Unsupervised Learning of Relations[†]

Matthew Cook[1⋆], Florian Jug[2], Christoph Krautz[2⋆⋆], and Angelika Steger[2]

[1] Institute of Neuroinformatics,
University of Zurich and ETH Zurich, Switzerland
cook@ini.phys.ethz.ch
[2] Institute of Theoretical Computer Science,
ETH Zurich, Switzerland
{fjug,ckrautz,asteger}@inf.ethz.ch

Abstract. Learning processes allow the central nervous system to learn relationships between stimuli. Even stimuli from different modalities can easily be associated, and these associations can include the learning of mappings between observable parameters of the stimuli. The data structures and processing methods of the brain, however, remain very poorly understood. We investigate the ability of simple, biologically plausible processing mechanisms to learn such relationships when the data is represented using population codes, a coding scheme that has been found in a variety of cortical areas. We require that the relationships are learned not just from the point of view of an omniscient observer, but rather the network itself must be able to make effective use of the learned relationship, within the population code representations. Using a form of Hebbian learning, local winner-take-all, and homeostatic activity regulation away from the periphery, we obtain a learning framework which is able to learn relationships from examples and then use the learned relationships for a variety of routine nervous system tasks such as inference, de-noising, cue-integration, and decision making.

1 Introduction

One of the key properties of the brain is the ability to notice and learn the relations between inputs in an unsupervised manner [1, 2, 3, 4]. It is believed that this is achieved by modifying the structure [5, 6, 7] and the dynamics [8, 9, 10] of biological neural networks, for example through the plasticity of synapses or other neural processes [11, 12, 13].

Phenomenologically, the ability of brains to discover relationships between otherwise independent events has been known since the pioneering work by Pavlov [2]. His work on dogs showed that a neutral stimulus (the ringing of a bell) can be induced to elicit an associated reaction (production of saliva) by ringing a bell every time the dog gets food. After training, the association

[†] The authors are listed in alphabetic order.

[⋆] Supported by EU Project Grant FET-IP-216593.

[⋆⋆] Supported by ETH Research Grant ETH-23 08-1.

between the bell and salivation may be due to the food representation being activated by the bell representation.

In the following decades, it became clear that the ability to learn relations between different sensory inputs is in fact omnipresent in our brains. As an example of an inference task, when we hear a sound, we can use the audio input to estimate (*inference*) the visual location of the corresponding visual input. This process is continually maintained by learning mechanisms: if we wear prism glasses that shift the visual input, it is possible to re-learn the correspondence. Or, given uncertain visual and audio cues about the location of a stimulus, we can combine these cues (*cue-integration*) to get a better estimate of the location. If visual and audio cues differ so much as to be inconsistent, for example due to light or sound being reflected so as to appear to have a different source, then we simply base our position estimate on the stronger of the two competing inputs (*decision*).

Biological data shows that neural populations, regions and areas encode specific sensory, motor, and cognitive modalities (see e.g. [14, 15] and contained references). Connected regions exchange signals and thus influence their mutual activity [14], and simulations have also exhibited such interactions in networks with hand crafted connectivity [9, 15, 16]. Simulations such as these have shown how inference, de-noising, cue-integration, or decision tasks can be performed on input received from different visual, motor, or other sources, for complex problems like coordinate transformations.

Given the presence of such abilities in the brain, the question that immediately arises is *how* the brain implements them. A major step in this regard was achieved by Zipser and Andersen [17], who trained an artificial neural network with simulated biological data using the backpropagation algorithm [18]. In their network, hidden nodes developed gain field properties [17]. While their result shows that neural networks are able to learn such tasks in principle, the learning strategy they used seems unlikely to be the one used in our brains: the back-propagation algorithm is a supervised learning scheme, using an externally generated error signal, and it is generally considered to be biologically implausible [19, 20].

