Techniques for the Emergence of Meaning in Machine Learning (ML)

Luca Dellanna
Email: Luca@luca-dellanna.com
Website: www.luca-dellanna.com
Twitter: @dellannaluca

A current bottleneck that prevents Machine Learning (ML) from being successful outside of a few restricted fields such as chess playing and highway driving is its impairment in appropriately using context to infer the meaning of what it is observing. This paper describes techniques to allow ML systems to derive meaning from context, derived from how the human cortex works.
In particular, this paper shows how the multiplication a horizontal vector representing a Sparse Distributed Representation (SDR) of patterns in sensory data by a vertical vector representing a SDR of patterns in context data followed by a pattern recognition operation on the resulting matrix results in the integration of relevant context and in the output of data containing meaning.

State of the art

Currently, the field of ML is dominated by two approaches: Deep Learning (DL) and Hierarchical Temporal Memory (HTM).
HTM depends on a hierarchy of pattern recognition operations, as shown in Figure 1. This approach is interesting because a hierarchy of units being able to apply pattern recognition operations to the output of the units below them is part of how our brain works. The operations of pattern recognition are applied by spatial poolers, as described in (Hawkins, Ahmad, & Dubinsky, 2011).

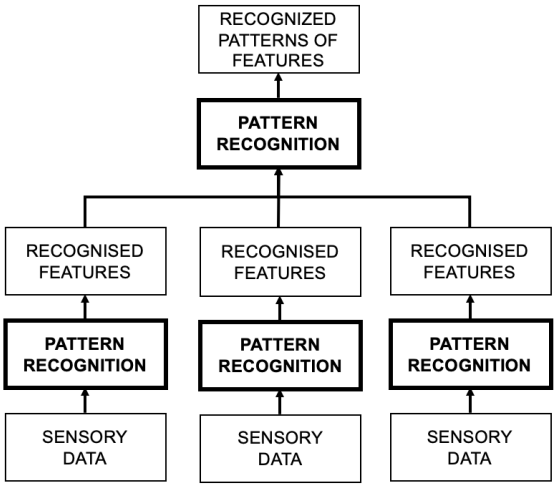


Figure 1. Architecture based on hierchical pattern recognition only

This approach fails to integrate context. Context is defined as any information which is not contained into the sensory data being considered at the moment and yet necessary to grasp the full meaning of what is described by such sensory data. For example, a machine examining one single written sentence could be able to discern the words used and the concepts that correspond to such words, but it cannot know its full meaning. For that, it needs context - who wrote that, who that person is, what that person is going through in his or her life, where that person wrote it, and so on.
The addition of temporal poolers, described in (Hawkins et al., 2011), allows to properly integrate temporal context (i.e., to recognize sequences) but does not contribute to the processing of other forms of context.
The easiest (but wrong) way to integrate context would be to feed it to the machine as if it were sensory data, as shown in Figure 2.

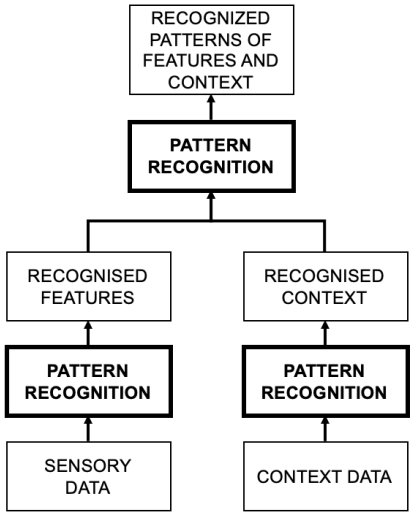


Figure 2. Architecture using context as content

However, this approach is not effective. By treating context as if it were sensory data (i.e., content), it is not able go beyond simple first-order correlations between context and content.
A good way to integrate context is simply to mimic how our brain integrates context at the level of the neurons in the neocortex. The technique is shown in Figure 3. The next two sections will respectively describe the precise processes taking place in our brain which inspired this approach and how to practically implement it in a ML system.
Producing all possible permutations of a content pattern (i.e., features) with a context pattern, whether relevant or not, might seem inefficient. Conversely, it is very effective, for three reasons:

