Extracting and exposing predictive cortical columns for selective attention

David Eriksson

Independent researcher
affiliated to Royal Institute of Technology, Stockholm, Sweden

Abstract

In several studies it has been proposed that the neocortex consists of functional modules. In this paper we will motivate how these modules could be relevant for attention and efficient learning. We present a biological plausible algorithm composed of one input layer with modules or columns that is connected to an output layer with forward and backward-projections, which in turn are inspired by those found in the neocortex. Since the algorithm is purely associative it exhibits one-shot learning capabilities. The algorithm is tested with a simple experiment where it finds the predictive columns for different tasks.

Introduction

In several studies it has been proposed that the neocortex consists of functional modules or groups of neurons. The neurons in such modules may share some property or may respond to related features (Rolls and Treves, 1998, Tanaka, 2002, Roland and Zilles, 1998). These modules are thought to be the smallest functional module in the cortex ranging from a group of mini columns (for example the direction sensible column in visual cortex) to blobs, patches and fields. If such module is a functional module it means that the activity of the neurons inside the module are more dependent of each other than to neurons outside the module. This inter-dependency will naturally decrease as the size of the functional module increases and as a result the neurons will change to a cooperative mode inside the module (Roland, 2003). However, those smaller inter-dependent modules are possibly relevant in a general way for attention and efficient learning. But before we go into that we will motivate the need of attention and define some terms.

The idea behind the algorithm presented in this paper is that the attention to a particular neuron (and column in this paper) is made based on statistical and informational aspects of the environment rather than implicit processing constraints of the observer (Dayan et al, 2000). For example some stimuli in the environment may be important to attend to in order to handle particular task (say A). However, for another task (say B) the same stimuli may not bring any information of how to handle task B and may even contribute with conflicting information since that stimuli was used to handle another task (task A). To conclude we may say that the stimuli mentioned above is *predictive* for task A but *un-predictive* for task B (Dayan et al, 2000). In this paper we will also call a neuron predictive if it represents a predictive stimuli.

Now we will motivate how a cortical module may be relevant for attention. Suppose that one particular neuron (say A) is predictive for a particular task. Also suppose that this neuron is located in an inter-dependent module of neurons in neocortex. For simplicity we will assume that the neurons in this module are mutual exclusively dependent of each other. Now suppose that another neuron (say N) inside the module is active. Then from the inter-dependency we know something about the activity of neuron A, namely that A is inactive. Since neuron A is predictive, the activity of neuron N is predictive as well. From there on it is probable that every neuron inside this inter-dependent group is predictive and therefore the whole module should be attended for the task. Therefore we will from now on call such a module *predictive*. Note that such predictive modules are supposed to be independent of each other.

By assigning a whole module to be predictive we believe that it is possible to get learning more efficient. To see what we mean we can return to the scenario where neuron A (and probably N too) was predictive. Now suppose that the conditions for handling the task have changed, i.e. neuron A is inactive and neuron N is active. Then suppose that the correct way of handling the task under the new conditions is presented. As a result, if the whole module is attended (which it should be since neuron A in the module is predictive and since the task is the same), the associations from neuron N to the presented way of handling the task will be learned more efficient.

In this paper we will present an algorithm that can identify and expose the predictive modules or columns. Since the algorithm is purely associative it exhibits one-shot-learning capabilities. It extracts information by observing pair of successive states. A state could be composed of, for example, external stimuli, internal motivation such as what task to handle and what actions to execute and what reward that has been retrieved or predicted. The pair of states can be the components in a sequence as well as the state-action pair and state-reward pair used in reinforcement learning. The first and following state in the state-pair is targeted for the input and output in a feed forward neural network. It is important that the input is composed of inter-dependent columns.

By adding a backward projection that mirrors the forward projection it is possible to "generate" the input-state based on the output-state (Geoffrey et al, 1995). Together with the inter-dependency of the input columns the back-projection will generate resonance in those input columns that are crucial for predicting the following state given the current state. Resonance in a column leads to exposure of the corresponding column. As a result, when a column is exposed, other parts in the state, for example the task, can be associated to that column, i.e. associating the task to the predictive column.

Method

The algorithm will be tested with a simple choice-experiment where it should learn to focus on the length of a stick when the task is to reach a banana and it should learn to focus on the visibility of the stick when the task is to signal. A conflict in this example is that a bright stick will work when the task is to signal, whereas a bright stick will not work when the stick is short and the task is to reach a banana.