Our goal is to exhibit the ability to learn arbitrary relationships using biologically plausible learning. We present a model that can learn the relationships between inputs in an *unsupervised* way (that is, without externally supplied error signals). In fact, our model is purely based on biologically motivated building blocks like population coding, Hebbian learning, and homeostatic activity regulation. After learning the relationship, our model can use the learned relation to improve its population code representations: the network will produce population codes for missing inputs based on supplied inputs (*inference*), will smooth noisy population codes (*de-noising*), will adjust population codes to be more consistent with each other (*cue-integration*), and will choose between alternative population code representations when faced with inconsistent data (*decision*). A key feature of our network is that its dynamics do not have to be modified from outside in order to switch between these tasks, or even to re-learn a relationship when it changes.

2 The Network and Its Dynamics

In this paper we consider the following network (see Figure 1) to demonstrate how relations between two sets of parameters X and Y can be learned. The network consists of two populations, A and B, consisting of n rate coded units each.

2.1 The Network

The units in A get input from an external source X by point-to-point connections, i.e., each unit in A receives input from exactly one unit in X and each unit in X sends input to exactly one unit in A. Similarly, a second input Y, is connected to B by point-to-point connections. X and Y are supposed to encode one single scalar value each. To realize this encoding we use what is known as population coding, see e.g. [21]. Intuitively, this means that each unit in X has one preferred value and that its firing rate depends on how close its preferred value is to the actual value. In Figure 1 we illustrated this encoding by representing X and Y by two (noisy) population codes.

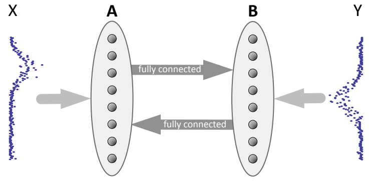

Fig. 1: Projection diagram of the sample network discussed in the text: two populations with bidirectional connectivity. Labeled ellipses represent populations of neural nodes. Dark gray arrows depict directed, full connectivity, light gray arrows indicate point to point connections used to feed population coded input into the network. Blue dots show one possible input of this kind.

The units within each of the populations A and B are laterally interconnected such that each population is effectively a soft winner-take-all circuit [22]. The connection weight $w_{i,j}$ between units i and j is defined as:

$$w_{i,j} = \gamma \cdot e^{-\frac{1}{2}(d(i,j)/\sigma)^2} - \delta\,. \tag{1}$$

The distance d between i and j is $d(i,j) = \min\{|i-j|, n-|i-j|\}$. In order to avoid boundary effects we let the distance measure wrap around. (Note that this implies that we identify the smallest value $v_{\min}$ and the largest value $v_{\max}$

encoded by the population code. In the case $v_{\min} = 0$ this means that we perform calculations modulo $v_{\max}$.)

Besides being laterally connected, the units in A and B are also interconnected. Effectively we connect each unit $i \in A$ to all units $j \in B$ and vice versa. The initial connection weights $w_{i,j}$ are set to values chosen randomly in $[0, 1]$.

Learning the relations between the inputs X and Y is done by adapting the connections between the populations A and B using a Hebbian learning rule [23].

2.2 The Dynamics

We simulate our network over discrete time steps. At time t the rate-coded units in A and B each have a real-valued activity level a, which we denote with a superscript as a^t. At each time step t each unit j updates its activity level a_j^t. This update is influenced by (i) the activities of the neurons in the same populations (via the lateral connections), (ii) the activities of the units in the other population (via the connections between A and B), and (iii) a homeostatic activity regulation term h_j^t (used to keep the activity level of each unit roughly constant over time).

We explain the details of the update below. Here we just outline the interplay between the main ingredients. The lateral connections implement soft winner-take-all dynamics (WTA) [22]. Essentially, they are used to “clean-up” noisy input. The weights $w_{i,j}^t$ between the populations A and B are updated by a Hebbian learning (HL) scheme, eventually encoding the learned relationship. The homeostatic activity regulation (HAR) [24] forces units to regulate themselves so that each unit is active roughly a given proportion of the time. This makes sure that every unit is used, and that each unit is used in moderation.

It is worth noting that the presented components work on quite different time scales. The WTA dynamics operate on a short time scale, allowing the network to converge quickly. HAR and HL operate on a much longer time scale, averaging over a much larger sample of inputs. A sketch of how Hebbian learning (HL), soft winner-take-all (WTA) and homeostatic activity regulation (HAR) play together is illustrated in Figure 2.

