizeau, J., and Kiebel, S. J. (2009). Reinforcement learning or active inference? *PloS one*, 4(7):e6421.

Häusser, M., Spruston, N., and Stuart, G. J. (2000). Diversity and dynamics of dendritic signaling. *Science*, 290(5492):739–744.

Hawkins, J. and Ahmad, S. (2016). Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in neural circuits*, 10.

Hawkins, J., Ahmad, S., and Dubinsky, D. (2010). Hierarchical temporal memory including HTM cortical learning algorithms. *Techical report, Numenta, Inc, Palto Alto*.

Hawkins, J. and Blakeslee, S. (2007). *On intelligence*. Macmillan.

Hebb, D. O. (1949). *The organization of behavior: A neuropsychological approach*. John Wiley & Sons.

Kneller, A. and Thornton, J. (2015). Distal dendrite feedback in hierarchical temporal memory. In *Proceedings of the 2015 International Joint Conference on Neural Networks*.

Knill, D. C. and Pouget, A. (2004). The Bayesian brain: the role of uncertainty in neural coding and computation. *Trends in neurosciences*, 27(12):712–719.

Lavin, A. (2017). NAB scoreboard. [Online website] Available: https://github.com/numenta/NAB.

Lavin, A. and Ahmad, S. (2015). Evaluating real-time anomaly detection algorithms the Numenta anomaly bench. In *14th International Conference on Machine Learning and Applications*.

Major, G., Larkum, M. E., and Schiller, J. (2013). Active properties of neocortical pyramidal neuron dendrites. *Annual review of neuroscience*, 36:1–24.

McCall, R. and Franklin, S. (2013). Cortical learning algorithms with predictive coding for a systems-level cognitive architecture. In *Second Annual Conference on Advances in Cognitive Systems Poster Collection*, pages 149–166.

Mnatzaganian, J., Fokoué, E., and Kudithipudi, D. (2016). A mathematical formalization of hierarchical temporal memory's spatial pooler. *Frontiers in Robotics and AI*, 3:81.

Mountcastle, V. B. (1997). The columnar organization of the neocortex. *Brain*, 120(4):701–722.

Numenta (2017). NuPIC: Numenta platform for intelligent computing. [Online code repository] Available: https://github.com/numenta/nupic.

Olshausen, B. A. and Field, D. J. (2004). Sparse coding of sensory inputs. *Current opinion in neurobiology*, 14(4):481–487.

Rao, R. P. and Ballard, D. H. (1999). Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *Nature neuroscience*, 2(1):79–87.

Seth, A. K. (2014). The cybernetic Bayesian brain. In *Open Mind*. Open MIND. Frankfurt am Main: MIND Group.

Skrynnik, A., Petrov, A., and Panov, A. I. (2016). Hierarchical temporal memory implementation with explicit states extraction. In *Biologically Inspired Cognitive Architectures (BICA) for Young Scientists*, pages 219–225. Springer.

Thornton, J. and Srbic, A. (2013). Spatial pooling for greyscale images. *International Journal of Machine Learning and Cybernetics*, 4(3):207–216.

von Helmholtz, H. (1867). *Handbuch der physiologischen Optik*, volume 9. Voss.

Carole Knibbe et al, eds., Proceedings of the ECAL 2017, Lyon, France, 4-8 September 2017,
(Cambridge, MA: The MIT Press, ©2017 Massachusetts Institute of Technology).