The choice-experiment is simulated with state-pairs as follows. First the algorithm is presented with a task (get a banana or to signal) and a stick (which can be long or short and bright or dull) and the stick constitutes the first state. Then the algorithm is told if the stick is appropriate ("will work" or "will not work") for the given task and this corresponds to the following state. This means that there are eight combinations:

- ***Task**: Banana. **State-pair**: (<u>first-state</u>: long and bright, <u>following-state</u>: "will work").*
- ***Task**: Banana. **State-pair**: (<u>first-state</u>: short and bright, <u>following-state</u>: "will not work").*
- ***Task**: Banana. **State-pair**: (<u>first-state</u>: long and dull, <u>following-state</u>: "will work").*
- ***Task**: Banana. **State-pair**: (<u>first-state</u>: short and dull, <u>following-state</u>: "will not work").*
- ***Task**: Signal. **State-pair**: (<u>first-state</u>: long and bright, <u>following-state</u>: "will work").*
- ***Task**: Signal. **State-pair**: (<u>first-state</u>: short and bright, <u>following-state</u>: "will work").*
- ***Task**: Signal. **State-pair**: (<u>first-state</u>: long and dull, <u>following-state</u>: "will not work").*
- ***Task**: Signal. **State-pair**: (<u>first-state</u>: short and dull, <u>following-state</u>: "will not work").*

All these combinations are shown to the algorithm (choice). The difficulty in the experiment is then to expose the predictive column for each task (Banana => Length of the stick and Signal => Brightness of the stick). In order to make the experiment more realistic the two tasks (banana, signal) and the different properties of the stick (long, short, bright, dull) are presented according to a probability distribution: P(A)=S, P(B)=1-S, P(A1) = A, P(A2) = 1-A, P(B1) = B and P(B2) = 1-B. As a result the output becomes: P(1) = R = S+(1-S)B and P(2) =1-R= S(1-A) + (1-S)(1-B). The inter-dependence in the columns in this example is mutually exclusive. For example the length of the stick is represented with a column consisting of two mutually exclusive units: long or short.

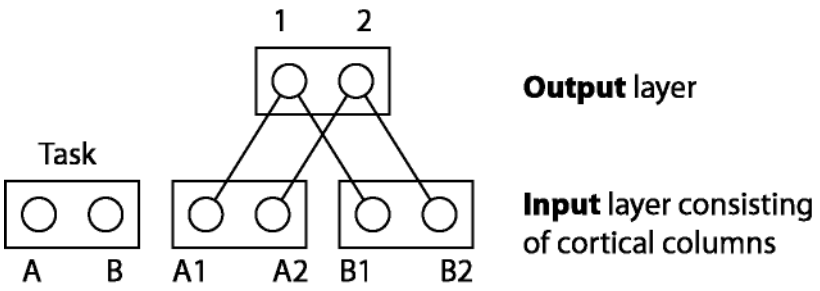

Figure 1. Architecture for the algorithm. A and B correspond to get a banana or to signal. A1 and A2 correspond to long or short stick. B1 and B2 correspond to dull or bright stick. 1 and 2 correspond to "will work" or "will not work". For example if the task is A (get a banana) then the activity of A1 and A2 is crucial: if A2 (short stick) is active then the correct output is 2 ("will not work").

As mentioned in the introduction the algorithm relies on forward and backward projections. For each forward connection there is one corresponding backward connection. Since there are associatively modifiable synapses in both directions the weights will be almost symmetrical. This is crucial since it enables activity at the output neurons to "generate" the activity at the input neurons and as a result building up a resonance in the predictive columns. Now there are two parts of the algorithm that is essential to understand. First we must estimate and motivate what the strength of the weights will be. The forward and backward weights will be treated as the same. Then we will motivate why there will be a resonance in the predictive column and competition in the un-predictive column. All motivations will primary be based on the Choice-experiment (figure 1).

The goal with the calculation of the weights is to show under what circumstances the weights are optimal, i.e. the weights are optimal when the major connections are (A1,1), (A2,2), (B1,1) and (B2,2) as figure 1 shows. In order to calculate the weights we have used a normalised learning rule (notice that the weight-calculation is biologically plausible since it is solely based on local information):

$$w_{A1->1} = \frac{P(A1 \cap 1)}{P(A1)P(1)} = \frac{P(1 \mid A1)P(A1)}{P(A1)P(1)} = \frac{P(1 \mid A1)}{P(1)} = \frac{S+(1-S)B}{R}$$

$$w_{A2->1} = \frac{P(A2 \cap 1)}{P(A2)P(1)} = \frac{P(1 \mid A2)P(A2)}{P(A2)P(1)} = \frac{P(1 \mid A2)}{P(1)} = \frac{0+(1-S)B}{R}$$

As we can see the term that differ is S. This is logical since S describes how many times we will get task A (get a banana) and therefore we have to attend to A1 and A2 (the length of the stick). As a result, if we never get this task (S=0) then the weights to A1 and A2 will be equal and then we do not know what to do given that task. The normalization has the property that the weight gets stronger between those units that have fired less frequently. An interpretation of this is that the weights between the characteristic neurons get stronger. As a result the storage capacity for correlated patterns will increase significantly. This has also been done by means of a non-monotonic transfer function (Morita M., 1993) and anti-Hebbian learning (Bogacz R. and Brown M., 2002).

