# Hierarchical Temporal Memory

Concepts, Theory, and Terminology

**Jeff Hawkins and Dileep George, Numenta Inc.**

## Introduction

There are many things humans find easy to do that computers are currently unable to do. Tasks such as visual pattern recognition, understanding spoken language, recognizing and manipulating objects by touch, and navigating in a complex world are easy for humans. Yet, despite decades of research, we have no viable algorithms for performing these and other cognitive functions on a computer.

In a human, these capabilities are largely performed by the neocortex. Hierarchical Temporal Memory (HTM) is a technology that replicates the structural and algorithmic properties of the neocortex. HTM therefore offers the promise of building machines that approach or exceed human level performance for many cognitive tasks.

HTMs are unlike traditional programmable computers. With traditional computers, a programmer creates specific programs to solve specific problems. For example, one program may be used to recognize speech and another completely different program may be used to model weather. HTM, on the other hand, is best thought of as a memory system. HTMs are not programmed and do not execute different algorithms for different problems. Instead, HTMs "learn" how to solve problems. HTMs are trained by exposing them to sensory data and the capability of the HTM is determined largely by what it has been exposed to.

HTMs are organized as a tree-shaped hierarchy of nodes, where each node implements a common learning and memory function. HTMs store information throughout the hierarchy in a way that models the world. All objects in the world, be they cars, people, buildings, speech, or the flow of information across a computer network, have structure. This structure is hierarchical in both space and time. HTM memory is also hierarchical in both space and time, and therefore can efficiently capture and model the structure of the world.

HTMs are similar to Bayesian Networks; however, they differ from most Bayesian Networks in the way that time, hierarchy, action, and attention are used. HTMs can be implemented with software on traditional computer hardware, but it is best to think of an HTM as a memory system.

This paper describes the theory behind HTMs, what HTMs do, and how they do it. It describes in detail the two most important capabilities of HTMs, the ability to discover and infer causes. It introduces the concepts behind two other HTM capabilities, prediction and behavior.

This paper describes the theory behind Numenta's products, but does not describe the products themselves. Separate documentation describes Numenta's products and how to apply HTM technology to real world problems.

HTM was derived from biology. Therefore, there is a detailed mapping between HTM and the biological anatomy of neocortex. Interested readers can find a partial description of this in Chapter 6 of the book *On Intelligence* (Times Books, 2004). It is not necessary to know the biological mapping of HTM to deploy HTM-based systems.

The concepts behind Hierarchical Temporal Memory are not particularly hard, but there are a lot of them so the learning curve can be steep. This paper is designed to be readable by any sufficiently motivated person. We don't assume any particular mathematical ability. (A complete mathematical description of the algorithms described in this paper is available from Numenta under license.) Learning how to design an HTM-based system is about as difficult as learning how to write a complex software program. Anyone can learn how, but if you are starting from scratch, there is a lot to learn.

The paper is divided into seven major sections, listed below.

**1. What do HTMs do?**

**2. How do HTMs discover and infer causes?**

**3. Why is hierarchy important?**

**4. How does each node discover and infer causes?**

**5. Why is time essential for learning?**

**6. Questions**

**7. Conclusion**

## 1. What do HTMs do?

It has been known for over twenty-five years that the neocortex works on a common algorithm; vision, hearing, touch, language, behavior, and most everything else the neocortex does are manifestations of a single algorithm applied to different modalities of sensory input. The same is true for HTM. So when we describe what HTMs do and how they work, our explanation will be in a language that is independent of sensory modality. Once you understand how HTMs work, you will understand how HTMs can be applied to a large class of problems including many that have no human correlate.

HTMs perform the following four basic functions regardless of the particular problem they are applied to. The first two are required, and the latter two are optional.

1) Discover causes in the world
2) Infer causes of novel input
3) Make predictions
4) Direct behavior

We will look at each of these basic functions in turn.

### 1.1 Discover causes in the world

Figure 1 shows how an HTM system relates to the world. On the left of this figure is a box representing a world the HTM is to learn about. The world consists of objects and their relationships. Some of the objects in the world are physical such as cars, people, and buildings. Some of the objects in the world may not be physical such as ideas, words, songs, or the flow of information on a network. The important attribute of the objects in the world from an HTM's perspective is that they have persistent structure; they exist over time. We call the objects in the world "causes". You can think of this word by asking questions such as "what was the ultimate 'cause' of the pattern on my retina" or "what was the ultimate 'cause' of the sound entering my ears". There is a hierarchy of causes active in the world at any moment in time. While listening to spoken language, the causes of the sound entering your ears are phonemes, words, phrases, and ideas. These are simultaneously active and all valid causes of the auditory input.

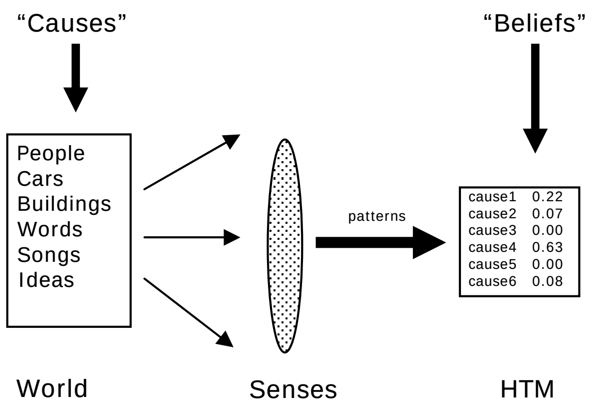

Figure 1

There is one large physical world that we all reside in. However, a particular HTM may be only concerned with a subset of this world. An HTM may be restricted to knowledge about financial markets, it may only interface to weather phenomenon, or it may only interface to and understand geophysical data, demographic data, or data collected from sensors attached to a car. From now on, when we refer to the "world" of the HTM, we mean the limited part to which the HTM is exposed.

On the right side of Figure 1 is an HTM. It interfaces to its world through one or more senses shown in the middle of the figure. The senses sample some attribute of the world such as light or touch, though the senses used by an HTM do not need to be the same senses humans have. Typically the senses don't directly detect the objects in the world. You don't have a "car sense" or a "word sense". Indeed, one of the goals of an HTM is to discover from the raw sensory input that objects like "cars" and "words" exist. Senses typically present an array of data to the HTM, where each element in the array is a measurement of some small attribute of the world. In a human, the optic nerve that carries information from the retina to the cortex consists of about one million fibers where each fiber carries information about light in a small part of visible space. The auditory nerve is about thirty thousand fibers, where each fiber carries information about sound in a small frequency range. The senses attached to an HTM will generally have a similar arrangement. That is, the sensory data will be a topologically arrayed collection of inputs, where each input measures a local and simple quantity.

All HTM systems have some type of sensory input, even if the data is coming from a file. From an HTM's perspective, there are two essential characteristics of sensory data. First, the sensory data must measure something that is directly or indirectly impacted by the causes in the world that you might be interested in. If you want the HTM to learn about weather, it must sense something related to weather such as air temperature and pressure at different locations. If the HTM is to understand computer network traffic, it might sense packets per second and CPU loads at routers. Second, the sensory data must change and flow continuously through time, while the causes underlying the sensory data remain relatively stable. The temporal aspect of sensory data can come from movements or changes of the objects in the world (such as a car driving by or the minute by minute fluctuations of a stock market), or it can come from movement of the sensory system itself through the world (as when you walk through a room or move your fingers over an object). Either way, the sensory data must change continuously over time for an HTM to learn.

The HTM receives the spatio-temporal pattern coming from the senses. At first, the HTM has no knowledge of the causes in the world, but through a learning process that will be described below, it "discovers" what the causes are. The end goal of this process is that the HTM develops internal representations of the causes in the world. In a brain, nerve cells learn to represent causes in the world, such as a cell that becomes active whenever you see a face. In an HTM, causes are represented by numbers in a vector. At any moment in time, given current and past input, an HTM will assign a likelihood that individual causes are currently being sensed. The HTM's output is manifest as a set of probabilities for each of the learned causes. This moment-to-moment distribution of possible causes is called a "belief". If an HTM knows about ten causes in the world, it will have ten variables representing those causes. The value of those variables – its belief – is what the HTM believes is happening in its world at that instant. Typical HTMs will know about many causes, and as you will see, HTMs actually learn a hierarchy of causes.

Discovering causes is at the heart of perception, creativity, and intelligence. Scientists try to discover the causes of physical phenomenon. Business people seek to discover the causes underlying markets and business cycles. Doctors seek to discover the causes of disease. From the moment you were born, your brain slowly learned representations for all the things you eventually came to know. You had to discover that cars, buildings, words, and ideas are persistent structures in the world. Before you are able to recognize something, your brain has to first discover that the thing exists.

All HTM systems need to go through a learning phase where the HTM learns what the causes in its world are. All HTMs first learn about the small and simple causes in their world. Large HTMs, when presented with enough sensory data, can learn high level, sophisticated causes. With sufficient training and proper design, it should be possible to build HTMs that discover causes humans have not been able to discover. After initial training, an HTM can continue to learn or not, depending on the needs of the application.

There can be much value in just discovering causes. Understanding the high level causes for market fluctuations, disease, weather, manufacturing yield, and failures of complex systems, such as power grids, is valuable. Discovering causes is also a necessary precursor for inference, the second capability of HTMs.

### 1.2 Infer causes of novel input

After an HTM has learned what the causes in its world are and how to represent them, it can perform inference. "Inference" is similar to pattern recognition. Given a novel sensory input stream, an HTM will "infer" what known causes are likely to be present in the world at that moment. For example, if you had an HTM-based vision system, you could show it pictures and it would infer what objects are in the picture. The result would be a distribution of beliefs across all the learned causes. If the picture was unambiguous, the belief distribution would be peaked. If the picture was highly ambiguous, the belief distribution would be flat because the HTM wouldn't be certain what it was looking at.

The current inferred beliefs of an HTM can be directly read from the system to be used elsewhere external to the HTM (something not possible in human brains!). Alternatively, the current belief can be used internally by the HTM to make predictions or to generate behavior.

