Cogn Neurodyn (2014) 8:71–80
DOI 10.1007/s11571-013-9262-0

BRIEF COMMUNICATION

Vague-to-crisp dynamics of percept formation modeled as operant (selectionist) process

Roman Ilin · Jun Zhang · Leonid Perlovsky · Robert Kozma

Received: 15 October 2012 / Revised: 10 June 2013 / Accepted: 26 June 2013 / Published online: 4 August 2013

Abstract We model the vague-to-crisp dynamics of forming percepts in the brain by combining two methodologies: dynamic logic (DL) and operant learning process. Forming percepts upon the presentation of visual inputs is likened to model selection based on sampled evidence. Our framework utilizes the DL in selecting the correct "percept" among competing ones, but uses an intrinsic reward mechanism to allow stochastic online update in lieu of performing the optimization step of the DL framework. We discuss the connection of our framework with cognitive processing and the intentional neurodynamic cycle.

Keywords Operant conditioning · Operant learning · Dynamic logic · Mixture models · Model selection · Intrinsic reward

R. Ilin
Sensors Directorate, Air Force Research Laboratory, Building 620, Sensors Directorate, WPAFB, OH, USA
e-mail: roman.ilin@wpafb.af.mil

J. Zhang (✉)
Department of Psychology, University of Michigan, 530 Church Street, Ann Arbor, MI, USA
e-mail: junz@umich.edu

L. Perlovsky
Sensors Directorate, Air Force Research Laboratory, Building 600, Sensors Directorate, WPAFB, OH, USA
e-mail: leonid.perlovsky@wpafb.af.mil

R. Kozma
Department of Mathematical Sciences, The University of Memphis, 373 Dunn Hall, Memphis, TN 38152, USA
e-mail: rkozma@memphis.edu

Introduction

Dynamic logic learning (DL) is a computational theory of cognitive processes (Perlovsky 2001), which emphasizes the "vague-to-crisp" aspect of perceptual and cognitive processing. DL postulates that the internal concepts (models), which correspond to various perceptual categories, are learned by way of gradual transition from fuzzy and uncertain ("vague") initial models to sharp and definite ("crisp") models. Such vague-to-crisp transition mechanism presumably provides computational efficiency by avoiding the computational complexity involved in associating perceptual inputs to internal models. The DL mechanism is postulated to exist due to the need of an organism to satisfy the "knowledge instinct" (KI), which is an intrinsic motivation of the organism and is satisfied when similarity between the models and the perceptions is maximized. DL is mathematically formulated in a Bayesian learning framework and is related to statistical parameter estimation in mixture models—under certain conditions it can be considered as a maximum likelihood joint parameter estimation and model selection. DL has resulted in improvements in signal processing (Perlovsky 2010).

Operant learning (OL) is a theoretical framework for adaptive learning by an intelligent agent as it interacts with its environment. It is characterized by selecting the next action based on current action probabilities with subsequent adjustment of the probabilities based on a reinforcement signal. The probability of the currently selected action changes based on the strength of the reinforcement. At the same time, the probabilities of the alternative actions change only due to normalization to assure that the sum of action probabilities equals to one. The OL algorithm is different from the Bayesian learning where all action probabilities are adjusted by computing the posterior

probability based on current input. However, previous work (Zhang 2009a, b) showed that these two styles of learning are closely related at the level of ensembles of learning agents.

In this work, extending our preliminary results (Ilin et al. 2011), we show that DL and OL dynamics can be merged into a unified computational framework, Accordingly, the original DL batch learning algorithm will be modified (1) to achieve online processing, and (2) to use OL style updates. The association weights between inputs and models are adjusted using the OL procedure, where the reinforcement signal corresponds to the similarity between the selected model and the current input, while the model parameters adjusted using an OL style procedure. We demonstrate the operation of the new algorithm using numeric examples. This combined algorithm provides an alternative formulation of the vague-to-crisp DL process, with the added advantage of reducing computational complexity and realizing online processing.

Methodology

Dynamic logic framework

As mentioned in the introduction, DL is a computational theory of perceptual and cognitive processing in the brain. It is built on the idea of internal models that correspond to both tangible and abstract concepts in the environment. The internal models form a hetero-hierarchy modeling the relationships existing between the concepts. In the absence of sensory input, the models exist is a "vague" form. They are actualized in a "crisp" form by coming into contact with the sensor data. The actualization of the models causes them to adjust to sensory input and form the best possible match. The DL theory postulates that the brain is found in constant need to increase the match between its internal models and the environment. Such need is called the knowledge instinct (Perlovsky 2001). Driven by the knowledge instinct, the brain is in the perpetual process of learning and improving its internal models.

Mathematically, the internal world of an agent is described as a set of parametric statistical models. Consider a finite set of possible concepts, of size H. Denote the internal model corresponding to a concept $h = 1\ldots H$, by M_h. The match (called "similarity" below) between a model M_h and an individual sensory input x_n is given by the joint probability density function $g(x_n, M_h)$. Each model depends on a set of parameters S_h, called "association weights". Denote the set of all sensor inputs by $X = \{x_n, n = 1\ldots N\}$. The total similarity between all models and all inputs is given by the data likelihood

$$L(X, M(S)) = \prod_{n=1}^{N} \sum_{h=1}^{H} g(x_n, M_h). \quad (1)$$

Here, N and H are the total number of inputs and models, respectively. Using Bayes theorem, the similarity can be expressed through the a priori probability p_h of model M_h and the conditional density of the data given the model:

$$g(x_n, M_h) = p_h g(x_n | M_h) \quad (2)$$

Note that here "a priori probability" p_h is used with respect to the gathering of sensory input x_n at a particular moment—it is the probability that model M_h is true (but with yet-to-be-specified parameters S_h) prior to receiving x_n among a stream of sensory inputs. Equation (1) is maximized by iteratively computing the following quantities (Perlovsky 2001):

$$f_{hn}^{I} = g(x_n, M_h) \Big/ \sum_{h'=1}^{H} g(x_n, M_{h'}), \quad \forall n, h \quad (3)$$

$$p_h^{I+1} = p_h^I + \varepsilon_r \sum_{n=1}^{N} f_{hn}^I \frac{\partial \log g(x_n, M_h)}{p_h^I}, \quad \forall h \quad (4)$$

$$S_h^{I+1} = S_h^I + \varepsilon \sum_{n=1}^{N} f_{hn}^I \frac{\partial \log g(x_n, M_h)}{\partial S_h^I}, \quad \forall h \quad (5)$$

In Eqs. (3–5), I is the iteration number, and ε is the learning rate. The association weight f between the model M_h and current input x_n is computed in Eq. (3) based on the parameter estimates of the current model. Equation (4) adjusts the a priori probabilities. Equation (5) adjusts the parameter estimates S_h by weighted gradient ascent using the current association weights. The vague-to-crisp process is achieved by proper initialization of the models and optionally introducing additional parameters controlling the fuzziness of the models. Note that the association weights are the posterior probabilities of models M_h given input x_n.

