Title: Hierarchical temporal memory

URL: https://en.wikipedia.org/wiki/Hierarchical_temporal_memory

PageID: 11273721

Categories: Category:Artificial neural networks, Category:Belief revision, Category:Deep learning, Category:Semisupervised learning, Category:Unsupervised learning

---

Hierarchical temporal memory (HTM) is a biologically constrained machine intelligence technology developed by Numenta . Originally described in the 2004 book On Intelligence by Jeff Hawkins with Sandra Blakeslee , HTM is primarily used today for anomaly detection in streaming data. The technology is based on neuroscience and the physiology and interaction of pyramidal neurons in the neocortex of the mammalian (in particular, human) brain.

At the core of HTM are learning algorithms that can store, learn, infer , and recall high-order sequences. Unlike most other machine learning methods, HTM constantly learns (in an unsupervised process) time-based patterns in unlabeled data. HTM is robust to noise, and has high capacity (it can learn multiple patterns simultaneously). When applied to computers, HTM is well suited for prediction, [1] anomaly detection, [2] classification, and ultimately sensorimotor applications. [3]

HTM has been tested and implemented in software through example applications from Numenta and a few commercial applications from Numenta's partners [clarification needed] .

Structure and algorithms

A typical HTM network is a tree -shaped hierarchy of levels (not to be confused with the " layers " of the neocortex , as described below). These levels are composed of smaller elements called region s (or nodes). A single level in the hierarchy possibly contains several regions. Higher hierarchy levels often have fewer regions. Higher hierarchy levels can reuse patterns learned at the lower levels by combining them to memorize more complex patterns.

Each HTM region has the same basic function. In learning and inference modes, sensory data (e.g. data from the eyes) comes into bottom-level regions. In generation mode, the bottom level regions output the generated pattern of a given category. The top level usually has a single region that stores the most general and most permanent categories (concepts); these determine, or are determined by, smaller concepts at lower levels—concepts that are more restricted in time and space [clarification needed] . When set in inference mode, a region (in each level) interprets information coming up from its "child" regions as probabilities of the categories it has in memory.

Each HTM region learns by identifying and memorizing spatial patterns—combinations of input bits that often occur at the same time. It then identifies temporal sequences of spatial patterns that are likely to occur one after another.

As an evolving model

HTM is the algorithmic component to Jeff Hawkins ’ Thousand Brains Theory of Intelligence. So new findings on the neocortex are progressively incorporated into the HTM model, which changes over time in response. The new findings do not necessarily invalidate the previous parts of the model, so ideas from one generation are not necessarily excluded in its successive one. Because of the evolving nature of the theory, there have been several generations of HTM algorithms, [4] which are briefly described below.

First generation: zeta 1

The first generation of HTM algorithms is sometimes referred to as zeta 1 .

Training

During training , a node (or region) receives a temporal sequence of spatial patterns as its input. The learning process consists of two stages:

The spatial pooling identifies (in the input) frequently observed patterns and memorise them as "coincidences". Patterns that are significantly similar to each other are treated as the same coincidence. A large number of possible input patterns are reduced to a manageable number of known coincidences.

The temporal pooling partitions coincidences that are likely to follow each other in the training sequence into temporal groups. Each group of patterns represents a "cause" of the input pattern (or "name" in On Intelligence).

The concepts of spatial pooling and temporal pooling are still quite important in the current HTM algorithms. Temporal pooling is not yet well understood, and its meaning has changed over time (as the HTM algorithms evolved).

Inference

During inference , the node calculates the set of probabilities that a pattern belongs to each known coincidence. Then it calculates the probabilities that the input represents each temporal group. The set of probabilities assigned to the groups is called a node's "belief" about the input pattern. (In a simplified implementation, node's belief consists of only one winning group). This belief is the result of the inference that is passed to one or more "parent" nodes in the next higher level of the hierarchy.

"Unexpected" patterns to the node do not have a dominant probability of belonging to any one temporal group but have nearly equal probabilities of belonging to several of the groups. If sequences of patterns are similar to the training sequences, then the assigned probabilities to the groups will not change as often as patterns are received. The output of the node will not change as much, and a resolution in time [clarification needed] is lost.