In most HTM systems, the sensory input always will be novel. In a vision system attached to a camera, there might be one million pixels as sensory input. If the camera were looking at real world scenes, it is highly unlikely that the same pattern would ever be presented to the HTM twice. Thus, HTMs must handle novel input both during inference and during training. In fact, HTMs don't have a separate inference mode. HTMs are always inferring causes even while learning (albeit inferring poorly before much training has occurred). As mentioned earlier, it is possible to disable learning after training and still do inference.

Many HTM applications will require time-varying sensory input to do inference, although some do not. It depends on the nature of the sense and the causes. We can see this distinction in humans. Our auditory and tactile senses can infer almost nothing without time. We must move our hands over objects to infer what they are through touch. Similarly, a static sound conveys little meaning. Vision is mixed. Unlike touch and hearing, humans are able to recognize images (i.e. infer causes) when an image is flashed in front of them and the eyes do not have time to move. Thus, visual inference does not always require time changing inputs. However, during normal vision we move our eyes, we move our bodies, and objects in the world are moving too. So static, flashed picture identification is a special case made possible by the statistical properties of vision. The general case, even for vision, is that inference occurs with time-varying inputs.

Even though it is sometimes possible to perform inference with a static sensory pattern, the theory behind HTMs shows that it is not possible to discover causes without having continuously changing inputs. Thus, all HTM systems, even ones that do static inference, need to be trained on time-varying inputs. It isn't sufficient that sensory input just changes, for this could be accomplished with a succession of unrelated sensory patterns. Learning requires that causes persist while the sensory input changes. For example, when you move your fingers over an apple, although the tactile information is constantly changing, the underlying cause – the apple – stays constant. The same is true for vision. As your eyes scan over the apple, the pattern on the retina changes, but the underlying cause stays constant. Again, HTMs are not restricted to human type sensors: changing market data, changing weather, and dynamic traffic flow over computer networks all would suffice.

Inferring the causes of novel input is valuable. There are many pattern recognition problems that humans find easy but existing computer algorithms are unable to do. HTMs can solve these problems rapidly and accurately, just like humans. In addition, there are many inference problems that humans have difficulty performing that HTM-based systems can solve.

### 1.3 Make predictions

HTMs consist of a hierarchy of memory nodes where each node learns causes and forms beliefs. Part of the learning algorithm performed by each node is to store likely sequences of patterns. By combining memory of likely sequences with current input, each node has the ability to make predictions of what is likely to happen next. An entire HTM, being a collection of nodes, also makes predictions. Just as an HTM can infer the causes of novel input, it also can make predictions about novel events. Predicting the future of novel events is the essence of creativity and planning. Leaving the details of how this works for later, we can state now what prediction can be used for. There are several uses for prediction in an HTM, including priming, imagination and planning, and generating behavior. A few words on these uses are warranted at this time.

**Priming**
When an HTM predicts what is likely to happen next, the prediction can act as what is called a "prior probability", meaning it biases the system to infer the predicted causes. For example, if an HTM were processing text or spoken language, it would automatically predict what sounds, words, and ideas are likely to occur next. This prediction helps the system understand noisy or missing data. If an ambiguous sound arrives, the HTM will interpret the sound based on what it is expecting.

In an HTM, we have the ability to set prior probabilities manually in addition to having prior probabilities set via prediction. That is, we can manually tell the HTM to anticipate or look for a particular cause or set of causes, thus implementing a directed search.

**Imagination and Planning**
HTMs automatically predict and anticipate what is likely to happen next. Instead of using these predictions for priming, an HTM's predictions can be fed back into the HTM as a substitute for sensory data. This process is what humans do when they think. Thinking, imagining, planning the future, and silently rehearsing in our heads are all the same thing, and achieved by following a series of predictions. HTMs can do this as well. Imagining the future can be valuable in many applications. For example, suppose a car is equipped with an HTM to monitor nearby traffic. If a novel situation occurs, the HTM can follow a series of predictions to see what likely events will happen in the future, and therefore can imagine dangerous situations before they occur.

Prediction is also at the heart of how HTMs can direct motor behavior, the fourth and last capability of HTM.

## 1.4 Direct behavior

An HTM that has learned the causes in its world, and how those causes behave over time, has in essence created a model of its world. Now suppose an HTM is attached to a system which physically interacts with the world. You can imagine an HTM being attached to a robot, but it doesn't need to be limited to that. What is important is that the system can move its sensors through its world and/or manipulate objects in its world. In such a system, the HTM can learn to generate complex goal-oriented behavior. A brief explanation will be given here.

Figure 2a shows a system with an HTM and the ability to generate simple behaviors. The motor components of this system have built-in, "reflexive", or hard-wired behaviors. These are simple behaviors that exist independently of the HTM.

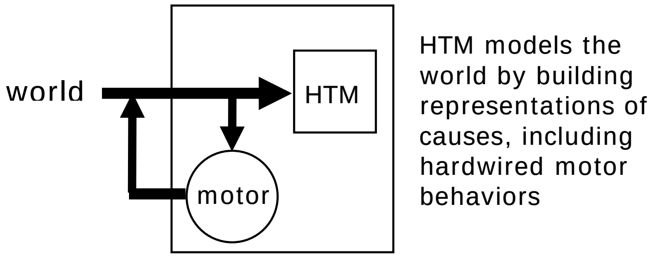
world
HTM
motor
HTM models the world by building representations of causes, including hardwired motor behaviors

Figure 2a

As the HTM discovers the causes in its world, it learns to represent its built-in behaviors just as it learns to represent the behaviors of objects in the outside world. From the HTM's perspective, the system it is connected to is just another object in the world. The HTM forms representations of the behaviors of the system it is attached to, and importantly, it learns to predict its activity. Next, through an associative memory mechanism, the HTM-based representations of the built-in behaviors are paired with the mechanisms creating the built-in behaviors themselves (Figure 2b). After this associative pairing, when the HTM invokes the internal representation of a behavior, it can cause the behavior to occur. If the HTM predicts that a behavior will occur, it can make the behavior happen in advance. Now the HTM is in a position to direct behavior. By stringing together sequences of these simple behaviors, it can create novel, complex, goal-oriented behaviors. To do this, the HTM performs the same steps it does when generating a string of predictions and imagining the future. However, now instead of just imagining the future, the HTM strings together the built-in behaviors to make them actually happen.

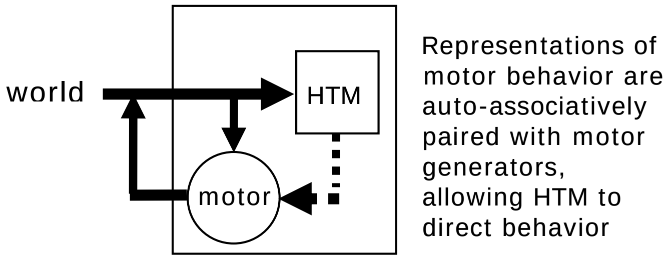
world
HTM
motor
Representations of motor behavior are auto-associatively paired with motor generators, allowing HTM to direct behavior

Figure 2b

You can observe the basics of this behavior learning mechanism in your own body. Behaviors such as eye movements, chewing, breathing, retracting an arm from a sharp object, walking, and even running are largely generated in older parts of the brain, not the neocortex. Most of the time, these behaviors are generated with little or no neocortical involvement. For example, you generally are unaware of how your jaw and tongue move when chewing, how your legs move when walking, and you are normally unaware of your breathing. However, you can consciously control your breathing and eye movements, or walk in an unusual way. When you do this your neocortex is in control. The neocortex didn't know how to do these when you were born. It had to learn how in the manner just described.

HTMs can direct the behavior of many different types of systems; they aren't limited to traditional robotics. Imagine an office building with heating and air conditioning. There are separate temperature controls on each floor. Now we attach an HTM to the building. Sensory inputs to the HTM consist of temperature sensors throughout the building as well as the settings on the temperature controls. The HTM might also get inputs representing the time of day, the number of people going in and out of the building, the current weather outside, etc. As the HTM learns, it builds a model of the building, which includes how the temperature controls behave relative to all the other things happening to and around the building. It doesn't matter if humans are changing the controls or some other computer. The HTM now uses its model to predict when things will happen, including when the temperature controls will turn on and off or be raised or lowered. By pairing the HTM's internal representations of these actions with the temperature controls, the HTM can start directing the "behavior" of the building. The HTM may be better at anticipating peak demands and therefore be better at maintaining desired temperatures or reducing consumed energy.

### Summary

We have briefly discussed the four capabilities of an HTM.

1) Discovering causes in the world
2) Inferring causes of novel input
3) Making predictions
4) Using prediction to direct motor behavior

These are fundamental capabilities that can be applied to many types of problems. Now we turn our attention to how HTMs actually discover and infer causes.

## 2. How do HTMs discover and infer causes?

HTMs are structured as a hierarchy of nodes, where each node is performing the same learning algorithm. Figure 3 shows a simple HTM hierarchy. Sensory data enters at the bottom. Exiting the top is a vector where each element of the vector represents a potential cause of the sensory data. Each node in the hierarchy performs the same function as the overall hierarchy. That is, each node looks at the spatio-temporal pattern of its input and learns to assign causes to this input pattern. Said simply, each node, no matter where it is in the hierarchy, discovers the causes of its input.

The outputs of nodes at one level become the inputs to the next level in the hierarchy. Nodes at the bottom of the hierarchy receive input from a small area of the sensory input. Therefore, the causes they discover are ones that are relevant to a small part of the sensory input area. Higher up regions receive input from multiple nodes below, and again discover the causes in this input. These causes will be of intermediate complexity, occurring over larger areas of the entire input space. The node or nodes at the top of the hierarchy represent high level causes that may appear anywhere in the entire sensory field. For example, in a visual inference HTM, nodes at the bottom of the hierarchy will typically discover simple causes such as edges, lines, and corners in a small part of the visual space. Nodes at the top of the hierarchy will represent complex causes such as dogs, faces, and cars which can appear over the entire visual space or any sub-part of the visual space. Nodes at intermediate levels in the hierarchy represent causes of intermediate complexity that occur over intermediate-sized areas of the visual space. Remember that all these causes need to be discovered by the HTM. They are not programmed in or selected by a designer.