The models are initialized to have comparable similarity to the incoming input, and therefore, initially the input is associated with more than one model. After the models are properly learned, the new input is correctly associated with the model that results in the maximum similarity or, equivalently, having the association weight close to 1. In (3–5), DL is described as a batch algorithm. This formulation has been used successfully in many applications, including the detection and tracking of targets (Deming 1998; Deming et al. 2007a, b; Perlovsky and Deming 2007; Ilin and Deming 2010). Appendix 1 provides more information on how the iterative procedure is derived.

Operant learning framework

Operant learning (OL) or operant conditioning is based on the process by which an animal modifies its behavior as a result of experiencing the consequences of its past behavior. OL uses the assumption that, in a given context the action is selected by the animal solely based on the action probabilities learned from past experience. Mathematically, given a repertoire of H actions with corresponding action probabilities p_h, the probabilities of *selected action h* are adjusted as follows

$$p_h^{n+1} = p_h^n + \varepsilon\,\theta_h\,(1 - p_h^n), \tag{6}$$

while the probabilities of the other actions $h' \neq h$ are given by

$$p_{h'}^{n+1} = p_{h'}^n - \varepsilon\,\theta_h p_h^n. \tag{7}$$

Here ε is the learning rate and θ_h is the reinforcement signal after action h is selected. The index n refers to the time step of the algorithm execution. These formulas express the idea that the probability of the currently selected action increases proportionally to the reward received after the action is selected. The probabilities of all other actions decrease in order to keep their sum equal one (normalization effect). The reward signal θ_h has to have upper bound in order to maintain the values of all probabilities between zero and one. Usually, the learning rate ε is chosen to avoid this problem. Even if occasionally wrong actions are selected and possibly rewarded, the right action choice will be learned after many trials given that the learning rate is sufficiently small (Zhang 2009a).

Sequential dynamic logic algorithm

One of the disadvantages of DL algorithm is the requirement to process all available data in a batch mode. It turns out that the algorithm can be modified for online data processing.

The algorithm given by (3–5) is modified for processing one input at a time. This turns the algorithm into stochastic gradient ascent. Such algorithm can be made more efficient by introducing the momentum term (Haykin 1999), which helps "remember" the derivatives obtained from processing previous inputs and make the gradient ascent trajectory smoother. Thus, the sequential DL algorithm is given as follows.

$$f_{hn} = p_h g\,(x_n|M_h) \Big/ \sum_{h'=1}^{H} p_h g\,(x_n|M_{h'}), \quad \forall h \tag{8}$$

$$D_{rh} = f_{hn} + \beta_r D_{rh}, \quad \forall h \tag{9}$$

$$p_h = p_h + \varepsilon_r \cdot D_{rh}, \quad \forall h \tag{10}$$

$$D_h = f_{hn} \frac{\partial \log g(x_n, M_h)}{\partial S_h} + \beta D_h, \quad \forall h \tag{11}$$

$$S_h = S_h + \varepsilon \cdot D, \quad \forall h \tag{12}$$

Equations (8–12) are applied to each input. Note that (8) is the same as (3) combined with (2). The gradients are computed based on a single input and the discounted values of the previous iteration gradients. New parameters β and β_r are introduced for the momentum terms.

Operant learning–dynamic logic (OL–DL) algorithm

Conceptually, the sequential DL procedure described in the previous section processes each input in two steps:

- Step 1: Equations (8–10): The input is compared with all models using Bayes theorem and the association weights between the input and the models are adjusted. These weights are used to adjust the model probabilities p_h.
- Step 2: Equations (11–12): the parameters of the models S_h are adjusted using gradient ascent, weighted by the association weights. The choice of new parameters is guided by the relative similarity between the input and the models.

In previous studies, (Deming 1998; Deming et al. 2007a, b; Perlovsky and Deming 2007; Ilin and Deming 2010) DL has been applied to scenarios with data inputs coming from multiple sources, such as target and clutter, and the task was to simultaneously estimate the parameters of all of the models. Probabilities p_h converge to the relative proportions of the data from different sources. Consider a scenario where all the data come from a single source. The algorithm will eventually identify the model best corresponding to that single source by adjusting its a priori probability to 1, while the a priori probabilities of all other models will become 0. This is equivalent to solving a model selection problem. In terms of operant conditioning, this is equivalent to learning the appropriate behavior through experience by adjusting internal parameters.

Dynamic Logic (DL) is formulated as a Bayesian model selection framework. Based on the above conceptual description, we can transform DL into a two stage OL framework, such as described in (Zhang 2009b). The first stage consists of the selection of the model, and the second stage consists of the selection of the model parameters. The repertoire of actions in the first stage consists of selecting one of H models, with model probabilities given by p_h. The repertoire of actions in the second stage consists of all possible parameter selection. In order to overcome the difficulty of considering an infinite number of possible actions, we assume that the parameters of each model can

take values from a reasonably large but finite domain. Suppose that there are N_{hs} possible choices of parameters of model h. The action probabilities for mode h are given by $p_{s|h}$, $s = 1 \ldots N_{hs}$.