In a more general scheme, the node's belief can be sent to the input of any node(s) at any level(s), but the connections between the nodes are still fixed. The higher-level node combines this output with the output from other child nodes thus forming its own input pattern.

Since resolution in space and time is lost in each node as described above, beliefs formed by higher-level nodes represent an even larger range of space and time. This is meant to reflect the organisation of the physical world as it is perceived by the human brain. Larger concepts (e.g. causes, actions, and objects) are perceived to change more slowly and consist of smaller concepts that change more quickly. Jeff Hawkins postulates that brains evolved this type of hierarchy to match, predict, and affect the organisation of the external world.

More details about the functioning of Zeta 1 HTM can be found in Numenta's old documentation. [5]

Second generation: cortical learning algorithms

The second generation of HTM learning algorithms, often referred to as cortical learning algorithms (CLA), was drastically different from zeta 1. It relies on a data structure called sparse distributed representations (that is, a data structure whose elements are binary, 1 or 0, and whose number of 1 bits is small compared to the number of 0 bits) to represent the brain activity and a more biologically-realistic neuron model (often also referred to as cell , in the context of HTM). [6] There are two core components in this HTM generation: a spatial pooling algorithm, [7] which outputs sparse distributed representations (SDR), and a sequence memory algorithm, [8] which learns to represent and predict complex sequences.

In this new generation, the layers and minicolumns of the cerebral cortex are addressed and partially modeled. Each HTM layer (not to be confused with an HTM level of an HTM hierarchy, as described above) consists of a number of highly interconnected minicolumns. An HTM layer creates a sparse distributed representation from its input, so that a fixed percentage of minicolumns are active at any one time [clarification needed] . A minicolumn is understood as a group of cells that have the same receptive field . Each minicolumn has a number of cells that are able to remember several previous states. A cell can be in one of three states: active , inactive and

predictive state.

Spatial pooling

The receptive field of each minicolumn is a fixed number of inputs that are randomly selected from a much larger number of node inputs. Based on the (specific) input pattern, some minicolumns will be more or less associated with the active input values. Spatial pooling selects a relatively constant number of the most active minicolumns and inactivates (inhibits) other minicolumns in the vicinity of the active ones. Similar input patterns tend to activate a stable set of minicolumns. The amount of memory used by each layer can be increased to learn more complex spatial patterns or decreased to learn simpler patterns.

As mentioned above, a cell (or a neuron) of a minicolumn, at any point in time, can be in an active, inactive or predictive state. Initially, cells are inactive.

If one or more cells in the active minicolumn are in the predictive state (see below), they will be the only cells to become active in the current time step. If none of the cells in the active minicolumn are in the predictive state (which happens during the initial time step or when the activation of this minicolumn was not expected), all cells are made active.

When a cell becomes active, it gradually forms connections to nearby cells that tend to be active during several previous time steps. Thus a cell learns to recognize a known sequence by checking whether the connected cells are active. If a large number of connected cells are active, this cell switches to the predictive state in anticipation of one of the few next inputs of the sequence.

The output of a layer includes minicolumns in both active and predictive states. Thus minicolumns are active over long periods of time, which leads to greater temporal stability seen by the parent layer.

Inference and online learning

Cortical learning algorithms are able to learn continuously from each new input pattern, therefore no separate inference mode is necessary. During inference, HTM tries to match the stream of inputs to fragments of previously learned sequences. This allows each HTM layer to be constantly predicting the likely continuation of the recognized sequences. The index of the predicted sequence is the output of the layer. Since predictions tend to change less frequently than the input patterns, this leads to increasing temporal stability of the output in higher hierarchy levels. Prediction also helps to fill in missing patterns in the sequence and to interpret ambiguous data by biasing the system to infer what it predicted.

Applications of the CLAs

Cortical learning algorithms are currently being offered as commercial SaaS by Numenta (such as Grok [9]).