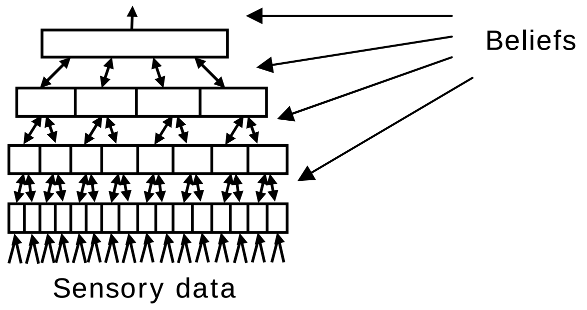

Figure 3

In an HTM, beliefs exist at all levels in the hierarchy, not just at the top level. A belief is an internal state of each node. You can think of it as a vector of scalar values where each element in the vector represents the probability that a cause is active.

Each element in the belief vector (i.e. each cause) stands on its own. Each cause can be understood and interpreted on its own and has its own meaning. In other words, the meaning of a variable representing a cause does not vary depending on what other causes might be active in the same belief vector. This does not mean the causes represented by a node are statistically independent, or that only one is active at a time. Several causes may be active at once. Representations used in HTM are different than say the representations used in ASCII codes. A particular bit in the eight bit ACSII code has no meaning on its own.

The outputs of nodes are also vectors. The outputs are similar to the belief of the node, and are derived from the belief vector. For now, we will act as if the outputs of a node are its belief. Even though this isn't completely correct, it will make it easier to describe the operation of HTMs.

With this in mind, we can say the inputs to a node are the beliefs from its child nodes. The output of a node is a belief that becomes part of the input to its parent(s). It is even correct to think of the lowest level sensory data as beliefs coming from a sensory system.

In an ideal world, there would be no ambiguity at each node. However, this does not occur in practice. One of the important properties of an HTM is that it rapidly resolves conflicting or ambiguous input as information flows up the hierarchy.

Each node in an HTM generally has a fixed number of causes and a fixed number of output variables. Therefore, an HTM starts with a fixed number of possible causes, and through training, it learns to assign meaning to them. The nodes do not "add" causes as they are discovered, instead, over the course of training the meaning of the outputs gradually change. This happens at all levels in the hierarchy simultaneously. A consequence of this learning methodology is that an untrained HTM cannot form very meaningful representations at the top of the hierarchy until nodes at the bottom of the hierarchy have undergone sufficient training.

The basic operation of each node is divided into two steps. The first step is to assign the node's input pattern to one of a set of quantization points (representing common spatial patterns of input). If a node has 100 quantization points, the node assigns a probability to each of the 100 quantization points that the current input matches that quantization point. Again, in this first step, the node decides how close (spatially) the current input is to each of its quantization points and assigns a probability to each quantization point.

In the second step, the node looks for common sequences of these quantization points. The node represents each sequence with a variable. As input patterns arrive over time, the node assigns to these variables a probability that the current input is part of each sequence. The set of these sequence variables is the output of the node, and is passed up the hierarchy to the parent(s) of the node.

A node also can send information to its children. The messages going down the hierarchy represent the distribution over the quantization points, whereas the messages going up the hierarchy represent the distribution over the sequences. Therefore, as information passes up the hierarchy, each node tries to coalesce a series of input patterns into a relatively stable output pattern. As information flows down the hierarchy, each node takes a relatively stable pattern from its parent node(s) and tries to turn it into a sequence of spatial patterns.

By assigning causes to sequences of patterns, there is a natural coalescing of time as patterns move from the bottom of the hierarchy to the top. Fast changing low-level input patterns become slower changing as they rise to the top. The opposite is also true. Relatively stable patterns at the top of the hierarchy can unfold into complex temporal patterns at the bottom of the hierarchy. The changing input patterns arriving at a node are analogous to a series of musical notes. Sequences of these notes are like melodies. If the input stream arriving at a node matches one of its learned melodies, the node passes the "name" of the melody up the hierarchy, not the individual notes. The next higher regions are doing the same thing, looking for sequences of sequences, etc. Each node predicts what note or notes are likely to follow next and these predictions are passed down the hierarchy to the child regions.

The number of levels of the hierarchy, the number of nodes at each level, and the capacity of each node are not critical to the basic theory of HTMs. Similarly, the exact connectivity between nodes is not critical as long as every two connected nodes have a clear parent/child relationship in the hierarchy. Figure 4 shows several variations of connectivity that are all valid HTMs. The design and capacity of a particular HTM must be matched to the problem being addressed and the available computing resources. A lot of effort may be required to get optimal performance. However, all configurations of HTM will work to some degree. In this regard, the system is robust.

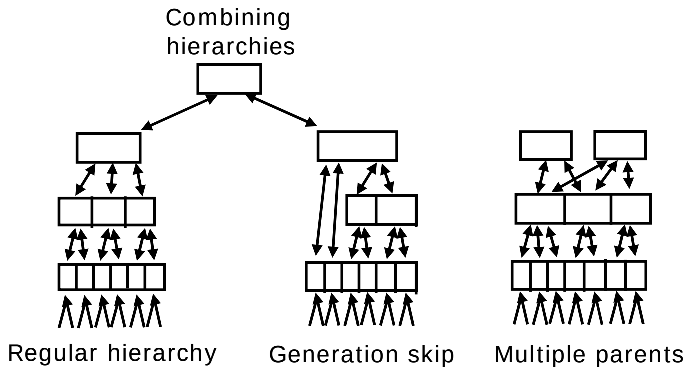

Figure 4

Given that each node in an HTM has to discover and infer causes (exactly what the entire HTM has to do albeit on a smaller scale), we are led to ask two questions. First, why is the use of a hierarchy important? That is, why is it easier to discover and infer causes using a hierarchy of nodes? Second, how does each node discover causes and do inference? After all, each node still has to solve the same problem that the entire system has to solve. We will address the first of these two questions next.

## 3. Why is a hierarchy important?

There are four reasons why using a hierarchy of nodes is important. We will touch on each one, starting with the most important.

### 3.1 Shared representations lead to generalization and storage efficiency

Many methods that have been proposed to do pattern recognition are unable to scale to large problems. Often these methods fail because the amount of memory required, and the amount of time required to train, grows exponentially as the problem space gets large, thereby making it impractical to build large systems. HTMs can require lots of training and large amounts of memory, but they do not suffer exponential problems of scale. The hierarchy in HTMs is the key to why they can scale. Causes in lower levels of the hierarchy are shared among higher-level causes, significantly reducing the amount of time and memory required to learn new causes, and providing the HTM a means to generalize previously learned behaviors to new and novel causes.

To help you understand why HTMs can solve problems that other algorithms cannot solve, we will look a bit more deeply into the difficulties these other approaches have had. We will use visual pattern recognition as our example problem because this problem has undergone much study and is familiar to many researchers. But remember, the HTM algorithm and the issues we are discussing are not specific to vision.

The most basic approach one can use to recognize objects in visual images is to store a prototypical representation for each object to be recognized. Unknown patterns are then put through a set of transformations to get them to match the prototypes. We will call this the "prototype and transformation" method. For example, if you wanted to recognize printed letters you could store a prototype image for each letter to be recognized. Given an unknown image you would first translate the unknown image in x-y coordinates to center it. Then you would perform a scaling transformation to make it the same size as the prototypes. Then you might rotate the unknown image. Finally you use some distance metric between the transformed unknown and the prototypes to determine the best match. This approach can work for simple problems such as printed character recognition but it quickly falls apart for more complex problems. For most real world objects, you can't identify a "prototypical" image. The number of possible transformations is nearly unlimited, and often there is no possible transformation that can be performed to convert an unknown into a prototype.

For example, imagine you are trying to recognize a picture of a dog. In your prototype dog image, the dog is facing left, and in your unknown image, the dog is facing right (of course you don't know this because the image is unknown). You could try a "rotation through plane" transformation on the unknown image and now it too would be facing left. However, what if you had two images, one of a Great Dane and the other of a Pekinese? A human would recognize both of these as dogs, but what kind of transformation could be used to convert one representation to the other? It is hard to say. Worse still, what if one picture was looking at the head of a dog, and the other was looking at the rear of the dog. Humans have no trouble recognizing both images as dogs, but in cases like this, there are no regular transformations that can convert the rear of a dog into the head of a dog. Vision scientists have tried many ways to overcome this problem. Ultimately they realized they have to store more than one prototype for each object to be recognized. Perhaps they need prototypes for each breed of dog and each from many different angles.

How many different examples of the objects are needed? If you could store every image of every dog you ever have seen, then supposedly it would be easier to recognize an unknown image as a dog by comparing it to all previously seen images. Of course, this is impractical. First, there are virtually an unlimited number of images that might be required for each object, and second, you still have to perform some transformations and apply a distance metric to compare a novel unknown to the many previously stored prototypes. Systems attempting to store multiple prototypes take too long to train and run out of memory. Thus, all methods of the types just described work only on simple problems. When applied to real world images, they fail. Today, general vision recognition by a computer remains unsolved.

HTMs can do large scale visual inference with practical amounts of memory and with limited processing time. HTMs do not perform any transformations as part of the inference process. A visual system built using an HTM does not rotate, translate, scale, or perform any other transformation on an unknown image to get it to "match" a prototype. Indeed, an HTM visual system doesn't even have stored prototypes in the usual sense. HTMs try to match inputs to previously seen patterns, but they do so a piece at a time and in a hierarchy.

To see how, let's start by imagining one node at the bottom of the hierarchy, a node looking at a small part of the visual input. If this node were looking at a 10x10 image patch (100 binary pixels), the number of possible patterns it might see is $2^{100}$, a very large number. Even if the node only saw a tiny fraction of the possible patterns, it couldn't store every pattern that it would likely see in its lifetime. Instead, the node stores a limited, fixed number, of patterns, say 50 or 100. These stored patterns are the quantization points. You can think of the quantization points as the most common patterns seen by the node during training. Further training will not increase the number of quantization points, but it can change them. At every moment, the node takes a new and novel input and determines how close it is to each stored quantization point. Note that this low-level node knows nothing about large objects such as dogs and cars because in a 10x10 pixel patch you can only see a small part of a big object. The causes this node can discover are limited in number and limited in complexity. Typically in a visual system, causes discovered by a node at the bottom of the hierarchy correspond to causes such as edges and corners. These causes can be part of many different higher level causes. An edge can be part of a dog, a cat, or a car. Therefore, the memory used to store and recognize low level causes will be shared among high level causes.