- As the next section will explain, that's exactly the process that takes place in our brain.
- Whereas single items representing contextualized features contain meaning, single items representing sets of context and features (where context and features are at the same level) do not contain meaning, unless there is a 1:1 relationship between a given context and a given feature (but this is seldom the case).
- The last operation depicted in Figure 3, the pattern recognition on all tentative contextualized features, takes care of weeding out the irrelevant ones. Relevancy, in an intelligent machine, is a proxy for predictive power. Something is relevant if it predicts. In this particular case, one contextualized feature is relevant if it predicts the activation of a pattern of contextualized features at the next step. The contextualized features which predict patterns observed over and over will create links (ie, synapses) with the patterns they contributed to; conversely, the contextualized features which do not predict consistently a pattern of contextualized features (and therefore are irrelevant) will not form such links.

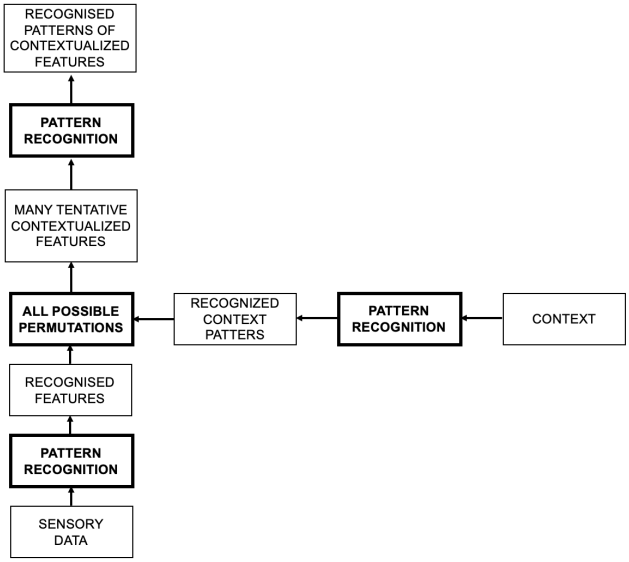


Figure 3. Architecture using context to expand content

The human neocortex

The representation of information as patterns. The cortex of the brain can be divided into regions. A region is a group of hundreds of thousands of neurons which receive input from a common set of sources. Regions represent the information flowing through them as patterns of neurons firing. A snapshot of the neurons firing at a given moment in a region[1] represents the information passing through that region at that moment.
Contrary to common beliefs, there is not a 1:1 correspondence between neurons and concepts. A single concept is not represented by a given single neuron firing, but by a given pattern of neurons firing. *The technical name of such patterns is "Sparse Distributed Representations"* (Hawkins et al., 2011).

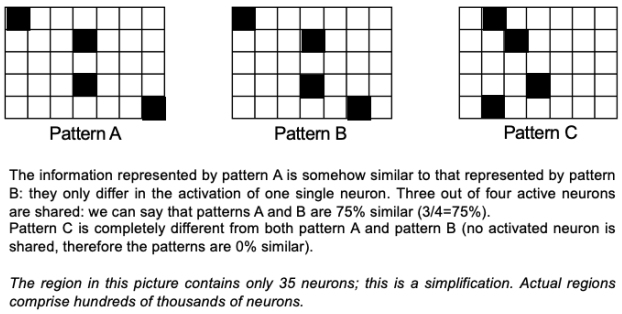
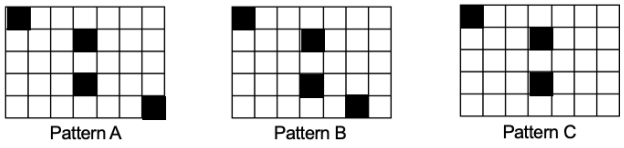


Figure 4. The human brain uses Sparse Distributed Representation to represent information

Similar information is represented by similar patterns. For example, a pattern representing "my golden retriever Harry" and one representing "a Labrador" will share a lot of active neurons. This makes sense because we want the information to be resistant to noise and to partial occultation, as described in the figurative example below.
In the sentence "similar information is represented by similar patterns", the meaning of "similar information" depends on which information enters the region. In a region receiving visual inputs, for example, two patterns which have similar representations will represent two objects which look similar. In a region tasked with processing olfactory inputs, two patterns which have similar representations will represent two

[1] At least in Layer 2, 3 and 4. Patterns of firing in Layers 5 and 6 probably represent information derived from the information represented by layers 2, 3 and 4.