The validity of the CLAs

The following question was posed to Jeff Hawkins in September 2011 with regard to cortical learning algorithms: "How do you know if the changes you are making to the model are good or not?" To which Jeff's response was "There are two categories for the answer: one is to look at neuroscience, and the other is methods for machine intelligence. In the neuroscience realm, there are many predictions that we can make, and those can be tested. If our theories explain a vast array of neuroscience observations then it tells us that we’re on the right track. In the machine learning world, they don’t care about that, only how well it works on practical problems. In our case that remains to be seen. To the extent you can solve a problem that no one was able to solve before, people will take notice." [10]

Third generation: sensorimotor inference

The third generation builds on the second generation and adds in a theory of sensorimotor inference in the neocortex. [11] [12] This theory proposes that cortical columns at every level of the hierarchy can learn complete models of objects over time and that features are learned at specific locations on the objects. The theory was expanded in 2018 and referred to as the Thousand Brains Theory. [13]

Comparison of neuron models

Few synapses

No dendrites

Sum input × weights

Learns by modifying weights of synapses

Thousands of synapses on the dendrites

Active dendrites: cell recognizes hundreds of unique patterns

Co-activation of a set of synapses on a dendritic segment causes an NMDA spike and depolarization at the soma [8]

Sources of input to the cell: Feedforward inputs which form synapses proximal to the soma and directly lead to action potentials NMDA spikes generated in the more distal basal [clarification needed] Apical dendrites that depolarize the soma (usually insufficient to generate a somatic action potential)

Feedforward inputs which form synapses proximal to the soma and directly lead to action potentials

NMDA spikes generated in the more distal basal [clarification needed]

Apical dendrites that depolarize the soma (usually insufficient to generate a somatic action potential)

Learns by growing new synapses

Inspired by the pyramidal cells in neocortex layers 2/3 and 5

Thousands of synapses

Active dendrites: cell recognizes hundreds of unique patterns

Models dendrites and NMDA spikes with each array of coincident detectors having a set of synapses

Learns by modeling the growth of new synapses

Comparing HTM and neocortex

HTM attempts to implement the functionality that is characteristic of a hierarchically related group of cortical regions in the neocortex. A region of the neocortex corresponds to one or more levels in the HTM hierarchy, while the hippocampus is remotely similar to the highest HTM level. A single HTM node may represent a group of cortical columns within a certain region.

Although it is primarily a functional model, several attempts have been made to relate the algorithms of the HTM with the structure of neuronal connections in the layers of neocortex. [14] [15] The neocortex is organized in vertical columns of 6 horizontal layers. The 6 layers of cells in the neocortex should not be confused with levels in an HTM hierarchy.

HTM nodes attempt to model a portion of cortical columns (80 to 100 neurons) with approximately 20 HTM "cells" per column. HTMs model only layers 2 and 3 to detect spatial and temporal features of the input with 1 cell per column in layer 2 for spatial "pooling", and 1 to 2 dozen per column in layer 3 for temporal pooling. A key to HTMs and the cortex's is their ability to deal with noise and variation in the input which is a result of using a "sparse distributive representation" where only about 2% of the columns are active at any given time.

An HTM attempts to model a portion of the cortex's learning and plasticity as described above. Differences between HTMs and neurons include: [16]

strictly binary signals and synapses

no direct inhibition of synapses or dendrites (but simulated indirectly)

currently only models layers 2/3 and 4 (no 5 or 6)

no "motor" control (layer 5)

no feed-back between regions (layer 6 of high to layer 1 of low)

Sparse distributed representations

Integrating memory component with neural networks has a long history dating back to early research in distributed representations [17] [18] and self-organizing maps . For example, in sparse distributed memory (SDM), the patterns encoded by neural networks are used as memory addresses for content-addressable memory , with "neurons" essentially serving as address encoders and decoders. [19] [20]

Computers store information in dense representations such as a 32-bit word , where all combinations of 1s and 0s are possible. By contrast, brains use sparse distributed representations (SDRs). [21] The human neocortex has roughly 16 billion neurons, but at any given time only a small percent are active. The activities of neurons are like bits in a computer, and so the representation is sparse. Similar to SDM developed by NASA in the 80s [19] and vector space models used in Latent semantic analysis , HTM uses sparse distributed representations. [22]