A node one step up in the hierarchy receives as its input the output of all its child nodes. (Assume the output of a node is just the distribution over the quantization points, ignoring for now the role of sequence memory.) This second-level node assigns quantization points to the most commonly occurring coincidences of lower level causes. This means that the second level node can learn to represent only those causes that are combinations of lower level causes. This restriction applies again and again as you go up the hierarchy. The design is such that we gain an exponential increase in memory efficiency by sharing representations in a hierarchy. The negative side of this constraint is that the system cannot easily learn to recognize new objects that are not made up of previously learned sub-objects. This limitation is rarely a problem because new objects in the world are generally formed by combinations of previously learned sub-objects.

Although sharing representations in a hierarchy makes inference possible, HTMs still may use a lot of memory. Think back to the example of recognizing a dog facing left or facing right. For an HTM-based visual system to recognize both images (i.e. assign them to the same cause), it has to be exposed to dogs or similar animals facing both left and right (and many other orientations). This requirement will make no difference at the lowest levels of the hierarchy, but it does mean that at mid-levels and higher, the HTM has to store many different combinations of low level objects and assign them to the same cause. Therefore, HTMs use a lot of memory, but the hierarchy ensures that the amount of memory needed is finite and of a practical size.

After sufficient initial training, most new learning occurs in the upper levels of the HTM hierarchy. Imagine you have an HTM that has been trained to recognize different animals through a visual sense. Now we present a new type of animal and ask the HTM to learn to recognize it. The new animal shares many attributes with previously learned animals. It might have eyes, fur, ears, tail, legs, or scales. The details of a new animal, such as it eyes, are similar or identical to the details learned previously and need not be relearned. As another example, consider that when you learn a new word you don't need to learn new letters, syllables, or phonemes. This greatly reduces both the memory and time it takes to learn to recognize new objects.

When training a new HTM from scratch, the lower-level nodes become stable before the upper-level nodes, reflecting the common sub-properties of causes in the world. As a designer of an HTM, you can disable learning for lower-level nodes after they become stable, thus reducing the overall training time for a given system. If an HTM is exposed to new objects that have previously unseen low-level structure, it will take much longer for the HTM to learn the new object and to recognize it. We see this trait in human performance. Learning new words in a language you are familiar with is relatively easy. However, if you try to learn new words from a foreign language which has novel sounds and phonemes you will find it is hard and takes longer.

Sharing representations in a hierarchy also leads to generalization of expected behavior. When exposed to a new animal, if you see a mouth and teeth, you have an automatic expectation that the new animal eats with the mouth and might bite you. This expectation might not seem surprising but it illustrates the power of shared sub-causes in a hierarchy. A new object in the world inherits the known behavior of its sub-components.

### 3.2 The hierarchy of HTM matches the spatial and temporal hierarchy of the real world

One of the reasons that HTMs are efficient in discovering causes and performing inference is that the structure of the world is hierarchical. Imagine two points in visual space. We can ask how correlated are the light values of those two points. If the points are very close to each other then their values will be highly correlated. However, if the two points are visually far apart it will be difficult to find correlations between them. HTMs exploit this structure by first looking for nearby correlations in sensory data. As you ascend the hierarchy, the HTM continues this process, but now it is looking for correlations of nearby causes from the first level, then correlations of nearby causes from the second level, etc.

The objects in the world, and the patterns they create on the sensory arrays, generally have hierarchical structure that can be exploited by the HTM's hierarchy. A body has major parts such as a head, torso, arms, and legs. Each of these is composed of smaller parts. The head has hair, eyes, nose, mouth, ears, etc. Each of these is composed of yet smaller parts. An eye has lashes, pupil, iris, and lid. At each level of the hierarchy, the subcomponents are near each other in the input pattern arriving from lower levels in the hierarchy.

Note that if you were to randomly mix up the pixels coming from a camera, then an HTM visual system would no longer work. It wouldn't be able to discover the causes in the world because it wouldn't be able to first find local correlations in its input.

HTMs do not just exploit the hierarchical spatial structure of the world. They take advantage of the hierarchical temporal structure of the world as well. Nodes at the bottom of an HTM find temporal correlations among patterns that occur relatively close together in both space and time: "pattern B immediately follows pattern A". Because each node converts a sequence of spatial patterns into a constant value, the next level in the hierarchy looks for sequences of sequences. The world is hierarchical in a temporal sense, not just spatially. For example, language is a hierarchically structured temporal sequence. Simple sounds are combined into phonemes, phonemes are combined into words, and words are combined into phrases and ideas. The temporal hierarchical structure of language may be obvious, but even vision is structured this way, at least for a system that can move about in the world. Visual patterns that are experienced sequentially in time are likely to be correlated. Patterns that are experienced far apart in time are less likely to be correlated in the raw sensory data, but may be correlated when looking at higher level causes.

Most real-world environments such as markets, traffic, biochemical reactions, human interactions, language, galaxies, etc. have both temporal and spatial structure, and both are hierarchical in nature. This structure is a natural result of the laws of physics where the forces of nature are strongest for objects that are close in space and time.

In summary, HTMs work because the world has spatial and temporal correlations that are hierarchically organized. Correlations are first found among nearest neighbors (in space and time). Each node in the hierarchy coalesces both time and space, and therefore, as information ascends the hierarchy of the HTM, the representations cover larger areas of sensory space, and longer periods of time.

When designing an HTM system for a particular problem, it is important to ask whether the problem space (and the corresponding sensory data) have hierarchical structure. For example, if you desire an HTM to understand financial markets, you might want to present data to the HTM where adjacent sensory input data are likely to be correlated in space and time. Perhaps this means first grouping stock prices by category, and then by industry segment. (E.g. technology stocks such as semiconductors, communications, and biotechnology would get grouped together in the first level. At the next level, the technology group is combined with manufacturing, financial, and other groups.). You could build a similar hierarchy for bonds, and then at the top combine stocks and bonds.

Here is another example. Suppose you want an HTM to model a manufacturing business. At the bottom of the hierarchy might be nodes that receive as inputs various manufacturing metrics. Another set of nodes at the bottom might receive as inputs marketing and sales metrics, and yet another set of low level nodes might receive as inputs financial metrics. The HTM is more likely to first find correlations among various manufacturing metrics than between the cost of advertising and the yield of a manufacturing line. However, higher up in the hierarchy, nodes can learn to represent causes global to the business, spanning manufacturing and marketing. The design of an HTM's hierarchy should reflect the likely correlations in its world.

This principle of mapping the hierarchy of an HTM to the hierarchical structure of the world applies to all HTM systems.

An interesting question is whether an HTM can have input that does not have a spatial hierarchy. For example, could an HTM have direct inputs representing words, as opposed to visual input of printed letters? Pure words do not have an obvious spatial ordering. In a sensory array where each input line represents a different word, how would we arrange the word inputs so that local spatial correlations can be found? We don't yet know the answer to this question, but we suspect that HTMs can work with such input. Our intuition is that the sensory space could have just a temporal hierarchal organization, although most causes have both. An argument for this supposition is that at the top of a hierarchy you no longer have spatial orientation, such as the top of a visual hierarchy. And yet this top node can be an input to a node that combines the top visual and top auditory beliefs. Above a certain point in the hierarchy, there is no clear spatial mapping, no topography to the representations. Biological brains solve this problem, suggesting HTMs can as well.

Sensory data can be arranged in more than two dimensions. Human vision and touch are arranged in two dimensions because the retina and skin are two-dimensional sensory arrays and the neocortex is a corresponding two-dimensional sheet. But suppose we are interested in having an HTM learn about the ocean. We can create a three-dimensional sensory array by placing temperature and current sensors at different depths for each latitude and longitude coordinate. This arrangement creates a three-dimensional sensory array. Importantly, we would expect to find local correlations in the sensory data as we move in any one of these three dimensions. We now can design an HTM where each first level node looks at data from a three-dimensional cube of ocean. The next level of the hierarchy would receive input from the low level nodes representing a larger cube of ocean, etc. Such a system would be better at discovering and inferring causes than one where the sensory data had to be collapsed onto a two-dimensional sensor array, as in a camera. Humans sometimes have difficulty interpreting high dimensional data and we go to lengths to create visualization tools to assist us. HTMs can be designed to "see" and "think" in three dimensions.

There is no reason why we have to stop at three spatial dimensions. There are mathematics and physics problems that reside in four or more dimensions, and some everyday phenomenon, such as the structure of a business, might best be analyzed as high dimensional problems. Many of the causes that humans sense via two-dimensional senses might more easily be analyzed via a higher dimensional HTM organization. High dimensional HTM is an area for exploration.

Some HTM designs will be more efficient than others at any particular problem. An HTM that can discover more causes at low levels of the hierarchy will be more efficient and better at discovering high level causes than an HTM that discovers fewer causes at low levels. Designers of some HTM systems will spend time experimenting with different hierarchies and sensory data arrangements trying to optimize both the performance of the system and its ability to find high level causes. HTMs are very robust; any reasonable configuration will work – that is, find causes – but the HTMs performance and ability to find high level causes will be determined by the node-to-node hierarchical design of the HTM, what sensory data is presented to the HTM, and how the sensory data is arranged relative to the low-level nodes.

In summary, HTMs work largely because their hierarchical design takes advantage of the hierarchical structure of the world. Therefore, a key part of designing an HTM-based system is:

1) Understanding whether the problem space has appropriate spatial-temporal structure.
2) Making sure that the sensory data is arranged to first capture local correlations in the problem space.
3) Designing the hierarchy to most efficiently exploit the hierarchical structure in the problem space.

## 3.3 Belief propagation ensures all nodes quickly reach the best mutually compatible beliefs

A connected graph where each node in the graph represents a belief or set of beliefs is commonly referred to as a Bayesian network. Thus, HTMs are similar to Bayesian networks. In a Bayesian network, beliefs at one node can modify the beliefs at another node if the two nodes are connected via a conditional probability table (CPT). A CPT is a matrix of numbers where the columns of the matrix correspond to the individual beliefs from one node and the rows correspond to the individual beliefs from the other node. Multiplying a vector representing the belief in a source node times the CPT results in a vector in the dimension and "language" of beliefs in the destination node.