Imagine that pattern A represents "my golden retriever Harry" and pattern B represents "a Labrador". Those two dogs share a lot of commonalities, so it makes sense that they are represented similarly.
The property that "patterns with similar meaning share many activated neurons" helps completing the patterns also in presence of noise or partial occultation. Let's imagine that pattern C is the result of seeing a paw and some golden furry behind a table. Even with partial information, my brain is able to recognize that what I'm seeing is either a Labrador or my golden retriever, because both of those patterns share 75% of their active neurons with pattern C.

Figure 5. In Sparse Distributed Representations, similar patterns of activation represent similar information

objects which smell similarly. In a region tasked with processing motor orders, two patterns which have similar representations will represent two similar movements. In a region tasked with processing abstract concepts, two patterns which have similar representations will represent two concepts with a similar meaning.

Pattern recognition. The regions in our brain use spatial pooling for recognizing patterns in their inputs (Hawkins et al., 2011). Spatial pooling broadly works as follows:

1. Neurons in a region are organized in minicolumns, each one comprising about 80 neurons.
2. Neurons in a given minicolumn share the same receptive field; in other words, a given subset of the neurons in the regions hierarchically below project their axons to the proximal dendrites of all the neurons in the minicolumn.
3. When a given percentage of the neurons in its receptive field activates at the same time, the minicolumn activates. A pattern in the input has been recognized. (The facts that concepts are represented through patterns and that patterns follow the rule "patterns that share a lot of active neurons represent similar features" mean that if a minicolumn activates when it recognizes an input activating at least, say, 80% of its dendrites, then it effectively both "pools together" similar patterns and is noise-resistant).
4. The active minicolumn inhibits nearby minicolumns.
5. The synapses in the active minicolumn update: synapses between active axons from the input and the neurons in the minicolumn get potentiated, whereas the ones between inactive axons and the neurons in the minicolumn get weakened. This causes learning.

Temporal contextualization. Regions of our neocortex use temporal context to attribute different meanings to sensorial stimuli which appear identical but are part of a different sequence. For example, a spatial pooler in a region tasked with processing auditory stimuli might recognize the pattern of stimuli representing the musical note "G". Temporal pooling will make the region represent such stimulus in different ways depending on the melody it is part of (i.e., depending on the musical notes that preceded it). In order to achieve this, our brain uses a process called temporal pooling (Hawkins et al., 2011). Temporal contextualization broadly works as follows:

1. Neurons in a minicolumn have, between others, two types of dendrites: proximal and basal. Proximal dendrites get activated by the axons of the neurons in the regions hierarchically below; basal ones, by those of the neurons in the same region.
2. In other words, basal dendrites help the neuron recognize patterns of activation in the other minicolumns of the region. This allows neurons to learn sequences such as "if minicolumns X, Y and Z are active then I'm likely to become active next". This is useful, for example to learn sequences of musical notes (melodies).
3. If enough of its basal dendrites are active, the neuron enters a "predictive state". Technically, it depolarizes just enough to "fire more easily" if depolarized through the proximal dendrites too, but not enough to fire if the proximal dendrites aren't active.
4. When a minicolumn activates, if any of its neurons is in predictive state, then only such neurons activate. If no neuron is in predictive state, then all neurons activate. Which neurons activate represent the sequence in which the concept represented by the minicolumn appears, and its position in it. (If all neurons activate, it means that no particular sequence has been memorized yet, thus all sequences are possible).
5. When a neuron switches from predictive to active, the basal synapses which received an action potential from an axon get reinforced and those which didn't get weakened. This ensures learning.