Hebbian Learning. The update of the weights $w_{i,j}^t$ depends on (i) the activities a_i^t and a_j^t of units i and j at time t, and (ii) two global parameters α_l and α_d. The Hebbian learning rate α_l regulates the speed at which connections get learned and is usually set to the same value as α_d, the Hebbian decay rate. The weights are updated according to:

$$w_{i,j}^{t+1} = (1 - \alpha_d) \cdot w_{i,j}^t + \alpha_l \cdot a_i^t \cdot a_j^t \,. \tag{2}$$

To speed up the running time of simulations it suffices to do these updates only after the WTA converged.

Homeostatic Activity Regulation. We use the following update formula for the homeostatic activity terms:

$$h_j^t = -c \cdot (\bar{a}_j^t - a_{\text{target}}) \,, \tag{3}$$

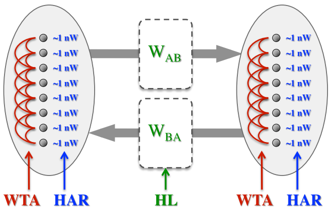

Fig. 2: The presented mechanism is a combination of three strategies. Synaptic connections between areas are controlled by Hebbian learning (HL). Local connections within an area support soft winner-take-all (WTA) dynamics, so nearby units within an area exhibit similar activity patterns. Homeostatic activity regulation (HAR) within each unit modulates the Hebbian learning so that a unit does not become permanently active or inactive, but maintains a desired average activity level.

where a_{target} is a parameter and

$$\bar{a}_j^t = (1-\omega)\bar{a}_j^{t-1} + \omega a_j^t \tag{4}$$

for some additional parameter ω defining the decay rate of the averaging.

Neural Units and Update Dynamics. At each discrete time step t each unit j updates its activity level a_j^t. To compute it we first take the weighted sum over the activity levels of all units connected to unit j. This includes both the lateral connectivity within the population as well as the connections coming from other populations. This sum is corrected by the homeostatic activity regulation term h_j^t. Finally we apply a non-linear function θ that restricts the activity level to the range $[0, 1]$. Formally the update rule is defined as

$$a_j^{t+1} = \theta(h_j^t + \sum_{i \in \mathit{\Gamma}_j^{\text{in}}} w_{i,j}^t \cdot a_i^t)\,, \tag{5}$$

where $\mathit{\Gamma}_j^{\text{in}}$ is the set of units connected to unit j, and

$$\theta(x) = \frac{1}{1 + e^{-m(x-s)}} \tag{6}$$

and m and s are parameters that determine the slope and the shift of $\theta(x)$.

3 Results

In the following we present our experimental results. Note that the network dynamics introduced in the previous chapter remains unchanged throughout all experiments that we present. In order to switch from one task to another we only change the input fed to the network.

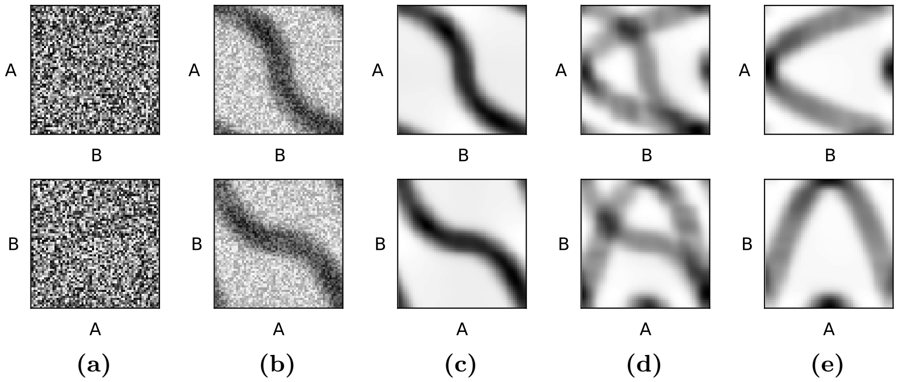

Fig. 3: The time course of learning and relearning in the sample network. Each plotted subfigure shows a snapshot of the connection weights W_{AB} (top row) and W_{BA} (bottom row) for different times during learning and relearning. The weights are color coded (black for strong, white for weak connections). **(a)** Initial random weights. **(b)** Weights captured during learning. **(c)** Weights after the relation $y = x^3$ was learned. **(d)** Weights captured during relearning. **(e)** Weights after the relation $y = x^2$ was learned.