A simple example will illustrate the idea. Assume we have two nodes where node A represents a belief about air temperature and has five output variables labeled "hot", "warm", "mild", "cold" and "freezing". Node B represents a belief about precipitation and has four output variables labeled "sunny", "rain", "sleet", and "snow". If we know something about the temperature, it can tell us something about the precipitation and vice-versa. A CPT matrix encapsulates this knowledge. It is fairly easy to populate appropriate values in the CPT by pairing the values of the nodes A and B as they change over time. Training the CPT and later inferring how knowledge in one node affects other nodes can even be done even when the beliefs of the two nodes are ambiguous or there is a distribution of beliefs. For example, node A may believe there is 0% likelihood it is "hot", 0% likelihood it is "warm", 20% likelihood it is "mild", 60% likelihood it is "cold", and 20% likelihood it is "freezing". Multiplying this temperature belief vector times the CPT will result in a vector representing the appropriate belief vector about precipitation.

Belief Propagation (BP) is a mathematical technique that is used with Bayesian networks. If the network of nodes follows certain rules, such as not containing any loops, BP can be used to force the entire network to quickly settle on a set of beliefs that are mutually consistent. With the appropriate network constraints, BP shows that the network will reach its optimal state in the time it takes a message to traverse the maximum length path through the network. BP doesn't iterate to reach its final state; it happens in one pass. If you force a set of beliefs on one or more nodes in a Bayesian network, BP will quickly force all the nodes in the network to reach mutually consistent beliefs.

It is helpful for the designer of HTM-based systems to have a basic understanding of Bayesian networks and Belief Propagation. A thorough introduction is beyond the scope of this paper but can easily be found on the internet or in books.

HTM uses a variation of Belief Propagation to do inference. The sensory data imposes a set of beliefs at the lowest level in an HTM hierarchy, and by the time the beliefs propagate to the highest level, each node in the system represents a belief that is mutually consistent with all the other nodes. The highest level nodes show what highest level causes are most consistent with the inputs at the lowest levels.

There are several advantages to doing inference this way. One is that ambiguity gets resolved as beliefs ascend the hierarchy. As an example, imagine a network with three nodes, a parent node and two children nodes. Child node A believes with 80% certainty that it is seeing a dog and with 20% certainty that it is seeing a cat. Child node B believes with 80% certainty that it is hearing a pig squeal and with 20% certainty that it is hearing a cat meow. Parent node C decides with high certainty that a cat is present and not a dog or pig. It chose cat because this belief is the only one that is consistent with its inputs. It made this choice even though "cat" image and "cat" sound were not the most likely beliefs of the child nodes.

Another advantage of hierarchical BP is that it is possible to make large systems that settle rapidly. The time it takes for an HTM to infer its input increases linearly with the number of levels in the hierarchy. However, the memory capacity of the HTM increases exponentially with the number of levels. HTM networks can have millions of nodes, yet still have the longest path be short, say five or ten steps.

We already have seen that many types of inference, such as hearing and touch, require time-changing patterns. Because basic belief propagation has no way of handling time-varying data, the concept of time must be added to do inference in these domains. It turns out that time is also needed to make a network self-training, even for problems such as static visual inference which at first doesn't seem to require time. The need to incorporate time will be explained in more detail later. An HTM network needs to be exposed to time-varying input, and it needs to store sequences of patterns for it to learn and solve most inference problems.

HTMs and Bayesian networks are both types of "graphical probability models". You can think of HTMs as similar to Bayesian networks but with some significant additions to handle time, self-training, and the discovery of causes.

BP also has several constraints that we don't want to adhere to in HTMs. One already has been mentioned. To guarantee that the system doesn't endlessly cycle or form false beliefs, BP prohibits loops in the network. There is evidence that for many types of networks, BP works even if there are loops. We believe this is true for HTM. In a typical HTM, each node sends its belief message to many other nodes (high fan out) and receives belief messages from many other nodes (high fan in). The high fan in and fan out reduce the likelihood of self-reinforcing false beliefs. The nodes in an HTM are also more sophisticated than in standard BP. Because of the coalescing and expansion of time-based sequences, it is difficult for a simple loop between several nodes to be self-reinforcing in a way that leads to false beliefs.

BP is a very powerful concept and a key part of how HTMs work. You should view HTMs as large Bayesian networks constantly passing beliefs between nodes in an effort to reach the most mutually compatible beliefs. The nodes at the bottom of the hierarchy are mostly driven by sensory patterns which are passed up through the hierarchy.

In an HTM, all the nodes are dynamic elements. Each node can use its internal memory of sequences combined with recent state information to predict what its next belief should be, and it passes these expectations down the hierarchy. In essence, each node can dynamically change its state based on its internal memory. So changes can originate anywhere in the network, not just at the sensory nodes. The other way that HTM nodes are dynamic is that the meaning of their beliefs changes through the learning process; as nodes discover causes, the meaning of their outputs change. This in turn changes the inputs to parent and child nodes, which also need to adjust.

In summary, there are three sources of dynamic change in an HTM. One occurs because of the changing sensory input. The second occurs as each node uses its memory of sequences to predict what will happen next and passes this prediction down the hierarchy. The third happens only during training and at a much slower time scale. As nodes learn, they change the meaning of their outputs, which affects other nodes which have to learn to adjust their meanings as well. Whenever the state of the network changes, whether due to sensory changes or internal prediction, the network quickly settles on the set of beliefs that are most mutually consistent. In human terms, what occupies our thoughts is sometimes driven by our senses and sometimes by our internal predictions

## 3.4 Hierarchical representation affords mechanism for attention

The hierarchy in an HTM provides a mechanism for covert attention. “Covert” attention is when you mentally attend to a limited portion of your sensory input. Humans can attend to a part of a visual scene. We can limit our perceptual experience to a variable size area in the center of our visual field, and we can even attend to objects that are off the center of our visual field. We can similarly attend to tactile input from one hand, the other hand, or our tongue.

Compare this to “overt” attention which is when you move your eyes, fingers, or body to attend to different objects. Many perceptual systems need a means for covert attention, if for no other reason than to attend to different objects in a complex scene.

Figure 5 illustrates the basic mechanism by which HTMs implement covert attention. Each node in the hierarchy sends beliefs to other nodes higher in the hierarchy. These connections are illustrated by small arrows. By providing a means to switch these pathways on and off, we can achieve the effect of limiting what the HTM perceives. The figure does not show the switching mechanism but highlights the active connections and nodes. The belief at the top of the hierarchy will represent the causes in a limited part of the input space.

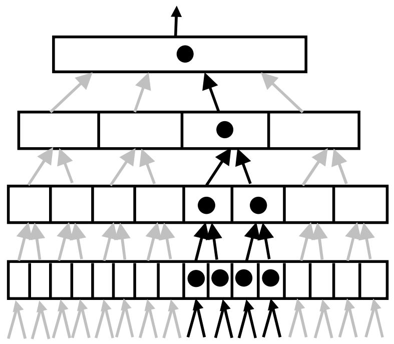

Figure 5

There are several possible ways these switches could be activated. At this time, we have verified that the basic principle works. Biology suggests there are several ways to do this including a bottom up method where strong unexpected patterns will open the pathway for attention, and a top down method driven by expectation. Further, it appears that in human brains, part of the pathway up the hierarchy is switchable for attention and part of it isn’t. Methods for switching covert attention in HTMs are being developed and tested.

In this section, we have covered the major reasons why the hierarchical design of HTMs is essential.

1) Shared representations reduce memory requirements and training time.
2) The hierarchical structure of the world (in space and time) is mirrored by the hierarchical structure of the HTM.
3) Belief propagation-like techniques ensure the network quickly reaches the best mutually consistent set of beliefs.
4) The hierarchy affords a simple mechanism for covert attention.

Now we are ready to turn our attention to how the individual nodes in an HTM work. Recall that each node has to discover causes and perform inference. Once we cover what nodes do and how they do it, we will have covered the basics of how HTMs, as a whole, discover causes and how they infer the causes of novel input.

## 4. How does each node discover and infer causes?

A node in an HTM doesn't "know" what it is doing. It doesn't know if its inputs represent light, sonar, economic data, words, or manufacturing process data. A node also doesn't know where in the hierarchy it is situated. So how can it self-learn what causes are responsible for its input? The answer is simple in theory, but a little complicated in practice.

Recall that a "cause" is just a persistent or recurring structure in the world. So a node wants to assign causes to recurring patterns in its input. There are two basic types of patterns, spatial patterns, and temporal patterns. Suppose a node has one hundred inputs and two of those inputs, $i_1$ and $i_2$ become active at the same time. If this happens often enough (far greater than by chance), then we can assume that $i_1$ and $i_2$ share a common cause. This is just common sense. If things occur together often, we can assume they have a common cause someplace out in the world. Other common spatial patterns might involve a dozen inputs that occur together. Let's say a node identifies the fifty most common spatial patterns found in its input. (There is no need to, nor is it possible to, enumerate "all" spatial patterns seen by the node). When a new and novel input pattern arrives, the node determines how close the new pattern is to the previously learned 50 patterns. The node assigns a probability that the new pattern matches each of the 50 known patterns. These spatial patterns are the quantization points discussed earlier.

Let's label the learned spatial patterns $\mathbf{sp}_1$ thru $\mathbf{sp}_{50}$. Suppose the node observes that over time $\mathbf{sp}_4$ often follows $\mathbf{sp}_7$, and it does so far greater than chance would allow. Then the node can further assume the temporal pattern $\mathbf{sp}_7$ - $\mathbf{sp}_4$ has a common cause. This is also common sense. If patterns repeatedly follow each other in time, then they are likely to share a common cause somewhere out in the world. Assume a node stores the 100 most common temporal sequences. The likelihood that each of these sequences is active is the output of the node. Those 100 sequences represent the 100 causes this node has learned.

Here then is what nodes in an HTM do. At every point in time, a node looks at its input and assigns a probability that this input matches each element in a set of commonly occurring spatial patterns. Then the node takes this probability distribution and combines it with previous state information to assign a probability that the current input is part of a set of commonly occurring temporal sequences. The distribution over the set of sequences is the output of the node and is passed up the hierarchy. Finally, if the node is still learning, then it might modify the set of stored spatial and temporal patterns to reflect the new input.