The process described above is a simplified[2] version of the temporal pooler described in (Hawkins et al., 2011). The output of pattern recognition represents the patterns - or features - recognized in the input.
As mentioned previously, there is no 1:1 relationship between a single active neuron and a feature of the object being

[2]In the version presented here, the concept of "predictive mode" is skipped. This concept is only relevant to allow for stability in the output - a property which, while valuable, is not necessary for the understanding of how functional contextualization works. For working implementations of the system described in this paper, it is recommended to use the full version of the temporal pooler descibed in(Hawkins et al., 2011).

observed. Because information are represented using Sparse Distributed Representations (SDRs), patterns of active neurons represent patterns of features. A property of SDRs is that SDRs that share a lot of active neurons represent similar inputs. There are two advantages in using SDRs: first, it confers noise resistance; second, it allows to represent never-seen-before features when they get observed. Imagine that a brain learned the SDR representation of an apricot and the SDR representation of an apple. Let's say that this brain never saw a peach before. The first time it is shown a peach, the pattern recognition will produce an SDR which shares some active neurons with the SDR of the apricot and some active neurons with the SDR of the apple. Immediately our brain will be able to have an idea of what that unknown object - the peach - is, of how much it weights, of how to interact with it and so on. **Using this property of SDRs, upon seeing a new object, our brain does not have to create a new category from scratch - it automatically processes a category which sharing properties with the two existing ones. It does not have to create downstream connections to the higher areas - it can use those that are already there from the two previously-learnt SDRs.**

Functional contextualization. Functional contextualization allows to attribute to a feature recognized during pattern recogntion context about its meaning or function.
Pattern recognition uses the proximal dendrites and temporal contextualization uses the basal ones. However, neurons have a third type of dendrite: apical ones. Whereas proximal dendrites connected to the axons of the neurons in regions hierarchically below, and basal ones to the axons of neurons in the same region, apical ones (also) receive projections from Layer 1 (L1) which in turn receives projections from other areas of the brain (Rubio-Garrido et al., 2009). I hypothesize that one of the functions of L1 is to collect information about the context - the output of the other regions, representing short-term memory or the processed sensory signals - and to relay it to the apical dendrites of the other neurons in the other layers. Such information is then used for functional contextualization.
Functional contextualization is similar to temporal contextualization: they both use information external to the content data to represent the content data differently depending on the context. Those two forms of contextualization work similarly because apical dendrites function similarly to basal ones: they help with the depolarization of the neuron, but can hardly make the neuron fire on their own. **The purpose of both functional and temporal contextualization is to ensure that morphologically similar outputs of pattern recognition which are semantically different (because having different context) are also represented morphologically different before entering the next region. In other words, they allow to integrate meaning inferred from context into the content of a sensorial observation.**

Functional contextualization comprises of the following steps:

1. If enough of its apical dendrites are active, the neuron enters a "predictive state". Technically, it depolarizes just enough to "fire easier" if receiving an action potential through the proximal dendrites, but not enough to fire if the proximal dendrites aren't active.

2. When a minicolumn activates, if any of its neurons is in predictive state, then only such neurons activate. If no neuron is in predictive state, then all neurons activate. Which neurons activate represent the context in which the concept represented by the minicolumn appears, and its position in it. (If all neurons activate, it's because no previously learnt context pattern was recognized and therefore all context patterns are possible.)

3. When a neuron switches from predictive to active, the apical synapses which received an action potential from an axon get reinforced, and those which didn't get weakened. This ensures learning.

The purpose of the functional contextualization. Pattern recognition takes some patterns as inputs and its role is to take those which are morphologically similar and to represent them as morphologically identical.
Functional contextualization takes those patterns that came out from the spatial pooler and are morphologically similar but semantically different (because they have different context, as perceived through the apical dendrites) and makes them morphologically different, so that the next spatial pooler to process them (in the upper region) does not conflate them.
In other words, functional contextualization allows to use in an efficient way information from outside the receptive field to prevent the region from outputting two identical patterns when they have a different meaning or function. It's not a system that works every time - as demonstrated by the many errors we humans make when trying to identify an object, especially when we can't interact with it - but it works well enough, and for sure better than if it wasn't present.