Now we will motivate why there will be resonance in the predictive column and competition in the un-predictive column. If the weights are correct then imagine the following scenario. Output 1 is active and that will generate an input with active A1 and B1 according to the weights. Now since output 1 is active and we only present the choice with correct input-output-pairs, at least one of the columns must have the first unit of the real input active (A1 or B1). Suppose that the task is A then at least A1 must be active. The crucial thing is that B can be anything (B1 or B2) and when B2 is active the generated input will not be the same as the real input. Since the columns are inter-dependent, there will be a competition between the generated input and real input. As a result the B column will be more uncertain or weak. On the other hand the other column, A, cannot have contradicting inputs since A is important in order to solve the task correctly. Therefore the generated input will be in resonance with the real input. To conclude, B can (it does not necessarily need to be weaker since the activity in B could be B1 and that would not have contradicted the generated input) be weaker if the task only requires column A to be correct. In other words if the current task requires column A to be correct then column A will have the strongest activity and as a result the current task can be associated with column A.

To implement the forward and backward projections we have used a neural network called BCPNN (Bayesian confidence propagation neural network) (Sandberg A. et. al, 1999). The reason for this is two fold. First the algorithm handles inter-dependent columns (the units are mutually exclusive). Second, the calculation of the weights fulfils the criteria that the weights between two neurons should be independent of the number of times that each neuron has been active.

The complete BCPNN algorithm for our experiment is listed below (figure 2). It is essentially an auto-associative neural network due to the forward and backward projections. The units are divided into one input group with two columns, where each consists of two units, and one output group with one column that also consists of two units. However, there are no lateral connections inside the output- and input layers (The lateral weights will be explicitly set in order to eliminate their influence). The bias has been omitted since the output activity should only be dependent on the input and should not be dependent on how active the neuron has been statistically. Indices (i) denote which column and (i‘) denote which unit inside the column. The column for the output (“will work” or “will not work”) is (i=3) and the columns for the input (which can be long or short and bright or dull) are (i=1 and 2).

$$P^{n}_{ii'jj'} = P^{n-1}_{ii'jj'} + t_L^{-1}((1-l_0^{\,2})x^{n-1}_{ii'}x^{n-1}_{jj'} + l_0^{\,2} - P^{n-1}_{ii'jj'})$$

$$P^{n}_{ii'} = P^{n-1}_{ii'} + t_L^{-1}((1-l_0)x^{n-1}_{ii'} + l_0 - P^{n-1}_{ii'})$$

$$w_{ii'jj'} = \frac{P_{ii'jj'}}{P_{ii'}P_{jj'}}$$

$$y^{0}_{ii'} = 0, x^{0}_{ii'} = 0$$

$$y^{n}_{ii'} = y^{n-1}_{ii'} + t_R^{-1}\left(\left[\sum_{j}^{H}\log\left(\sum_{j'}^{U_i} w_{ii'jj'}x^{n-1}_{jj'}\right)\right] + a_{ii'}\log(I_{ii'}) - y^{n-1}_{ii'}\right)$$

$$x^{n}_{ii'} = \frac{e^{y^{n}_{ii'}}}{\sum_{k\in U_i}^{U_i} e^{y^{n}_{ik}}}$$

The network is driven by the activity-vector: $I_{ii'}$, in which the first four elements are input and the last two elements are output. During 100 iterations all the correct combinations are fed to the network (of course the task is not), which yields about 12 iterations per combination if

S=0.5, A=0.5 and B=0.5. More precisely each combination will run for $100*(S)^{banana}*(1-S)^{signal}*(A)^{long}*(1-A)^{short}*(B)^{bright}*(1-B)^{dull}$ iterations. These 100 iterations are then repeated three times. The parameters used in this experiment are: $t_L=500$, $l_0=0.0001$, $t_R=1$, $a_{ii'\in output}=1$ and $a_{ii'\in input}=0.1$.

Since the degree of resonance in each column should correspond to the degree of exposure, a measure of the resonance (or anti-competition) in each column must be defined. We have chosen to simply set the exposure of each column to $abs(x_{i1}-x_{i2})$, but other more biologically plausible variants are possible. The measured correlation between each task and the exposure of each column that we have used is exactly the normalised correlation used for the weights in the neural network. All task-column combinations result in a square correlation matrix in which the rows correspond to the task and the columns correspond to the columns. This correlation matrix is averaged during the 100 last iterations. The correctness is measured as the relative difference between the diagonal elements (banana⇔length and signal⇔colour) and the other elements. The results for various probabilities of S, A and B are plotted in the result section.