Let's look at a simple example from vision for illustration. Figure 6a shows input patterns that might be seen by a node in the first level of a hypothetical visual HTM. This node has 16 inputs representing a 4x4 pixel patch in a binary image. The figure shows several possible patterns that might appear in this patch of pixels. Some of these patterns are more likely than others. You can see that patterns which might be part of a line or corner will be more likely than patterns that look random. However, the node doesn't know it is looking at a 4x4 pixel patch and has no designed-in knowledge of what patterns are common and what they might "mean". All it sees is 16 inputs that have values between 0 and 1. It will look at its inputs over a length of time and try to determine which patterns are the most common. It stores representations for the most common patterns. The designer of the HTM designates the number of spatial patterns, or quantization points, this node can represent.

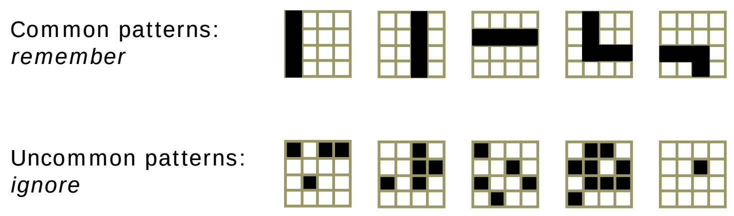

Figure 6a

Figure 6b shows three sequences of spatial patterns that might be seen by our low-level visual node. The first two rows of patterns are sequences that will likely be common. You can see they represent a line moving from left to right and a corner moving from upper left to lower right. The third row is a sequence of patterns that is unlikely to be seen by this node. Again, the node has no way of knowing which sequences will be likely, nor what the sequences mean. All it can do is try to learn the most common sequences.

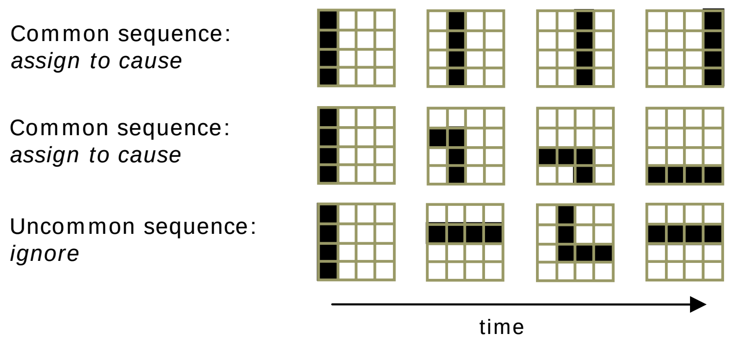

Figure 6b

Not illustrated in the visual example above is another common (but not always necessary) function of a node. If the HTM is making predictions, it uses its sequence memory to predict what spatial patterns are likely to happen next. This prediction, in the form of a probability distribution over the learned spatial patterns, is passed down the hierarchy to the child nodes. The prediction acts as a "prior" biasing the lower level nodes.

In summary, we can say that each node in an HTM first learns to represent the most commonly occurring spatial patterns in its input. Then it learns to represent the most commonly occurring sequences of those spatial patterns. The node's outputs going up the hierarchy are variables that represent the sequences, or more precisely, the probability those sequences are active at this moment in time. A node also may pass predicted spatial patterns down the hierarchy.

So far we have dealt with the simple explanation of what a node does. Now we will discuss some of the options and challenges.

## Handling distributions and real-world data

The example patterns in the previous figures are not realistic. Most nodes will receive more than 16 inputs, and consequently the input patterns seen by a node looking at real world data will be larger, messier, and will almost never repeat. In addition, the inputs are generally graded probabilities, not binary. Therefore, the node has to be able to decide what the most common spatial patterns are without ever seeing any particular pattern twice and without seeing "clean" patterns.

A similar problem exists for learning temporal patterns. A node has to determine the most common sequences of spatial patterns but has to do so looking at distributions of spatial patterns. It will never see clean data as shown in Figure 6b.

The fact that a node always sees distributions means it is generally not practical to simply enumerate and count spatial and temporal patterns. Probabilistic techniques must be used. For example, the idea of a sequence in an HTM is generally not as clean as the sequence of notes in a melody. In a melody, you can state exactly how long the sequence is and how many elements (notes) it contains. But for most causes in the world, it is not clear when a sequence begins or ends, and there are possible branchings at many elements. An analogy would be walking the streets of a familiar town. The path you take is a sequence of events. However, there is not a set path through the town. The "sequence" of streets in the town can vary as you can turn right or left at each intersection. Also, there isn't an obvious beginning or end to the sequence. Yet when you are anywhere in the town, you know it. As you walk the streets, you are confident you are in the same town.

Another problem presents itself when learning sequences. To form sequences, the node has to know when new spatial patterns arrive, that is, when the sequence elements occur. For example, when listening to a melody each new note has a sudden onset which clearly marks the beginning of a new spatial pattern. The melody is a sequence of spatial patterns where each pattern is marked by the onset of a new note. Some sensory patterns are like melodies, but many are not. If you slowly rotate an object while looking at it, there isn't a clear concept of when a new spatial pattern has arrived. Nodes in an HTM have to decide when the change in the input pattern is sufficient to mark it as a new event.

There is much prior art on how to learn spatial patterns with messy real world data. Some of these models try to precisely model parts of the visual cortex. There is less prior art on learning sequences from distributions, at least not in ways that will work in an HTM.

Numenta has developed and tested several algorithms that solve these problems. However, we believe that the algorithms, specifically those for learning sequences, will be under development for many years. There also may be variations of the algorithms, each suited to a particular sensory input or problem space.

Fortunately, most designers of HTM-based systems need not understand the details of these algorithms. They can specify the size of the nodes, the dimensions of their inputs and outputs, and the overall HTM configuration, without worrying about the details of the learning algorithms within the nodes. However, we anticipate some people, especially early on, will want to understand these algorithms and perhaps modify them. Numenta realizes we need to extend the algorithms for future uses and other researchers will want to do the same. They may want to improve their performance, experiment with variations, and modify the algorithms to tune them to particular types of problems. To facilitate this, Numenta will make the source code for our algorithms available. Numenta's software platform is designed to easily plug in new algorithms. We believe HTMs will work as long as the node's learning algorithm performs some variation of spatial quantization and sequence learning, and there may be many ways to achieve these functions.

Next we will address why time-varying inputs are necessary for learning.

## 5. Why is time necessary to learn?

Earlier we stated that an HTM could infer causes of “static” sensory patterns; the prime example being vision. (You can recognize images when they are flashed in front of your eyes.) However, we also stated that time-varying inputs are necessary to learn. Even a static vision system must be presented with a motion picture image of objects moving about in the visual field for it to learn properly, to discover causes. Why does learning require time-varying input?

We already have provided part of the answer to this question. Because each node learns common sequences of patterns, the only way a node can do this is if it is presented with sequences of patterns over time. It may be obvious that the only way to understand language, or music, or touch is by learning and recognizing sequences. However, what about static vision? Why would we need to train an HTM with moving images if, in the end, all we want to do is recognize static images? Also, what does a node do with a static pattern if it has memorized sequences? It is the answers to these questions we want to discuss now.

At the most basic level, pattern recognition entails assigning unknown input patterns to one of a number of categories. Say we have a vision system that can recognize 1,000 objects, or categories. There are an almost unlimited number of possible images we can show to our system and we hope that it will assign each unknown image to the correct category. If “horse” is one of the categories our system can recognize, there are many billions of visual patterns that you would immediately see as “horse”. We want our vision system to do the same.

Therefore, pattern recognition is a “many-to-one” mapping problem. Many input patterns get mapped to each category. We will introduce a new term to describe many-to-one mapping: “pooling”. Pooling means assigning multiple patterns to one label, that is putting them in the same pool.

Every node in an HTM must perform pooling if the hierarchy as a whole is to infer causes. Each node has to pool inputs even when just recognizing spatial patterns. We already have seen two mechanisms for pooling, although we didn’t label them as such. Spatial quantization is a pooling mechanism based on spatial similarity. In this case, we take an unknown pattern and determine how close it is to each quantization point. Two patterns that sufficiently “overlap” are considered the same. So, many possible input patterns are pooled into each quantization point. This form of pooling is a weak one and not sufficient on its own to solve most inference problems. The second pooling method is the learning of sequences. Here a node maps many quantization points to a single sequence. This method of pooling is more powerful because it allows arbitrary mappings. It allows a node to group together different input patterns that have no spatial overlap. It permits arbitrary many-to-one mappings.

Consider, for example, recognizing the image of a watermelon. The outside of a watermelon does not look at all like the inside of a watermelon. Yet if you were to see two images, one of the outside of a watermelon and one of the inside of a watermelon, you would identify both as a watermelon. There is no significant “spatial” overlap between the two images or even parts of the images, so it is in a sense an “arbitrary” mapping. How does the HTM “know” these two input patterns represent the same thing? Who tells it that input pattern A and input pattern B, which are completely different, should be considered the same? The answer is time. If you hold a cut watermelon in your hand and move it about, turn it over, and rotate it etc. you will see a continuous flow of patterns, which will progress from seeing the outside of the watermelon to seeing the inside of the watermelon to everything in between. The ultimate cause, “watermelon”, persists over time as the input patterns change.

Of course no node in the HTM remembers this entire sequence. Nodes at the bottom of the hierarchy learn sequences that are fairly short, dictated by the small area of the input pattern they can see. Nodes at the next level are more stable. They learn sequences of the sequences at the bottom level. Stability increases as patterns ascend the hierarchy. With sufficient training you will find the output of the highest level in the hierarchy remaining stable for the duration of the input sequence. Each node in the hierarchy does spatial pooling and temporal pooling. Without temporal pooling (i.e. the learning of sequences) it would be impossible for the HTM to learn on its own that the outside of a watermelon and the inside of a watermelon share a common cause.

This argument holds for almost all possible high level causes in the world, whether they are inherently temporal (such as speech, music, and weather) or whether they can be inferred statically (such as vision). Thus, time-varying inputs are necessary to learn the causes in the world.

Before we go on to discuss how an HTM can recognize a static image, we need to take a digression and discuss a problem with the above argument. There are situations where even with temporal pooling, an HTM may have difficulty learning the common cause of different inputs, at least not without some help.