The SDRs used in HTM are binary representations of data consisting of many bits with a small percentage of the bits active (1s); a typical implementation might have 2048 columns and 64K artificial neurons where as few as 40 might be active at once. Although it may seem less efficient for the majority of bits to go "unused" in any given representation, SDRs have two major advantages over traditional dense representations. First, SDRs are tolerant of corruption and ambiguity due to the meaning of the representation being shared (distributed) across a small percentage (sparse) of active bits. In a dense representation, flipping a single bit completely changes the meaning, while in an SDR a single bit may not affect the overall meaning much. This leads to the second advantage of SDRs: because the meaning of a representation is distributed across all active bits, the similarity between two representations can be used as a measure of semantic similarity in the objects they represent. That is, if two vectors in an SDR have 1s in the same position, then they are semantically similar in that attribute. The bits in SDRs have semantic meaning, and that meaning is distributed across the bits. [22]

The semantic folding theory [23] builds on these SDR properties to propose a new model for language semantics, where words are encoded into word-SDRs and the similarity between terms, sentences, and texts can be calculated with simple distance measures.

Similarity to other models

Bayesian networks

Likened to a Bayesian network , an HTM comprises a collection of nodes that are arranged in a tree-shaped hierarchy. Each node in the hierarchy discovers an array of causes in the input patterns and temporal sequences it receives. A Bayesian belief revision algorithm is used to propagate feed-forward and feedback beliefs from child to parent nodes and vice versa. However, the analogy to Bayesian networks is limited, because HTMs can be self-trained (such that each node has an unambiguous family relationship), cope with time-sensitive data, and grant mechanisms for covert attention .

A theory of hierarchical cortical computation based on Bayesian belief propagation was proposed earlier by Tai Sing Lee and David Mumford . [24] While HTM is mostly consistent with these ideas, it adds details about handling invariant representations in the visual cortex. [25]

Neural networks

Like any system that models details of the neocortex, HTM can be viewed as an artificial neural network . The tree-shaped hierarchy commonly used in HTMs resembles the usual topology of traditional neural networks. HTMs attempt to model cortical columns (80 to 100 neurons) and their interactions with fewer HTM "neurons". The goal of current HTMs is to capture as much of the functions of neurons and the network (as they are currently understood) within the capability of typical computers and in areas that can be made readily useful such as image processing. For example, feedback from higher levels and motor control is not attempted because it is not yet

understood how to incorporate them and binary instead of variable synapses are used because they were determined to be sufficient in the current HTM capabilities.

LAMINART and similar neural networks researched by Stephen Grossberg attempt to model both the infrastructure of the cortex and the behavior of neurons in a temporal framework to explain neurophysiological and psychophysical data. However, these networks are, at present, too complex for realistic application. [26]

HTM is also related to work by Tomaso Poggio , including an approach for modeling the ventral stream of the visual cortex known as HMAX. Similarities of HTM to various AI ideas are described in the December 2005 issue of the Artificial Intelligence journal. [27]

Neocognitron

Neocognitron , a hierarchical multilayered neural network proposed by Professor Kunihiko Fukushima in 1987, is one of the first deep learning neural network models. [28]

See also

Artificial consciousness

Artificial general intelligence

Belief revision

Cognitive architecture

Convolutional neural network

List of artificial intelligence projects

Memory-prediction framework

Multiple trace theory

Neural history compressor

Neural Turing machine

Related models

Hierarchical hidden Markov model

References

Further reading

Ahmad, Subutai; Hawkins, Jeff (25 March 2015). "Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory". arXiv : 1503.07469 [q-bio.NC].

Hawkins, Jeff (April 2007). "Learn like a Human" . IEEE Spectrum .

Maltoni, Davide (April 13, 2011). "Pattern Recognition by Hierarchical Temporal Memory" (PDF) . DEIS Technical Report. Italy: University of Bologna.

Ratliff, Evan (March 2007). "The Thinking Machine" . Wired .

External links

HTM at Numenta

HTM Basics with Rahul (Numenta), talk about the cortical learning algorithm (CLA) used by the HTM model on YouTube