Interpreting the results of functional contextualization. The output of pattern recognition represents a set of features recognized in the input (lines, musical notes, etc. - the kind of features it recognizes depends from the input the region receives, for example a region receiving auditory inputs will recognize musical notes).
The output of temporal contextualization represents a feature in a sequence. For example, "the musical note G in the melody BCBGFG".
The output of functional contextualization represents a feature in a context - a contextualized feature. For example, "the musical note G in a sad song". Or, "a cup in the kitchen",

"my cup", or "a cup which contains hot coffee".
However, functional contextualization is the result of pairing features our brain is observing to contexts we are experiencing - creating all possible combinations, regardless of whether they are meaningful or not. For example, if a person observes for the first time a coffee cup and he happens to be in a room with red walls, the region processing those two elements is likely to activate a neuron contributing to the representation of the contextualized feature "coffee cup in a red room". Is this data element meaningful? In other words, does the fact that the coffee cup is in a red room confer it any meaningful properties, some that should lead to a prediction which should influence our decisions? The region having processed the input does not know and does not care. It will be the task of the next region to determine whether this piece of information is meaningful.
The next region in the hierarchy will perform again pattern recognition to the set of contextualized features. Those which are meaningful will lead to the activation of minicolumns. As a result of the last step of pattern recognition (learning) the synapses between the axons of the neurons representing the contextualized feature and the proximal dendrites of the activated minicolumn will reinforce. Those contextualized features which are not meaningful will instead not lead to the activation of a same minicolumn with consistency and will therefore be weakened, to the point of perhaps disappearing.
In other words, the functional processing unit of the human neocortex - a group of neurons which is able to decode meaning from sensory input - is not a single region but the union of a region plus the part of the following one which is producing pattern recognition.

Implementing contextualization in a ML system

A Machine Learning system implementing the techniques described in the previous section is made of a set of nodes called "regions" organized in a loose hierarchy made of levels such as the one depicted in Figure 3. The figure is greatly simplified; an actual ML system will likely have a few dozens or hundreds of such units, organized on a few levels.
Regions have two sources of data: content and context. For a given region, content data is the current output of the region or regions just below it in the hierarchy. Context data is the current output of some or all of the regions in the system.
Regions at the bottom level receive content data from sensors or from interfaces to the external world. Each region at the lower level has to receive a different subset of the available sensory data, so that they process different aspects of it. Regions at the upper layers instead receive content data from the level(s) just below them.
Both content data and context data should take form of a Sparse Distributed Representation (SDR). An SDR has the following properties: it contains a fixed number of bits, only a few of its bits are active at a given time (say, about 2%), and SDRs at a given location which share similar active bits represent similar content. If the data entering a region is not in SDR form, it has to be converted into it. SDRs and how to encode data in them have been described extensively in (Hawkins et al., 2011).

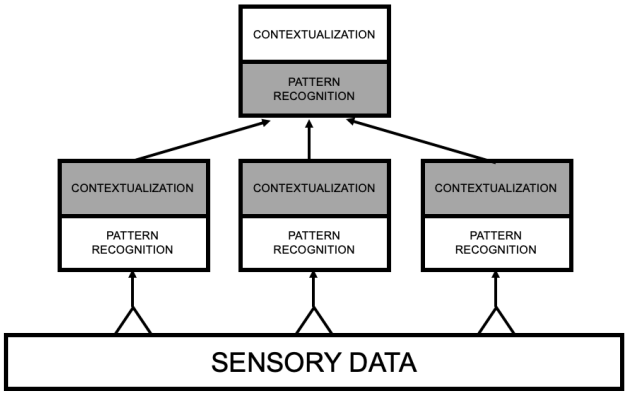


Figure 6. Architecture of a system implementing contextualization

Pattern recognition. This section describes how to perform pattern recognition using a spatial pooler (Hawkins et al., 2011). However, other algorithms can probably be used, as long as they: accept an SDR as input, produce an SDR as output, the output has a considerably lower number of bits than the input and it implements some form of learning.