### The role of supervision

Suppose you were shown pictures of food and asked to identify each picture as either a “fruit” or a “vegetable”. If shown an apple or an orange, you say “fruit”. If shown a potato or an onion, you say “vegetable”. How did you learn that apple and orange are both in the same category? You never held an apple in your hand that turned into an orange as you moved it. It doesn’t seem possible for an HTM to learn on its own, using spatial and temporal pooling, that apples and oranges are to be grouped in the same category.

The problem just described, of learning “fruits” from “vegetables”, is clear, but the same problem can occur for situations like the watermelon. It isn’t guaranteed that an HTM will always learn the desired causes by sensing sequences of input patterns. For example, what if our HTM was first trained on insides of watermelons, then on outsides of watermelons? It would naturally assign these separate patterns to two different causes. Subsequently, we train the HTM on cut watermelons where it is shown sequences moving from the inside to the outside. It is possible that the HTM will form a higher level cause (watermelon) that represents the pooling of the two lower level causes (inside red thing becoming outside green thing). Then again, it might not. It depends on the design of the hierarchy, how you train the HTM, and the statistics of the input. If it takes a long time to transition between the inside and outside views, or if there are many intermediate steps, the HTM might not pool causes as you would hope. Humans suffer the same problem when learning things like what are fruits, or who are impressionist painters. It isn't always obvious from the sensory data.

The solution to this class of problem is straightforward. Learning correct categorization, that is learning the correct causes, can be made much faster and more certain by supervising the training. In an HTM, this is done by imposing a prior expectation on the top level node(s) in the hierarchy during learning. This process is analogous to a parent saying "fruit", "vegetable", or "watermelon" as you play with your food. You might discover on your own that some foods have seeds and others don't, and therefore discover the category we call "fruit", but it will be faster and more certain if someone just tells you this fact.

A parent speaking a word such as "fruit" causes a stable pattern at the top of the auditory hierarchy (this pattern is the cause representing the sound of the word). This stable pattern is then imposed on top of the visual hierarchy making it easy to form the desired categorization of visual input.

In an HTM, we can simply impose states at the top level(s) of the hierarchy as we train.

For some HTM applications, it is best to supervise training and for others it is best not to supervise. For example, if we have an HTM that is trying to discover high-level causes of stock market fluctuations, we probably don't want to impose our prior beliefs on the system. After all, the goal of this system is to discover causes humans have not discovered already. Alternately, if we have an HTM that is being used to improve safety in a car by looking at nearby traffic, it might make sense to supervise the training on what are dangerous situations and what aren't, as opposed to letting it discover these on its own.

## Inference with static images

Now we can address the last question for this section. Say we have an HTM-based vision system. We train it with time-varying images. It forms representations of causes, either on its own or with supervision. Each node in the hierarchy pools spatial patterns in sequences. Now when a static image is presented to the system, how does it infer the cause of the image? Specifically, given that the HTM has memory of sequences of patterns, how does it infer the correct causes when it only sees a static input?

The answer is straightforward. When the static image is presented to the bottom nodes of the hierarchy, the nodes form a distribution over the spatial quantization points and from this they form a distribution over the learned sequences. With no temporal data to work with, the distribution over the sequences will be broader than it would be if temporal data were available. However, the node will form a distribution over the sequences in either case. The Belief Propagation techniques of the hierarchy will try to resolve the ambiguity of which sequences are active. It turns out that in vision, it is often possible to do so. That is, with vision, the low-level ambiguity resultant from not having temporal sensory data can still be resolved (via Belief Propagation) as the data ascends the hierarchy. If the image is sufficiently unambiguous, the top of the hierarchy is certain what cause(s) it is seeing.

This is not always the case. You can imagine a wooded scene within which is hidden a camouflaged animal. When shown this scene, you don't see the animal. However, if the animal moves, even a little bit, relative to the background, the percept of the animal jumps out. In this case, the added movement information tightens the distribution over sequences in the lower levels of the hierarchy, which is sufficient to resolve the ambiguity of the input. In fact, taking this idea to the extreme, it is possible to present a completely random field of black and white pixels, and by moving a subset of these random pixels in a coordinated way produce an "image". The spatial pattern is always random at every point in time but you still see an image because of the movement of the pixels.

## The representation of time

For some temporal patterns, the specific or relative time between the elements in the sequence is important. For example, the times between notes in a melody or the times between phonemes in spoken words, are an important part of these causes. Biological brains have the ability to learn sequences with or without specific timing information. If there are consistent time intervals in a sequence, the brain will learn them. If there are no consistent time intervals in a sequence, the brain will store the sequence without time. In the book *On Intelligence*, a proposal was made as to how the brain stores this timing information. HTMs need equivalent mechanisms to discover and infer causes that require specific timing information. Fortunately, many applications do not require this.

At this point, we have covered most of the concepts of HTMs. The next section provides answers to some commonly asked questions about HTM technology.

## 6. Questions

This section contains some common questions about HTMs and a few miscellaneous topics that have not yet been covered. It is not essential to know this material to deploy HTM-based systems, although it might clarify some of the topics already discussed. The order of the questions is not meaningful.

### How does motivation and emotion fit into the theory of HTM?

A common question we hear is, "There seems to be no role for emotion and motivation in HTMs. How can the HTM know what is important and what is not?"

In biological brains, there are several systems involved in evaluating the emotional saliency of different situations. These emotional centers are highly-evolved sub-systems that are tuned for their task. They are not located in the neocortex. As a general rule, these emotional sub-systems communicate with the neocortex in a fairly simple way. They send signals that spread broadly throughout the entire neocortex. These signals are related to rates of learning and arousal. It is as if the sub-systems are saying, "I will evaluate the emotional saliency of the current situation and when I see something important, I will tell you, the neocortex, to remember it."

HTM-based systems need a similar learning control signal. Most of the time, it will be as simple as the designer of the system deciding when the HTM should be learning and at what rate. A visual inference HTM might be trained in the laboratory under ideal conditions and later deployed with no ability to learn further. Some applications might have an automatic learning saliency system, such as a car that automatically turns on learning when the brakes are applied hard. So even though HTM-based systems have no emotions per se, the functional role of emotions related to the neocortex can be easily met.

### What happened to the CPTs? When and how are they trained?

Recall that Bayesian networks send belief messages between nodes. Further recall that CPTs (Conditional Probability Tables) are two-dimensional memory matrices that convert a belief in one node into the dimension and language of the belief in another node. The CPT allows the belief at one node to modify the belief at another node. Earlier we illustrated CPTs with the example of nodes representing temperature and precipitation. After that, we didn't explain how the CPTs were learned.

Well, we did, but in different language. In an HTM, the CPTs used in passing information from node to node going up the hierarchy are formed as a result of learning the quantization points. The quantization function itself is the CPT. By contrast, in a traditional Bayesian network the causes at each node would be fixed, and the CPT would be created by pairing instantaneous beliefs between two nodes. We can't do this in HTMs because the causes represented by each node are not fixed and have to be learned. Learning the quantization points is in essence a method of creating a CPT on the fly.

The CPTs that pass messages down the hierarchy between two nodes can be learned in the traditional way once two nodes have been trained. It also may be possible to use a transposed version of the feed-forward CPT as the feedback CPT.

There is one more difference in the CPTs as implemented in an HTM vs. a traditional Bayesian network. In a traditional Bayesian network, if three child nodes project up to a single parent node there would be three separate CPTs, one between each child and the parent. Biology suggests brains don't do it this way. In a brain, the messages from all the child nodes are mixed together resulting in a single CPT/quantization function. There are reasons to believe the biological method is superior. This is the method Numenta has implemented in its HTM framework.

### Why are the number of spatial patterns and temporal sequences in each node fixed?

There are two basic approaches one can take to learning spatial and temporal patterns in a node. The first approach is to look at the incoming patterns and enumerate them. For example, when defining spatial quantization points you could incrementally build a list of quantization points as you see new inputs. If a new input is not close to a previously seen input, you create a new quantization point. If it is close enough to a previously seen input, you assume it is the same and don't create a new quantization point. Over time, you build a longer and longer list of spatial quantization points. The same approach could be used for learning sequences. You could dynamically build a longer and longer list of sequences as new inputs arrive.

The second approach is to start with a fixed number of spatial quantization points and a fixed number of sequences. Initially, they have random meanings. As inputs arrive at the node you modify the definition of the existing quantization points and sequences. For example, you would take a new input and decide which of the initially random quantization points the new input is closest too. Then you would modify this quantization point to "move" it closer to the new input. You have to do this gradually because other nodes in the network are dependent on the output of the first node. If you rapidly changed the meaning of a quantization point or a sequence, the other nodes would be confused.

At Numenta, we have experimented with both of these methods. It is possible that both can work. We are currently focused on the latter method for a few reasons. First, we believe that the biological neocortex uses this method. Therefore, we are certain it can work for the range of problems humans can solve. Another reason is that by sticking to a fixed number of quantization points and a fixed number of sequences in each node, all learning in the system is restricted to gradual changes. The meanings of the causes at each node change slowly over time, and although other nodes need to adjust accordingly, nothing happens dramatically. The dimension of the inputs, outputs, and therefore the dimensions of the CPTs are all fixed; only the values in the memory matrices change.

Using the method where the number of quantization points and the number sequences can vary, at first seems easier, but it can lead to difficulties as the dimensions of inputs and outputs change.

## How are temporal patterns represented?

A common question is how long are the sequences that are stored in a node? Again, there are two basic ways you can approach this problem; one is to fix the length of sequences and the other is to make the sequence length dynamic. In this case, biology uses dynamic length sequences and Numenta has chosen to emulate that method, although this isn't necessarily a requirement. To give you a sense of how this works we will use a music analogy.

Imagine our node has twelve quantization points, each one corresponding to the twelve notes in a musical octave. As inputs come into the node, each of these twelve quantization points become active corresponding to what note is being played.

Next we assign ten variables to each quantization point. There are ten "C's", ten "D's", and ten "A flats", etc. for a total of 120 variables. Each of these variables represents its particular note at one location in one sequence. The node can learn any number of sequences and the sequences can be any length, with the restriction that the node only has ten of each note to work with. At one extreme, the node could learn one sequence of 120 notes long, where each note is used exactly ten times. At the other extreme, it could learn sixty sequences of length two. But no matter how many sequences it learns and no matter what the length of any individual sequence, it only has ten of each note available.