3.1 Learning and Relearning

In order to feed interpretable input we have set a preferred stimulus p_i for each node i in A and B. To encode the value v in X (or Y) we set the input x_i (y_i) for node i in A (B) according to:

$$x_i(v) = C \cdot e^{-(v-p_i)^2/(2\sigma^2)} . \tag{7}$$

This enables us to feed arbitrary scalar values to populations A and B. If these values satisfy any functional relation, the network will learn the relationship hidden in a sequence of input pairs. Note that the weights between populations A and B are constantly changing over time. If after a certain relationship was learned the input changes and a different relation is presented the weights will change to reflect the new relation. Figure 3 shows how the weight matrices W_{AB} and W_{BA} change in the course of learning and relearning.

3.2 Inference Tasks

After the network has learned a relation we can then also omit one of the inputs and infer the other value. This is done as follows. We only feed input in X (or in Y) and let the network converge. After convergence one can use the activities in A and B to compute the population vector [25] giving us the values v_A and v_B encoded by A and B.

Figure 6 shows the result of such inference tasks. We tested the inference accuracy by encoding all values $v_i \in \{p_i | i \in A\}$ in X (respectively all values $v_j \in \{p_j | j \in B\}$ in Y) and observing the values v_B (respectively v_A) computed by the network.

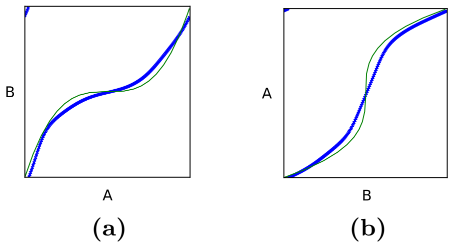

Fig. 4: Simple inference in sample network after the network has learned the relation $y = x^3$ (green thin line in both plots). **(a)** shows the results of the inference tasks (thick blue line) for a set of population codes fed to A (horizontal axis). **(b)** like (a) but for the opposite inference direction.

3.3 De-noising and Cue-Integration Tasks

In all of the following examples we add some noise on top of the activations computed with Equation (7). Figure 5(a) shows how the network performs inference with noisy input.

In addition to such noisy input signals our network can also cope with noisy values v_X and v_Y. Figure 5(b) is an example for the case when the inputs in X and Y are not in line with the learned relation R. The network settles in a state where the computed values v_A and v_B are again consistent with R. Figure 5(c) shows the same experiment but with different input strength set in X and Y. Note that the population receiving the stronger input gets significantly less shifted towards a place consistent with R than the other one.

Note that the soft winner-take-all implemented in our populations A and B is the reason for the described phenomena to work.

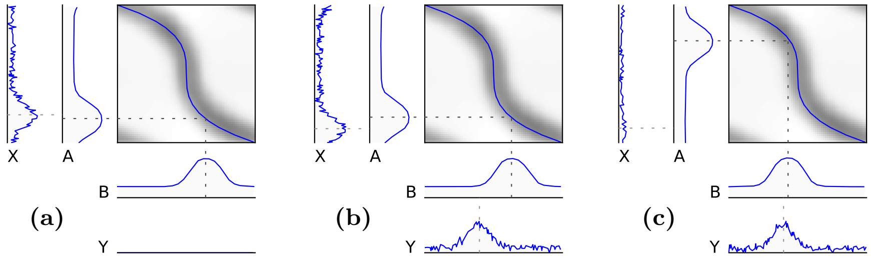

Fig. 5: Inference and de-noising: **(a)** An example of inference from X to Y which shows also the de-noising properties of the network with respect to noise in the firing rates of the units, **(b)** when two inputs are presented which are inconsistent with the learned relation the network shifts both peaks until their positions are in accordance with the relation, **(c)** same as (b) but with unequal reliability of the inputs (unequal input strength); note that the larger (more reliable) peak is much less shifted as in (b).