Result

Below we have plotted the correctness when we vary the balance of the two tasks (Only "signalling"-combinations are presented for S=0 and an equal number of "signalling" and "get banana" for S=0.5) and the balance between the properties of the stick (A=B, so when A=B=0, then only the short and dull stick-combinations are presented to the algorithm).

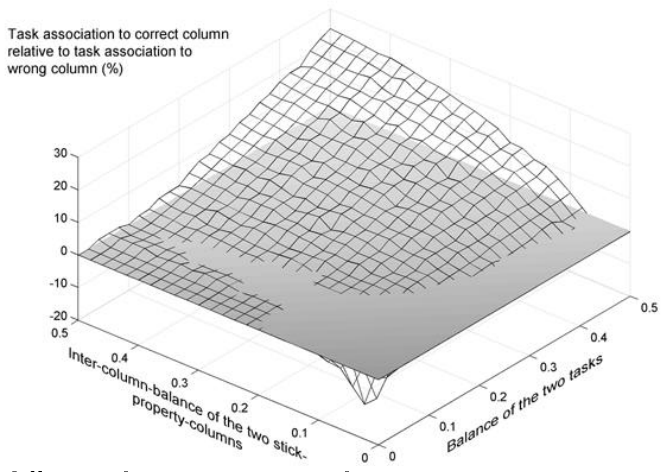

Figure 3. Percentage in difference between correct and incorrect associations. Since we cannot plot three-dimensional functions with ease, we had to set A=B for this figure.

In the figure we can se that the maximum difference is 25% between the predictive and un-predictive column. The maximum occurs when all combinations are presented for an equal period of time, i.e. when both the stick property balance and the task balance are 0.5. The difference is more sensitive to how balanced the tasks are than the balance of the two properties of the stick. It is only when the stick property balance is less than 0.075 or the task balance is less than 0.1 that the algorithm exposes the un-predictive column more than the predictive.

Discussion

We have now presented a biologically plausible algorithm that exposes the predictive columns more than the un-predictive columns. Since the algorithm make use of direct association in order to get competition between generated input and real input, it exhibits one-shot learning capabilities. One improvement to the algorithm could be by exploiting the association between task and predictive column during learning. That would probably give even better quantitative results.

Furthermore it would be interesting to se how the algorithm manages more complex cases, i.e. more inputs, columns, outputs and tasks. Initial theoretical studies of this kind suggest that the difference between the wrong weight and correct weight will increase as the size of the columns increase. Also the correct weight should always be larger than the wrong weight. The motivation is that the wrong weight ($w_{A2->1}$ in the choice-experiment) could only be encouraged when the corresponding column is un-predictive (the (1-S)B-term in the choice experiment). In contrast the correct weight ($w_{A1->1}$ in this case) could be encouraged in both cases (the S- and (1-S)B-term in the choice experiment), i.e. when the corresponding column is un-predictive as well as predictive.

If the weights are correct for a more complex case then it is probable that the whole algorithm should work. This is because it generally holds that the competition could only occur un-predictive columns. In other words the predictive columns will always generate resonance and as a result the predictive columns will always be exposed.

Finally we would like to consider the analogy between our approach and the two terms *uncertainty* and *unreliability* (Dayan et al, 2000). First, one important component in *uncertainty* is the novelty of the stimuli: if the stimuli is novel, then it should have a larger impact of the weight-update. This is exactly the effect of the normalizing terms in the weight-update used in the BCPNN-algorithm. Finally the *unreliability* is similar to the degree of competition that can emerge in un-predictive columns.

References

Bogacz, R & Brown, MB (2002). *An anti-Hebiian model of familiarity discrimination in the perirhinal cortex*. Computational Neuroscience Proceedings.

Dayan, P, Kakade, S & Montague PR (2000). *Learning and selective attention*. Nature, **3**, 1218-1223.

Hinton, GE, Dayan, P, Frey, BJ & Neal, RM (1995). *The wake-sleep algorithm for unsupervised neural networks.* Science, **268**, 1158-1160.

Morita, M (1993). *Associative memory with nonmonotone dynamics,* Neural Networks, **6,** 115-126.

Roland, PE & Zilles, K (1998). *Structural divisions and functional fields in the human cerebral cortex*. Brain Research Reviews, **26**, 87-105.

Roland, PE (2003). Personal communication.

Rolls, ET & Treves, A (1998). *A Neural Networks and Brain Function*. Oxford University Press.

Sandberg, A, Lansner, A, Peterson, KM & Ekeberg, Ö (1999). *A Palimpsest Memory based on an Incremental Bayesian Learning Rule*. Neurocomputing, **32-33**, 987-994.

Tanaka, K (2002). *Zeroing in on the higher brain functions*. Riken Research Highlights News. Brain Science Institute. Laboratory for cognitive brain mapping. **251**.