Initialization:

1. A set of "region" objects is created (say, 20).

2. Regions are mapped in a hierarchical configuration such as that of 6.

3. For each region, an array of "subordinate regions" is created, containing the references to the regions hierarchically below it whose output is used as content data. A region can be subordinate to more than one region. Some regions will have no subordinates (those at the lowest level of the hierarchy).

4. Each region will use as its input the output of its subordinate regions. If a region has more than one subordinate, its input will be the concatenation of the outputs of its subordinate regions. If a region has no subordinates, then its input will be a subset of the sensory data (which has to be encoded as an SDR).

5. For each region, a set of "minicolumn" objects are created (say, 1000).

6. For each minicolumn, an array of “synapse” objects is created (say, 100). Each synapse has two properties: a decimal “permanence value” and a pointer to a bit of the input to the region (for example, one synapse could have a pointer to the 3rd bit of the input). The permanence value is always a decimal comprised between 0 and 1. If any of the following operations would set it below 0 or above 1, it is set to 0 and 1 respectively.

7. For each minicolumn, each synapse is assigned a permanence value which is a random decimal chosen with a standard distribution with mean 0.7.

8. For each minicolumn, each synapse is assigned a reference to a given bit of the input. Usually, minicolumns tend to have synapses pointing to bits which are not too far from each other, though it is not a hard rule.

9. For each minicolumn, a “boost” property is created, with an initial value of 0.

Processing:

1. For each minicolumn, a “score” is computed, which is the count of its synapses whose permanence value is at least 0.7 and which point to an active bit of the input. The score is then multiplied by 1 plus the minicolumn’s boost value. If the minicolumn’s score is bigger than 75 then the minicolumn is marked as “potentially active”. If it’s lower than that, the minicolumn is marked as inactive.

2. Each potentially active minicolumn tries to inhibit the nearby others. There are many ways to do this. One is described in (Hawkins et al., 2011) and another is as follows: for each minicolumn, a set of similar active minicolumn is formed by getting the potentially active minicolumns for which at least 70 of its synapses point to input bits which are also pointed to by the synapses of the minicolumn being processed. Then, the synapses in this set are ranked by descending score, and the top, say, 10% are marked as active. The others are marked as inactive. The values of this step are chosen so that, in general, only about 2% of the minicolumns are active at any given time. This process ensures that only the minicolumns representing the most relevant patterns activate.

3. For each active minicolumn, for each of its synapses, the permanence value is increased by 0.1 if the synapse was active and it is decreased by 0.1 if it wasn’t.

4. The boost value of each active minicolumn is set to 0, and the boost value of each inactive minicolumn is increased by 0.1.

The coefficients 1000, 100, 0.7, 75, 70%, 10%, 2% and 0.1 have been set based on individual experience. Depending on the sensor and environment the ML system operates in, other coefficients might produce better results.

Temporal contextualization. This section describes how to perform temporal contextualization using a simplified[3] version of the temporal pooler described in (Hawkins et al., 2011).

Initialization:

1. A spatial pooler is set up (as described in the previous section).

2. Each minicolumn is assigned an array of “neuron” objects, say 100.

3. For each neuron, an array of “synapses” objects is created (say, 100). Each synapse has two properties: a decimal “permanence value” and a pointer to a bit of the input to the region (for example, one synapse could have a pointer to the 3rd bit of the input). The permanence value is always a decimal comprised between 0 and 1. If any of the following operations would set it below 0 or above 1, it is set to 0 and 1 respectively.

4. For each neuron, each synapse is assigned a permanence value which is a random decimal chosen with a standard distribution with mean 0.7.

5. For each minicolumn, for each neuron, each synapse is assigned a reference to a cell in another minicolumn.

Whenever a minicolumn of the spatial pooler becomes active (per the process described in the section named "pattern recognition"), the following steps are performed:

1. For each neuron of the minicolumn, a score is computed as the count of its synapses which point to active minicolumns. If the score is higher than, say, 50 then the neuron is set as active. The threshold has to be chosen so that, in general, after the learning phase of the system is complete, about 2% of the neurons in the pooler are active at a given time.