Suppose that model h and model parameters s have been selected at a given step. The change of action probabilities is given by the following expressions, similar to (6a)–(6b) in (Zhang 2009b):

$$p_h^{n+1} = p_h^n + \varepsilon\, \theta_n\, (1 - p_h^n) \tag{13}$$

$$p_{h'}^{n+1} = p_{h_2}^n - \varepsilon\, \theta_n p_{h'}^n, \quad h' \neq h \tag{14}$$

$$p_{s|h}^{n+1} = p_{s|h}^n + \frac{\varepsilon\, \theta_n\, (1 - p_{s|h}^n)}{p_h^{n+1}} \tag{15}$$

$$p_{s'|h}^{n+1} = p_{s'|h}^n - \frac{\varepsilon\, \theta_n p_{s'|h}^n}{p_h^{n+1}}, \quad s' \neq s \tag{16}$$

Unlike in the common operant conditioning scenario, the reinforcement signal now does not originate from the environment. Rather it is computed according to the DL principles, based on how similar the selected model with selected parameters is to the observed input. It is therefore a function of $g\,(x_n, M_h)$. The exact functional form can be specified in many different ways; in this work we define the reinforcement as follows:

$$\theta_n = \left(A + \max\left(-A, \log g\,(x_n|M_h)\right)\right)^3 \tag{17}$$

In Eq. (17), constant A is the cutoff value for the logarithm of the likelihood function that is selected empirically. The transformation in (17) ensures that the reward is greater than zero, which is necessary for applying operant reinforcement learning. The power function is applied to increase the efficiency of the algorithm by making the difference between high and low likelihood inputs larger. The reward structure suggested here has a simpler functional form comparing to our previous publication (Ilin et al. 2011).

Equations (13–17) define the OL version of the DL framework (OL–DL), which is a two-stage process. The first stage results in the adjustment of the a priori probabilities of the models. The second stage adjusts parameter probabilities of the selected model. The reward is intrinsic and is a function of the similarity between the model and the input, in accordance with the knowledge instinct principle (Perlovsky 2001). The repeated application of the algorithm leads to learning the correct model and its parameters resulting in increased average reward, again in accordance with the DL principles. The new formulation is simpler and computationally more efficient than the sequential algorithm in (8–12). Note that during each input presentation only one similarity function is computed for the selected model, as opposed to the need to compute all similarities in (8).

Computational efficiency

Unlike the sequential version of the DL algorithm, the OL-DL algorithm does not involve the derivatives of the models. Therefore, it is arguably more biologically plausible. It can also be extended to more than two hierarchical levels in order to handle more complex models. These are the main motivations behind deriving the algorithm. In terms of computational efficiency, the number of operations in the DL algorithm is $O\,(ENM)$, where N is the size of the data, E is the number of training epoch, which is the number of times each data point is presented to the algorithm, and M is the number of models. In the case of OL–DL, the number of operations is $O\,(EN)$. The constant hidden inside the big-O notation may be larger for OL–DL as the speed of learning depends on the learning constant ε and consequently OL–DL may require more epochs to converge. The results obtained for a simple problem given in the next section show that the difference in the number of epochs is not significant.

Demonstration of results with the OL–DL algorithm

Description of the data set and applied models

Let us consider the following example to illustrate the operation of the algorithms. The data consist of $N = 800$ two-dimensional points originated from a stochastic source with unknown probability distribution. We experiment with three kinds of distributions: (a) uniform, (b) Gaussian with full covariance matrix, and (c) Gaussian with diagonal covariance matrix. Examples of the three kinds of data sets are shown in Fig. 1. We assumed that the mean value of all three data sources is 0 and is known to the algorithms.

The data points ("sensory inputs") are presented one by one, and the task is to learn the most appropriate model for these data. Since we are dealing with synthetic data, the repertoire of possible models includes: (1) uniform probability density, (2) Gaussian probability density with full covariance matrix, and (3) Gaussian probability density with diagonal covariance matrix. The uniform probability density is the same for all data points and equals the inverse of the area covered by the data. The area is updated as new data points are received by computing the minimum and the maximum coordinate in each dimension. Data are presented to the algorithm in random order that changes after each cycle (epoch) through all N points. In the case of sequential DL algorithm, the learning rates were set to the following values: $\varepsilon = 0.001$, $\varepsilon_r = 0.01$, $\beta = 0.9$, $\beta_r = 0.5$.

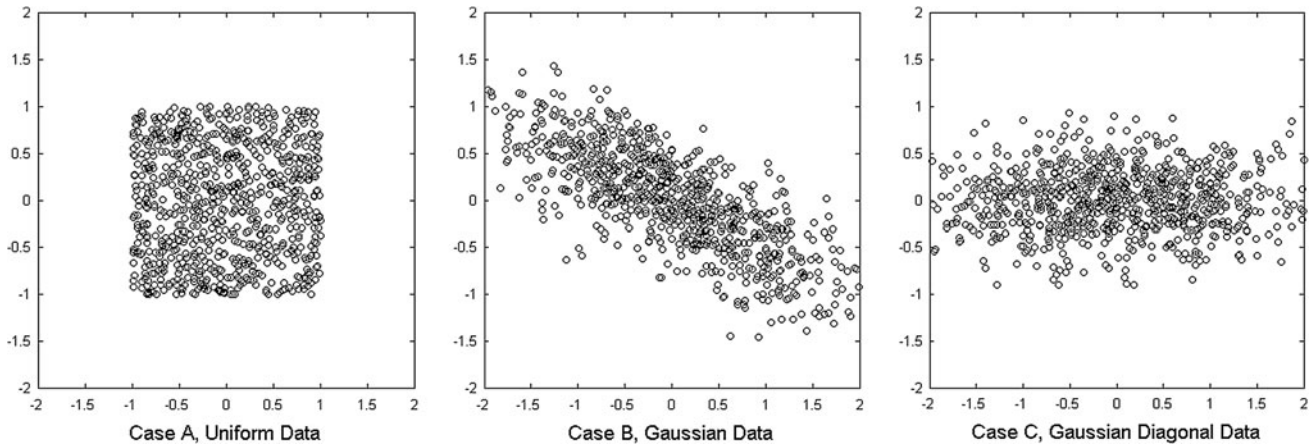


Fig. 1 Data samples from the three cases of two-dimensional probability density functions considered in this paper

Results

The sequential DL and the OL–DL algorithms described above have been implemented in Matlab with the three models. The execution of both algorithms can be illustrated by considering the evolution of the a priori model probabilities p_h. Since there are three possible models, we use the probability simplex, which is a triangle corresponding to the set of non-negative points adding to one. The computation starts with a point close to the center of the triangle. In the process of applying either algorithm, the probability of the model that corresponds to the data source increases to the value close to 1, while the probabilities of the other two models decrease to values close to 0. Figure 2 illustrates the performance of OL–DL algorithm with all three types of data source. We ran 50 experiments for each of the cases and as the figure shows the correct models are selected in each case. Similar results were produced for the sequential DL algorithm.