Clearly, such a de-noising task gets more and more difficult (and unreliable) depending on "how much" the value fed into Y differs from the "true" value that is consistent with the input in X. Eventually, if this difference gets too large, then the system will stop to find a compromise between these two values and instead will start to neglect one of the inputs. That, is the network will decide between the two values.

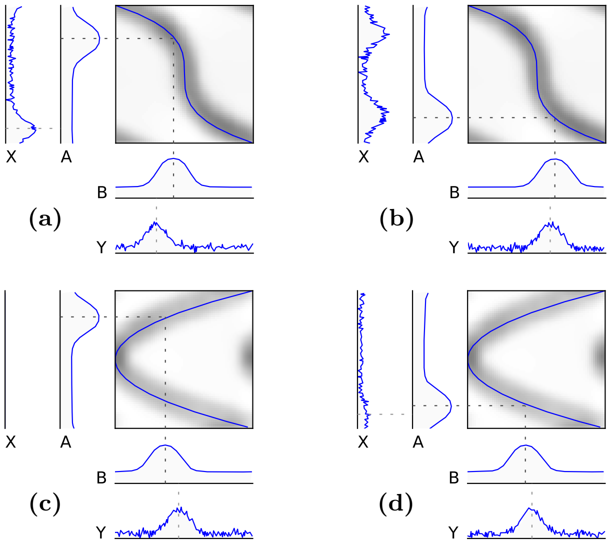

Fig. 6: Decision tasks. **(a)** when the peaks of the inputs are too far apart the network decides for one of the inputs and infers the other one, **(b)** when in X there are two contradicting inputs present, while one is being supported by the input in Y, the network decides for that combination of peaks., **(c)** in the case of a non-invertible function as $y = x^2$ there exist two possible peak positions and the systems decides for one of the two, **(d)** same as (c) but the networks decision is biased by a very small input fed to X.

3.4 Decision Tasks

As indicated at the end of the previous section, the network can be forced to decide whether to follow input X or input Y. Figure 6 shows the input and the settled state for four decisions being performed by the network.

Given inputs of similar strength the variance in the noise determines how the network will decide. If the inputs are of equal strength the network will essentially decide on one of them "randomly", meaning that small artifacts from the learning history will be responsible for the decision.

Figure 6(b) illustrates another, more complicated decision task. In this example the input in X actually contains two peaks. If the second input relates to one of these two peaks (with respect to R), the network will reinforce this peak and settle in a state consistent with R.

If the learned relation R corresponds to a non-invertible function (like $y = x^2$ for x in $[-1, 1]$) then, clearly, an input in Y may be in correspondence with more than one consistent X-value. The network will then have to pick one of the possible solutions. This example is illustrated in Figure 6(c). In addition, Figure 6(d) illustrates that already a seemingly small "noise" in the input in X suffices to move the generated value to the one that has a higher consistency with the input.

4 Discussion

In this paper we showed that it is possible to setup the dynamics of a simple network in such a way that it can learn the relations between two inputs X and Y. After learning, that is, after presentation of sufficiently many related input pairs, the network is then able to (*i*) *infer* missing input, (*ii*) to mediate between slightly conflicting inputs (*de-noising*), (*iii*) *cue-integration*, and (*iv*) *decide* between strongly conflicting inputs. If one would continue to present strongly conflicting inputs the system will then gradually change and eventually have learned the new relation.

The building blocks of our network, population coding, soft winner-take-all, Hebbian learning, and homeostatic activity regulation, are all biologically well motivated.

The next step, clearly, is to learn higher order relations between more than two input signals. To achieve this it will be necessary to replace the effectively one-dimensional populations used in our network by more complex recurrent networks capable of encoding these higher order relationships. Indeed, the internal connectivity of the areas, reflecting the topology of the input space, would ideally be learned based on the observed inputs themselves. This would allow both higher dimensional transformations (such as those related to gain fields [9]) and more abstract relationships to be learned with the same mechanisms.