It is more complicated than this, but this analogy gives the basic flavor of how we believe neocortex stores sequences and of the approach currently favored by Numenta.

## HTMs are built of discrete regions, but biological brains are more continuous. What is the difference?

So far, HTMs have been described as a hierarchical collection of discrete nodes. The use of discrete nodes is common to Bayesian networks and indeed to all graphical probability models.

However, biology suggests brains don't work this way. In an HTM, the bottom of the hierarchy is a collection of many small nodes; in a brain the bottom of the hierarchy is one continuous region of cortex.

Obviously, nature's approach works just fine. However, we don't yet have the mathematical tools to understand continuous models as well as we can discrete models. At Numenta, we have so far stayed with the discrete node approach. We have demonstrated that this approach works and are studying whether it can be made to perform as well as the continuous approach. We ultimately expect to move to a continuous model.

## HTMs model the world. But in doing so they don't remember specific details and events. Humans have the ability to remember some specifics. How can this be done in HTM-based systems?

HTMs as described in this paper and as implemented so far by Numenta do not have the ability to remember specific events. HTMs actually throw away the details in an effort to build a model of the world. For example, if you trained an HTM-based visual system to recognize dogs, it won't remember any of the specific visual images of dogs it was trained on. There might be millions of patterns the HTM was trained on and none are remembered in detail. The process of discovering causes in the world is one of learning the "persistent" structure in the world, not one of learning any particular pattern that was only seen once.

However, humans do remember specific unitary events, especially if the event was emotionally salient. If something particularly bad or particularly good happens, you are likely to remember the details of it for a long time. For example, you almost certainly can't remember what you ate for lunch three weeks ago. However, if during that lunch you became very ill, or something bizarre happened to you, you might remember details of the lunch for the rest of your life.

In biological brains, the hippocampus is strongly implicated in forming these "episodic" memories. We believe we understand enough about the relationship between the hippocampus and the neocortex to create an equivalent "episodic" memory as an adjunct to HTMs. At this time, we are continuing to consider this capability and ultimately expect to add it to HTMs.

## What does the fovea do and do HTM-based systems need one?

Light falling on the retina at the back of the eye forms an inverted but otherwise undistorted image on the retina. However, the light receptor cells in the retina have a non-uniform distribution. There is a high concentration of these cells in the center of the retina, called the fovea, leading to a distorted representation of the image in the optic nerve and ultimately at the first level of the cortical hierarchy. As objects move in the world and as our eyes move, the severe distortion caused by the fovea appears in different parts of the visual scene.

It is surprising that our visual percept of the world has no evidence of this distortion. The theory behind HTMs explains why this is so. Like brains, HTMs form high-level representations that are invariant to distortion of their input. HTMs form high-level representations of the world as the world really is, not as it is sensed. The patterns arriving from the senses are not what we ultimately care about. It is the persistent causes in the world that we care about and that are discovered and represented by the HTM. A blind person and a deaf person form nearly identical models of the world even though they have completely different sensory systems. They discover the same causes through completely different sensory patterns. The low-level sensory data are just a means of discovering the causes in the world. Different senses, and distorted senses, all will suffice as long as they sufficiently sample the causes we care to learn.

The question we want to consider is not how recognition occurs despite the fovea. That is just a consequence of how HTMs work. The question of interest is, are foveal-type mechanisms valuable and should the principle be applied to HTM-based systems?

The fovea acts like an attention mechanism. Whatever is in the center of the visual field is over-represented and likely to be represented at the top of the visual hierarchy. Moving the eyes, in conjunction with the covert attention mechanism described earlier, allows fine detail in a scene to be perceived. You can think of it like a zoom lens on a camera.

In theory, if the entire retina had a density of receptor cells equivalent to the fovea then there would be less need to move the eyes. Perception of fine detail could be achieved with just covert attention. However, this would take a lot more cortex and more memory. So ultimately the fovea is a means of saving resources, while maintaining high acuity.

The same principle will probably be useful in some HTM applications. For example, an HTM-based system that learns about weather could have a sensory array that samples data from weather stations spread across a territory. This information could be presented to the HTM in a two-dimensional array. However, there could be a region of the sensory array that samples weather stations that are closer together. This is equivalent of a fovea. By repositioning this region of high acuity, the HTM would be able to focus on the weather detail in a particular area. Many HTM-based systems could use a similar approach.

To date, Numenta, while having thought about these ideas, has not tested all of them. It is an interesting area for experimentation.

### Are there ethical issues to be considered with HTMs?

For several years, we have discussed and explored the question of whether HTMs present any ethical dilemmas. We have consistently reached the unequivocal conclusion that HTMs do not pose any unusual ethical concerns. They present similar upside and similar potential for misuse as many other technologies such as computers or the internet. This subject is discussed in more depth in the book *On Intelligence*.

## 7. Summary and Conclusion

HTMs capture the algorithmic and structural properties of the neocortex, and as such, present the first opportunity to solve many previously unsolved problems in pattern recognition and machine intelligence. An easy way to think about HTMs is that they are suitable for tasks humans find easy to do yet computers find hard to do. However, the algorithmic properties of HTM are very flexible, and therefore, once understood, also can be applied to many problems outside of human ability.

### Capabilities

The foremost capability of HTM technology is its ability to discover the causes underlying sensory data. Causes are persistent and recurring structures in the world. The concept of “cause” captures everything from language, to physical objects, to ideas, to the laws of physics, to the actions of other people. Like humans, HTMs can learn to model all these things and more. By attaching an HTM to one or more senses, the HTM gradually and automatically builds internal representations of the causes in its world. This learning phase requires repeated exposure to sensory input over time. Causes must persist during training.

There is a hierarchy of causes in the world coexistent at any point in time, and HTMs build a corresponding hierarchy of representations. The instantaneous value of one of these representations is called a belief.

The representations formed at the top of the hierarchy are of the highest level causes. These high-level causes are ones that persist over the longest periods of time, and can span the entire sensory input space. Causes represented lower down in the hierarchy span shorter periods of time and smaller areas of the input space.

Discovering causes is a requisite first step towards later recognition, but for many applications it is an end in itself.

After discovering causes, HTMs can rapidly infer the causes underlying novel inputs. Inference is similar to “pattern recognition”. When an HTM sees a novel input, it determines not only the most likely high-level cause(s) of that input, but also the hierarchy of sub-causes. Designers of HTM-based systems can query the HTM to see what is being recognized

Each node in the network can use its memory of sequences to predict what should happen next. A series of predictions is the basis of imagination and directed behavior.

HTMs can perform several types of attention. Covert attention can be achieved by selectively disabling pathways in the hierarchy; this allows the HTM to attend to a subset of its entire input. Attentional priming can be achieved by setting a desired belief at the top of the hierarchy; this implements a directed search. Overt attention involves manipulating objects via behavior.

## Technology

Technically, HTMs can be considered a form of Bayesian network where the network consists of a collection of nodes arranged in a tree-shaped hierarchy. Each node in the hierarchy self-discovers a set of causes in its input through a process of finding common spatial patterns and then finding common temporal patterns. Unlike many Bayesian networks, HTMs are self-training, have a well-defined parent/child relationship between each node, inherently handle time-varying data, and afford mechanisms for covert attention.

Sensory data is presented at the “bottom” of the hierarchy. To train an HTM, it is necessary to present continuous, time-varying, sensory input while the causes underlying that sensory data persist in the environment. That is, you either move the senses of the HTM through the world, or the objects in the world move relative to the HTM’s senses.

Inference also is performed with time-varying input, although in some cases, such as vision, it is possible to perform inference tasks with static sensory input.

During inference, information flows up the hierarchy starting at the lowest level nodes closest to sensory input. As the information rises up the hierarchy, beliefs are formed at successively higher nodes, each representing causes over larger and larger spatial areas and longer and longer temporal periods.

Belief propagation-like techniques lead all nodes in the network to quickly reach beliefs that are consistent with the bottoms-up sensory data. Top-down predictions can influence the inference process by biasing the network to settle on predicted causes.

HTMs are memory systems. By this we mean that HTMs must learn about their world. You sometimes can supervise the learning process but you can’t program an HTM. Everything an HTM learns is stored in memory matrices at each node. These memory matrices represent the spatial quantization points and sequences learned by the node.

Being a new technology, there are many advances ahead in our understanding of HTMs. For example, we need to improve our ability to measure and define the capacity of an HTM. We need to develop useful heuristics for how best to specify hierarchies to match particular problems. We have a lot to do in order to improve our training methods. And although we have developed algorithms for spatial quantization and time-based pooling, we are certain they can and will be improved. There will be many years of advances and refinements as we learn how to use this technology.

The first implementation of the Numenta HTM platform is on standard Linux-based computers. The platform tools will run on anything from a single CPU to clusters with many CPUs. We anticipate various forms of custom hardware will ultimately be developed specifically for HTMs but this is not necessary today.

## Implications

HTM is a powerful new computing paradigm that may ultimately equal the importance of traditional programmable computers in terms of societal impact and financial opportunity.

One of Numenta’s goals is to maximize the beneficial impact of HTM technology. The approach we are taking to achieve this is to create a platform that makes it easy for engineers and scientists to experiment with the HTM technology, develop HTM-based applications, and to create exciting business opportunities based on HTM. In addition to documenting the platform and tools, Numenta will make available the source code for many parts of the platform. This source code access should allow developers to better understand how Numenta’s tools work and provide an opportunity and financial incentive to extend the platform.

Because HTMs model the large-scale structure and function of the neocortex, Numenta’s tools also should be useful in the fields of psychology, education, psychiatry, and neuroscience as a way to explore the capabilities of healthy humans and to better understand mental disease.

We only have just started to develop this compelling new paradigm of intelligent computing based on HTM technology. We are putting in place the platform and theoretical foundation today. We expect to make great progress over the coming years in understanding the limits of HTMs, how they will scale, how they will perform, and what problems they will solve.

Most importantly, we look forward to building a community of people working on this technology, and applying it to a broad range of challenging real-world problems.