In order to apply the OL version of the DL algorithm, we made the model parameters discrete, as follows. The covariance matrix has three independent parameters. Each parameter is quantized over a certain range and a covariance matrix for each parameter combination is generated and stored in the computer memory. We experimented with different sets of possible covariance matrices. The results reported here employ a set of 36 matrices. Some of the possible choices for the covariance matrix are shown in Fig. 3, for both diagonal and full covariance matrix. The diagonal covariance model has also been made discrete with 28 possible choices. We also experimented with larger number of choices, up to 300, producing similar results. Although the number of possible action choices may be an important parameter, its influence will be studied in the future work. If the number of choices should become too large, the hierarchical schemes for model design will be implemented.

The parameters of the OL algorithm are set as follows: $\varepsilon = 0.00001$, $A = 10$. The initial action probabilities are assumed to be equal, and are set to: $p_h^0 = 1/3$, $p_{slh}^0 = 1/N_{hs}$. Simulation showed that the probability of correct model increases to 1 and the probabilities of the other two models decrease to 0. Similarly, the probability of the correct model parameters increases to 1. This process, along with the reward as a function of iteration is shown in Fig. 4. The reinforcement signal jumps up and down as the function of the selected models and depending on the model parameters selected in each iteration. In order to see the trend we smoothed the time series θ_n with moving average filter. The reward increases steadily over the execution of the algorithm.

We measured the number of epochs necessary for the algorithms to converge. The convergence criterion was met when one of the model probabilities p_h exceeded 0.99. The results for sequential DL and OL–DL are shown in the table below.

Table 1 shows that the number of times the full data set had to be presented to the algorithm was not significantly different for both algorithms. However, the OL–DL algorithm performs fewer computations because it only updates one of the models in each iteration.

Conclusions and discussion

This contribution explored connection between two computation frameworks for model selection: DL and operant conditioning. When applied to a problem involving simultaneous model selection and parameter estimation, the DL framework is conceptually similar to a two-stage OL sequence with intrinsically generated reinforcement signal. In its existing formulation, DL is a Bayesian learning framework. In this work we reformulated dynamic

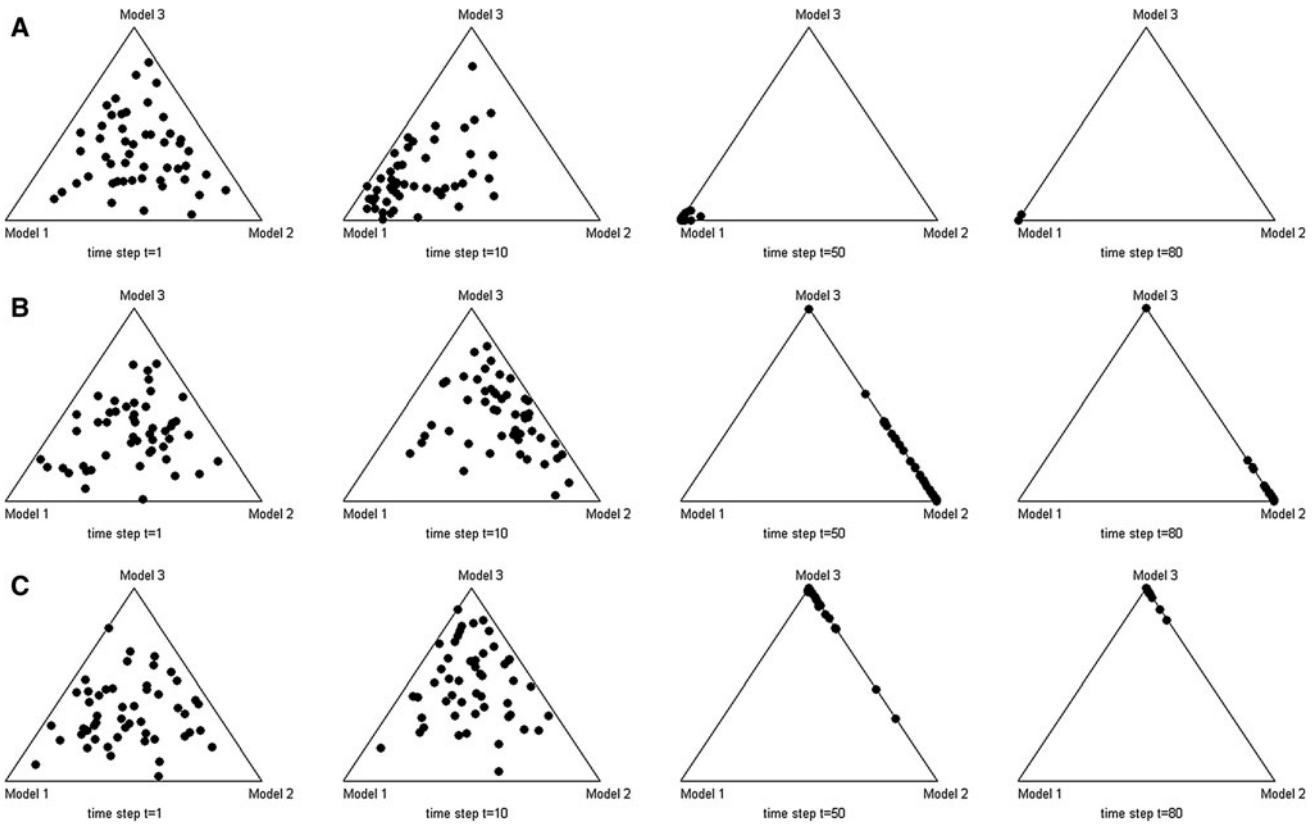


Fig. 2 Evolution of a priori probabilities p_h in case of sequential DL algorithm. Model 1—Corresponds to the uniform, Model 2—Gaussian with full covariance, Model 3—Gaussian with diagonal covariance. The *triangle* represents the probability simplex, where all points satisfy the relation $p_1 + p_2 + p_3 = 1$