References

[1] Carew, T.J., Hawkins, R.D., Kandel, E.: Differential classical conditioning of a defensive withdrawal reflex in aplysia californica. Science **219**(4583) (1983) 397–400

[2] Pavlov, I.: Conditioned Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex. Oxford University Press, London (1927)

[3] Rescorla, R., Wagner, A.: Variations in the Effectiveness of Reinforcement and Nonreinforcement. In: Classical Conditioning II: Current Research and Theory. Appleton-Century-Crofts, New York (1972) 64–99

[4] Wagner, A.R., Logan, F.A., Haberlandt, K., Price, T.: Stimulus selection in animal discrimination learning. J. Exp. Psychol. **76**(2) (1968) 171–180

[5] Bailey, C., Kandel, E.: Structural changes accompanying memory storage. Annu. Rev. Physiol. **55** (1993) 397–426

[6] Bailey, C., Chen, M.: Morphological basis of long-term habituation and sensitization in aplysia. Science **220**(4592) (1983) 91–93

[7] Bailey, C., Chen, M.: Morphological basis of short-term habituation in aplysia. J. Neurosci. **8**(7) (1988) 2452–2459

[8] Kandel, E., Spencer, W.A.: Electrophysiology of hippocampal neurons. ii. afterpotentials and repetitive firing. J. Neurophysiol. **24** (1961) 243–259

[9] Salinas, E., Sejnowski, T.: Gain modulation in the central nervous system: where behavior, neurophysiology, and computation meet. Neuroscientist (2001) 430–440

[10] Rutishauser, U., Ross, I., Mamelak, A., Schuman, E.: Human memory strength is predicted by theta-frequency phase-locking of single neurons. Nature (2010) online first

[11] Barco, A., Bailey, C.H., Kandel, E.: Common molecular mechanisms in explicit and implicit memory. J. Neurochem. **97**(6) (2006) 1520–1533

[12] Kandel, E.: Cellular mechanisms of learning and the biological basis of individuality. In: Principles of Neural Science. 4th edn. McGraw-Hill, New York (1991)

[13] Martin, S.J., Grimwood, P.D., Morris, R.G.: Synaptic plasticity and memory: an evaluation of the hypothesis. Annu. Rev. Neurosci. **23** (2000) 649–711

[14] Felleman, D., Essen, D.V.: Distributed hierarchical processing in the primate cerebral cortex. Cereb. Cortex **1**(1) (1991) 1–47

[15] Salinas, E., Abbott, L.F.: Coordinate transformations in the visual system: how to generate gain fields and what to compute with them. Prog. Brain Res. **130** (2001) 175–190

[16] Pouget, A., Sejnowski, T.: Spatial transformations in the parietal cortex using basis functions. J. Cognitive Neurosci. **9**(2) (1997) 222–237

[17] Zipser, D., Andersen, R.A.: A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. Nature **331**(6158) (1988) 679–684

[18] Rumelhart, D.E., Hinton, G., Williams, R.J.: Learning representations by back-propagating errors. Nature **323** (1986) 533–536

[19] Crick, F.: The recent excitement about neural networks. Nature **337** (1989) 129–132

[20] Zipser, D., Rumelhart, D.: The neurobiological significance of the new learning models. In: Computational neuroscience. MIT Press, Cambridge (1993)

[21] Deneve, S., Latham, P., Pouget, A.: Efficient computation and cue integration with noisy population codes. Nat. Neurosci. **4**(8) (2001) 826–831

[22] Douglas, R., Martin, K.: Recurrent neuronal circuits in the neocortex. Curr. Biol. **17**(13) (2007) 496–500

[23] Hebb, D.: The Organization of Behavior: A Neuropsychological Theory. Wiley, New York (1949)

[24] Turrigiano, G., Nelson, S.: Homeostatic plasticity in the developing nervous system. Nat. Rev. Neurosci. **5** (2004) 97–107

[25] Georgopoulos, A.P., Kalaska, J.F., Caminiti, R., Massey, J.T.: On the relations between the direction of two-dimensional arm movements and cell discharge in primate motor cortex. J. Neurosci. **2**(11) (1982) 1527–1537