2. If an active minicolumn has no active neuron, then all its neurons become active.

[3] In the version presented here, the concept of "predictive mode" is skipped. This concept is only relevant to allow for stability in the output - a property which, while valuable, is not necessary for the understanding of how functional contextualization works. For working implementations of the system described in this paper, it is recommended to use the full version of the temporal pooler descibed in(Hawkins et al., 2011).

3. For each active neuron, for each synapse, its permanence value is increased by 0.1 if it is pointing to an active minicolumn and it is decreased by 0.1 otherwise.

4. The output of the pooler is a Sparse Distributed Representation (SDR) of its neurons, where each active neuron is represented by a 1 and each inactive one by a 0.

Functional contextualization (without temporal contextualization). This method is used in machines whose sources of data do not contain regular sequences. Most often, they do. In such cases, it is suggested to use the method described in the next section, "functional contextualization with temporal contextualization". This section has been written mostly for the clarity of the reader, to allow him to understand how functional contextualization works alone, before implementing it together with temporal one.

Initialization:

1. A spatial pooler is initialized (as described in the previous section).

2. A second spatial pooler is initialized. It will be referred to as “context spatial pooler”. It is initialized similarly as the content spatial pooler, with the difference that its input is not the output of the regions hierarchically below, but the concatenated input of some or all of the regions in the system. (HTM practitioners will consider the context spatial pooler as containing one cell per minicolumn - it does not perform any temporal pooling.)

3. Each minicolumn of the content spatial pooler is assigned an array of “neuron” objects, say 100.

4. For each neuron, an array of “synapse” objects is created (say, 100). Each synapse has two properties: a decimal “permanence value” and a pointer to a bit of the input to the region (for example, one synapse could have a pointer to the 3rd bit of the input). The permanence value is always a decimal comprised between 0 and 1. If any of the following operations would set it below 0 or above 1, it is set to 0 and 1 respectively.

5. For each neuron, each synapse is assigned a permanence value which is a random decimal chosen with a standard distribution with mean 0.7.

6. For each minicolumn, for each neuron, each synapse is assigned a reference to a minicolumn in the context spatial pooler.

Whenever a minicolumn of the content spatial pooler becomes active, the following steps are performed:

1. For each neuron of the minicolumn, a score is computed by counting its synapses which point to active minicolumns in the context spatial pooler. If the score is higher than, say, 50 then the neuron is set as active. The threshold has to be chosen so that, in general, after the learning phase of the system is complete, about 2% of the neurons in the pooler are active at a given time.

2. If the minicolumn spatial pooler has no active neuron, then all its neurons become active.

3. For each active neuron, for each synapse, its permanence value is increased by 0.1 if it is pointing to an active minicolumn and it is decreased by 0.1 otherwise.

4. The output of the pooler is a Sparse Distributed Representation (SDR) of its neurons, where each active neuron is represented by a 1 and each inactive one by a 0.

Functional contextualization (with temporal contextualization). In the previous section, a system able to perform functional contextualization but not temporal one was described. In this section, a system being able to perform both is described instead.

Initialization:

1. A spatial pooler is initialized (as described earlier). It will be referred to as “content spatial pooler”.

2. A second spatial pooler is initialized. It will be referred to as “context spatial pooler”. It is initialized similarly as the content spatial pooler, with the difference that its input is not the output of the regions hierarchically below, but the concatenated input of some or all of the regions in the system.

3. Each minicolumn is assigned an array of “neuron” objects, say 100.

4. For each neuron, two arrays of “synapse” objects are created (say, 100 each). The first array is referred to as “basal synapses” and the other as “apical synapses”. Each synapse has two properties: a decimal “permanence value” and a pointer to a bit of the input to the region (for example, one synapse could have a pointer to the 3rd bit of the input). The permanence value is always a decimal comprised between 0 and 1. If any of the following operations would set it below 0 or above 1, it is set to 0 and 1 respectively.