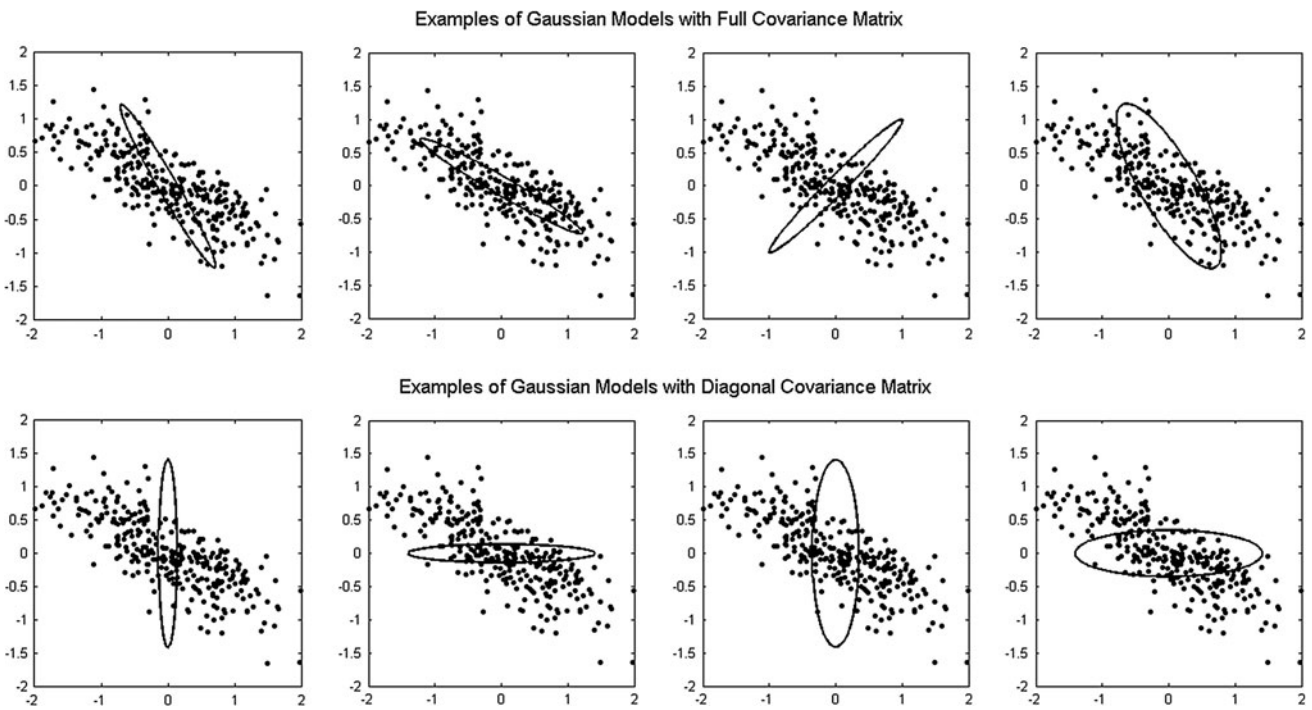


Fig. 3 Illustration of several possible choices for the covariance matrix for the full and diagonal covariance model. Sample data are displayed in the background. The ellipses correspond to the 2-standard deviation ellipses of the respective two-dimensional Gaussian distributions

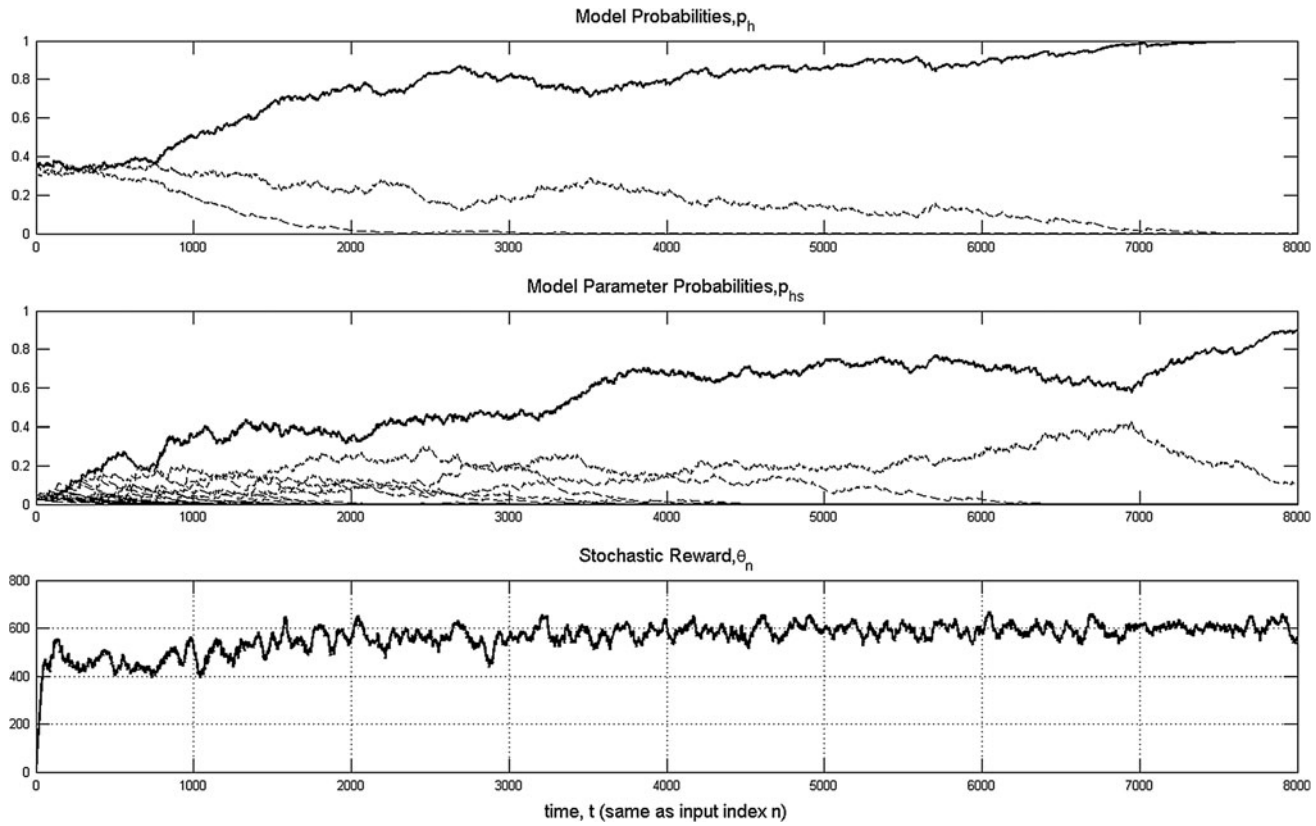


Fig. 4 Illustration of the OL dynamic logic algorithm (OL–DL). *Top* model probabilities p_h. *Middle* Gaussian model parameter probabilities $p_{s|h}$ for h corresponding to the true model. *Bottom* Moving average of the reward signal. The average value has been determined over 50 iterations

Table 1 Convergence of the algorithms

Algorithm	Number of epochs to convergence (mean ± standard deviation)
Sequential DL	9.55 ± 2. 15
OL–DL	9.15 ± 3.75

logic in terms of OL, at the same time preserving its main principles.

The principles investigated include (1) intrinsic reward, stemming from the postulated existence of the knowledge instinct, and (2) vague-to-crisp learning process for concept formation. The intrinsic reward increases with increased similarity between the input data and the selected model. The initial model selection and parameter selection probabilities are set to be equal, making all model choices equally possible. They are "vague concepts" in DL terms. The final result is the single model and single parameter value selected with probability one, corresponding to crisp concepts. We illustrated the operation of the new OL–DL algorithm through an example of model selection and parameter optimization. We demonstrate that the introduced combined algorithm provides an alternative formulation of the vague-to-crisp DL learning process, with the added advantage of reducing computational complexity and realizing online processing.

Neural mechanisms of DL and corresponding operant conditioning behaviour are of significant interest to researchers in the neuroscience of decision making. Biological mechanisms of learning, including OL, have been studied in the context of perceptual processing, see, e.g. (Stemme et al. 2011; Neiman and Loewenstein 2013). The idea of vague-to-crisp transitions found support in previous neuroimaging studies in the field of visual processing (Bar et al. 2006). In addition to early visual cortex, these authors identified involvement of fusiform gyrus and the prefrontal cortex (PFC) in operation of this mechanism. They have hypothesized that vague representations propagate fast via the magnocellular dorsal pathway (bottom-up signal) from early visual cortex to the PFC, in addition to more systematic and slower propagation along the ventral visual pathway. Bottom-up and top-down signals are integrated in object-processing regions of the occipital-temporal cortex (fusiform gyrus). Kveragra et al. (2011) in their study of processing of contextual information also identified vague-to-crisp mechanisms, involving the parahippocampal, retrosplenial, and medial prefrontal cortices.

A variety of representations for internal models have been considered (Rao et al. 2002), including inverse dynamics and forward models. OL–DL mechanisms considered in this paper support these types of models, including parametric, analytic, and probabilistic. OL–DL models mechanisms of bottom-up and top-down signal interaction; in the top-down stream representations serve as forward models, in the bottom-up stream representations serve as inverse models. A salient innovative property of OL–DL is that it supports contemporaneous learning of multiple models and assignment/association of data with models while avoiding combinatorial complexity that used to be a typical difficulty of these type learning problems (Perlovsky et al. 2011).

Levine and Perlovsky (2008) discussed functions of the orbitofrontal cortex (OFC), anterior cingulate cortex (ACC), and the dorsolateral prefrontal cortex (DLPFC) in terms of the postulated "knowledge instinct". The arguments there appear to have drawn supported from recent results on involvement of dopamine and opioid neurons (Litman 2005; Fiorillo 2011). At least two separate motivational systems are involved: "wanting" associated with mesolimbic dopamine activation and "liking" associated with nucleus accumbens opioid activation (Berridge and Robinson 1998, 2003; Berridge 2004; Tindell et al. 2005; Zhang et al. 2009). It is likely that OL–DL as well as mechanisms of the knowledge instinct involve both cognitive and motivational neural substrates. The above discussion is only a step toward understanding these mechanisms, and opens up a wide area for future research.

Models based on attractor dynamics have been used successfully for describing cognitive processing (Kay et al. 1995; Satel et al. 2009; Li and Nara 2008). It is anticipated that our proposed DL approach is of relevance to neurodynamics in a broader context, exemplified by the pioneering experimental and theoretical brain research by Freeman (1999). According to Freeman's dynamical system models of neural dynamics, cognition is described through a trajectory moving across a convoluted attractor landscape with valleys corresponding to memory patterns (Freeman 1975; Skarda and Freeman 1987; Kozma et al. 2003). In the basal mode of the intentional neurodynamic process, the brain is in a high-dimensional dynamic state, and the trajectory of the system explores the dynamic attractor landscape. This can be described as a gaseous chaotic state (Kozma et al. 2012), which may be interpreted as a *vague perceptual state* according to DL. When an input pattern is presented to the system, the oscillations undergo a phase transition and the trajectory is switched to a localized memory wing, which is described as condensation to a liquid-like cognitive state. This phase transition corresponds to the act of identification and decision making, and can be associated with the *formation of a crisp state* in terms of DL. The new state is maintained for some time, until conditions for a new quick switch are ready, when the whole cognitive cycle starts again. In this context, the vague-to-crisp transition can be viewed as a manifestation of the perceptual transition when the input data are perceived and identified in brains. Freeman's attractor dynamics approach suggests a consistent framework for perceptual learning process, but does not specify the model selection details. *In the present work, the operant conditioning process suggests a method for implementation of above perceptual mechanism.* Future work aims at comprehensive evaluation of the advantages of OL–DL algorithm with respect to alternative learning approaches. Our results advanced the OL and DL framework concerning both the computational efficiency and the biological plausibility, especially in the framework of the intentional neurodynamic and cognitive computing paradigms.