5. For each neuron, each synapse is assigned a permanence value which is a random decimal chosen with a standard distribution with mean 0.7.

6. For each minicolumn, for each neuron, each basal synapse is assigned a reference to another minicolumn in the content spatial pooler, and each apical synapse is assigned a reference to a minicolumn in the context spatial pooler. (The context spatial pooler does not perform any temporal contextualization - it can be considered as a spatial pooler with one cell per minicolumn).

Whenever a minicolumn of the spatial pooler becomes active, the following steps are performed:

1. For each neuron, a score is computed by counting its synapses which point to active minicolumns. If the score is higher than, say, 100 then the neuron is set as active. The threshold has to be chosen so that, in general, after the learning phase of the system is complete, about 2% of the neurons in the pooler are active at a given time.

2. If an active minicolumn has no active neuron, then all its neurons become active.

3. For each active neuron, for each synapse, its permanence value is increased by 0.1 if it is pointing to an active minicolumn and it is decreased by 0.1 otherwise.

4. The output of the pooler is a Sparse Distributed Representation (SDR) of its neurons, where each active neuron is represented by a 1 and each inactive one by a 0.

Discussion

The purpose of the functional contextualization. Pattern recognition takes some patterns as inputs and its role is to take those which are morphologically similar and to represent them as morphologically identical.
Functional contextualization takes those patterns that came out from the spatial pooler and are morphologically similar but semantically different (because they have different context, as perceived through the apical dendrites) and makes them morphologically different, so that the next spatial pooler to process them (in the upper region) does not conflate them.

Compression and expansion. Information enters the Machine Learning system through sensors (or interfaces to the outer world). Each such sensor has a given bandwidth and describes information observed along a given dimension. For example a small camera might use 20000 bits to represent the images of the observed objects at any given moment of time.
As the information enters the first "region" of the ML system, it undergoes an operation of pattern recognition which compresses such information. This happens because the region uses less minicolumns than bits in the input (the number of minicolumns necessarily represents the number of bits in the output of the pattern recognition information). For example, 1000 minicolumns might be used: in such case, the input will be of 20000 bits and the output of 1000 bits: compression occurred.
Moreover, such compression takes place along the dimension of the input. The output of pattern recognition, while smaller, still represents visual features (information along the visual dimension).
The output of pattern recognition then undergoes contextualization. Each active minicolumn activates some of its neurons. Following the previous example, in which visual information encoded using 20000 bits gets processed by a region with 1000 minicolumns and 100 neurons per minicolumn (100000 neurons in total), the output of the region is the configuration of active neurons: it then comprises of 100000 bits).
Each region compresses information along the dimensions the input of the spatial pooler was encoded into, and expands it across the dimensions of the inputs of the temporal and functional poolers - in other word, time and context.
The genius of our brain is that by constantly changing the dimensions along which the information is encoded as it passes from region to region, it allows it to progressively integrate meaning (across the new dimensions). Moreover, following the expansions by compressions (made with pattern recognitions across the last dimension) it ensures that only the information relevant across that dimension is retained. In other words, it ensures that the higher regions encode information *only* across the dimensions they need to for taking decision. An exceptionally efficient and self-organizing system.

Conclusion

This paper explained the mechanisms used by the human brain to integrate context into meaningful interpertation and described how to implement them in machine learning systems.
In particular, it showed how the alternation of compression on one dimension followed by an expansion on another - those dimensions becoming more and more abstract - leads to the emergence of meaning.
Moreover, it clarified one common misunderstanding: one brain region does not constitute a processing unit of the human brain; the union of the process of expansion in one region (the useapical and basal dendrites to select the neurons that fire in an active minicolumn) and of the process of pattern recognition in the following one does.

References

Hawkins, J., Ahmad, S., & Dubinsky, D. (2011). Hierarchical temporal memory including htm cortical learning algorithms.

Rubio-Garrido, P., et al. (2009). Thalamic input to distal apical dendrites in neocortical layer 1 is massive and highly convergent.