Acknowledgments This work is supported in part by the United Stated Air Force Office of Scientific Research, Department of the Air Force, work order #11RY06COR.

Appendix 1: Dynamic logic iteration

Let us consider the logarithm of the total similarity (1)

$$\log L\left(x, M\left(S\right)\right) = \sum_{n=1}^{N} \log \sum_{h=1}^{H} g\left(x_n, M_h\right) \tag{18}$$

The derivative of (18) with respect to the parameter S_h of model M_h can be written as follows.

$$\begin{aligned} \frac{\partial \log L}{\partial S_h} &= \sum_{n=1}^{N} \frac{\partial}{\partial S_h} \log\left(\sum_{h=1}^{H} g\left(x_n, M_h\right)\right) \\ &= \sum_{n=1}^{N} \frac{g\left(x_n, M_h\right)}{\sum_{h'=1}^{H} g\left(x_n, M_{h'}\right)} \frac{\partial \log g\left(x_n, M_h\right)}{\partial S_h} \end{aligned} \tag{19}$$

Introduce the following quantity

$$f_{hn} \equiv \frac{g\left(x_n, M_h\right)}{\sum_{h'=1}^{H} g\left(x_n, M_{h'}\right)} \tag{20}$$

The gradient is now expressed as follows

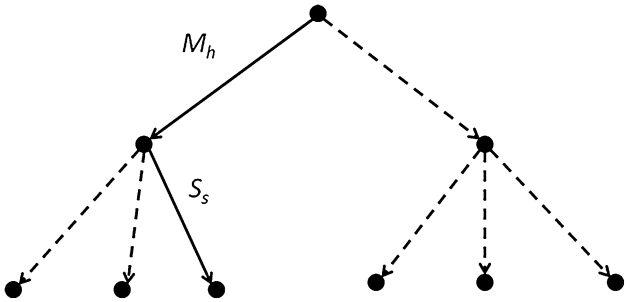
$$\frac{\partial \log L}{\partial S_h} = \sum_{n=1}^{N} f_{hn} \frac{\partial \log g\left(x_n, M_h\right)}{\partial S_h} \tag{21}$$

Instead of performing gradient ascent using (21) we can iteratively compute the quantities in (20) with fixed values of model parameters, and then perform a gradient ascent step with fixed values of the quantities f_{hn}. These quantities are referred to as "association weights" in the main text. The

resulting procedure is what is given by (3–5). The proof of convergence of this procedure is given in (Perlovsky 2001), and is not repeated here due to space constraints. Note that f_{hn} is the posterior probability of the model h given the observed value x_n. The algorithm can be alternatively derived using the principal of expectation maximization, although this is not how it was originally derived.

Appendix 2: Sketch of proof of OL–DL convergence

The algorithm given by (13–16) consists of a two-step action selection, illustrated below.



After the model and the parameters are selected, the reward value θ_n is computed. We can show executing the algorithm (13–16) results in increase of the expected reward. Indeed, consider the following quantity

$$E[\Delta\theta_n] = E[\theta_n - \theta_{n-1}]. \tag{22}$$

This is the expected value of the difference between reward received in the current and in the previous iteration. The reward is a function of the selected model and model parameters $\theta_n = \theta_n^{hs}(x_n)$. The expectation can be written by summing and integrating over the probability distributions associated with all the random quantities: the input data and the selected model and its parameters:

$$E[\theta_{n+1} - \theta_n] = \int_{\forall x_n} \sum_{h=1}^{H} \sum_{s=1}^{M} \left\{ p_h^{n+1} p_{s|h}^{n+1} \theta_{n+1}^{hs}(x_{n+1}) - p_h^n p_{s|h}^n \theta_n^{hs}(x_n) \right\} p(x_n)\, dx_n. \tag{23}$$

This expression can be rewritten by (1) bringing the integral inside the summations since it is only the reward that depends on the input, and (2) substituting the expressions (14–16) for the next action probabilities. We also need to separate the terms that correspond to the selected model and model parameters, with indices h and s, and the terms for the rest of the action and parameter selections, with indices h' and s'. We will omit writing the explicit dependency of the reward on input to make the expressions more clear.

$$\begin{aligned} E[\theta_{n+1} - \theta_n] = & \int_{\forall x_n} \left(p_h^n + \varepsilon\, \theta_n^{hs}(1 - p_h^n)\right)\left(p_{s|h}^n + \tfrac{\varepsilon}{p_h^n}\theta_n^{hs}\left(1 - p_{s|h}^n\right)\right)\theta_{n+1}^{hs} p(x_n)\, dx_n \\ & + \sum_{\substack{s'=1 \\ s' \neq s}}^{M} \int_{\forall x_n} \left(p_h^n + \varepsilon\, \theta_n^{hs}(1 - p_h^n)\right)\left(p_{s'|h}^n - \tfrac{\varepsilon}{p_h^n}\theta_n^{hs} p_{s'|h}^n\right)\theta_{n+1}^{hs} p(x_n)\, dx_n \\ & + \sum_{\substack{h'=1 \\ h' \neq h}}^{H} \sum_{\substack{s'=1 \\ s' \neq s}}^{M} \int_{\forall x_n} \left(p_{h'}^n - \varepsilon\, \theta_n^{hs} p_{h'}^n\right) p_{s'|h'}^n \theta_{n+1}^{hs} p(x_n)\, dx_n - \sum_{h'=1}^{H} \sum_{s'=1}^{M} \int_{\forall x_n} p_{h'}^n p_{s'|h'}^n \theta_n^{hs} p(x_n)\, dx_n \end{aligned} \tag{24}$$

The first term in (24) corresponds to the selected model and parameter actions. The second term corresponds to the selected model and its parameter actions that were not selected. The third term corresponds to the models that were not selected, and the fourth term is the expected

reward from the previous time step. Note that the expected value of the reward does not depend on the time step:

$$\int_{\forall x_n} \theta_n^{hs} p\left(x_n\right) dx_n = \int_{\forall x_{n+1}} \theta_{n+1}^{hs} p\left(x_{n+1}\right) dx_{n+1} = E\left[\theta\right]. \quad (25)$$

Taking (25) into consideration, opening the parenthesis, and disregarding the terms with ε^2, we can show that the following expression holds:

$$E\left[\Delta\theta_n\right] \geq 0. \quad (26)$$

This means that the algorithms (13–16) results in the increase of the expected reward and therefore to the selection of the best possible action sequence.

References

Bar M, Kassam K, Ghuman S, Boshyan AS, Schmid J, Dale AM et al. (2006) Top-down facilitation of visual recognition. In: Proceedings of the national academy of sciences, vol. 103, pp 449–454

Berridge KC (2004) Motivation concepts in behavioral neuroscience. Physiol Behav 81:179–209

Berridge KC, Robinson TE (1998) What is the role of dopamine in reward: hedonic impact, reward learning, or incentive salience? Brain Res Rev 28:309–369

Berridge KC, Robinson TE (2003) Parsing reward. Trends Neurosci 26:507–513

Deming R (1998) Automatic buried mine detection using the maximum likelihood adaptive neural system (MLANS). In: Proceedings of the 1998 IEEE ISIC/CIRA/ISAS joint conference, Gaithersburg, MD, 14–17 Sept 1998

Deming R, Schindler J, Perlovsky L (2007) Concurrent tracking and detection of slowly moving targets using dynamic logic. In: 2007 IEEE international conference on integration of knowledge intensive multi-agent systems: modeling, evolution, and engineering (KIMAS 2007), Waltham, MA, April 30–May 3

Deming R, Schindler J, Perlovsky L (2007) Track-before-detect of multiple slowly moving targets. In: IEEE radar conference 2007, Waltham, MA, 17–20 April

Fiorillo CD (2011) Transient activation of midbrain dopamine neurons by reward risk. Neuroscience 197:162–171

Freeman WJ (1975) Mass action in the nervous system: examination of the neurophysiological basis of adaptive behavior through the EEG. Academic Press, New York

Freeman WJ (1999) How brains make up their minds. Weidenfeld and Nicolson, London

Haykin S (1999) Neural networks: a comprehensive foundation. Prentice Hall, Prentice

Ilin R, Deming R (2010) Simultaneous detection and tracking of multiple objects in noisy and cluttered environment using maximum likelihood estimation framework. In: IEEE international conference, OCEANS'10. Sydney, Australia

Ilin R, Zhang J, Perlovsky L, Kozma R (2011) Dynamic logic for "Vague-to-Crisp" perception and dynamics of operant (selectionist) learning. In: 3rd international conference on cognitive neurodynamics (ICCN 2011), Niseki, Hokkaido, Japan, June 2011

Kay L, Shimoide K, Freeman WJ (1995) Comparison of EEG time series from rat olfactory system with model composed of nonlinear coupled oscillators. Int J Bifurcation Chaos 5(03):849–858

Kozma R, Freeman WJ, Érdi P (2003) The KIV model—nonlinear spatio-temporal dynamics of the primordial vertebrate forebrain. Neurocomputing 52–54:819–826

Kozma R, Freeman WJ, Vitiello G (2012) Adaptation of the generalized carnot cycle to describe thermodynamics of cerebral cortex. In: Proceedings of the IEEE 2012 international joint conference neurology network. IJCNN2012, Brisbane, Australia, June 2012 pp 3229–3236

Kveraga K, Ghuman AS, Kassam KS, Aminoff EA, Hämäläinen MS, Chaumon M, Bar M (2011) Early onset of neural synchronization in the contextual associations network PNAS, 108(8): 3389–3394

Levine DS, Perlovsky LI (2008) Neuroscientific insights on biblical myths: simplifying heuristics versus careful thinking: scientific analysis of millennial spiritual issues. Zygon J Sci Religion 43(4):797–821

Li Y, Nara S (2008) Novel tracking function of moving target using chaotic dynamics in a recurrent neural network model. Cogn Neurodyn 2(1):39–48

Litman JA (2005) Curiosity and the pleasures of learning: wanting and liking new information. Cogn Emot 19(6):793–814

Neiman T, Loewenstein Y (2013) Covariance-based synaptic plasticity in an attractor network model accounts for fast adaptation in free operant learning. J Neurosci 33(4):1521–1534

Perlovsky L (2001) Neural networks and intellect. Oxford University Press, UK

Perlovsky LI (2010) Neural mechanisms of the mind, Aristotle, Zadeh, and fMRI. IEEE Trans Neural Netw 21(5):718–733

Perlovsky L, Deming R (2007) Neural networks for improved tracking. IEEE Trans Neural Netw 18(6):1854–1857

Perlovsky LI, Deming RW, Ilin R (2011) Emotional cognitive neural algorithms with engineering applications. Dynamic logic: from vague to crisp. Springer, Heidelberg, Germany

Rao RPN, Olshausen BA, Lewicki MS (2002) Probabilistic models of the brain: perception and neural function. Bradford Books, MIT Press, Cambridge, MA

Satel J, Trappenberg T, Fine A (2009) Are binary synapses superior to graded weight representations in stochastic attractor networks? Cogn Neurodyn 3(3):243–250

Skarda CA, Freeman WJ (1987) How brains make chaos in order to make sense of the world. Behav Brain Sci 10:161–195

Stemme A, Deco G, Lang EW (2011) Perceptual learning with perceptions. Cogn Neurodyn 5(1):31–43

Tindell AJ, Berridge KC, Zhang J, Pecina S, Aldridge JW (2005) Ventral pallidal neurons code incentive motivation: amplification by mesolimbic sensitization and amphetamine. Eur J Neurosci 22:2617–2634

Zhang J (2009a) Adaptive learning via selectionism and Bayesianism. Part I: connection between the two. Neural Netw 22(3):220–228

Zhang J (2009b) Adaptive learning via selectionism and Bayesianism. Part II: the sequential case. Neural Netw 22(3):229–236

Zhang J, Tindell AJ, Berridge KC, Zhang J, Aldridge JW (2009) A neural computational model of incentive salience. PLoS Comput Biol 5:1–14