Evolution, development and learning with predictor neural networks

Konstantin Lakhman and Mikhail Burtsev

Neuromorphic Cognitive Systems Lab, Kurchatov NBICS-Centre,
National Research Center “Kurchatov Institute”, 1 Kurchatov sqr., Moscow, Russia
klakhman@gmail.com

Abstract

Predictor neural networks is an innovative model for self-learning.

Introduction

Model

Overview

The proposed model for the generation of the neuromorphic controller consists of the three main blocks: evolutionary algorithm, development algorithm and algorithm for self-learning during the agent’s life (Fig. 1).

We used neuroevolutionary model based on duplications and divergence of neurons’ functional roles (Lakhman and Burtsev, 2013). We had to adapt this principle because in the current research the topology of neuromorphic controller is not directly encoded in the genome of the agent. Instead, genome is a network consists of so-called *neuronal pools*, and controller’s structure is determined during the development process.

According to the neuronal group selection theory (Edelman, 1993) development algorithm implements two different objectives: generation of a primary repertoire of behavior and generation of a *“diversity of repertoires”* required for further self-learning during life. During the developmental process agent’s genotype (network of neuronal pools) translates into phenotype, i.e. the controller providing the agent’s behavior in the environment. Groups of neurons are encoded in the genome via the neuronal pools notion. Neurons from the same pool construct similar but not identical connections with the rest of the network. The most connected parts of the network are determined via stochastic endogenous activations of neurons in the isolation from model environment. These parts will be active in the very beginning of the agent’s life. The remaining parts of the network form a set of *silent neurons* that will be used for further learning.

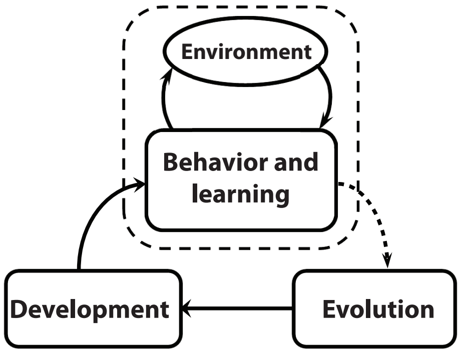

Figure 1: Schematic representation of the model’s components

Behavior in accordance with the theory of functional systems (Anokhin, 1974) occurs in three stages: generation of an action and prediction of the result; evaluation of the action result; formation of the new functional systems of neurons if additional leaning is needed. We present realizations of these phases on the level of individual neurons of the controller. Controller’s neural network structure consists of two types of connections between neurons: effector synapses and *predictor synapses*. Effector synapses correspond to the standard connections in the theory of artificial neural networks, that transmit excitation between neurons. Predictor synapses do not serve for the excitation propagation and are used by neurons to generate prediction of its future activity. Moments of the *mismatch* between agent’s actions and environment are detected in the case of discrepancy of formed prediction and observed activity of the neurons. It is necessary to conduct learning procedure at such a moments. This procedure includes activation of the silent neurons, that specialize to modify agent’s behavior in the problematic states of the environment via special organization of their connectivity structure. Agent’s ability to generate meaningful and useful prediction at the neuronal level is subject to selection process of the predictor connections in the course of evolution.

Environment with hierarchy of goals and agent's controller

We used hypercubic environment (Lakhman and Burtsev, 2013) as the model environment, where a population of agents evolves. Here we give only a brief description of this model and a concept of goals in it. State of the environment is represented as a binary string and at each discrete time step the agent performs an action – changing the state of the one bit of this string. Thus agent is moving along edges of the hypercube. The goal in this environment is defined as the consecutive changes of a particular bits of the state vector. Set of such a goals in the environment forms a branched hierarchy: one goal could be nested in another one, i.e. be a beginning of it, and two different goals could have identical nested goal.

Agent operates in the environment for a fixed amount of time, performs actions and therefore achieves various goals. Fixed reward is associated with each goal, that affects the reproductive success of an agent in the evolution. However, in contrast to paper (Lakhman and Burtsev, 2013) we have changed the mechanism of reward's recovery in case when the agent attempts to repeatedly reach the same goal in a short amount of time – recovery period of a goal. Previously, the reward associated with a goal after it's achieving was reseted to zero level and then linearly recovered to its initial value. In the current research we had to introduce the mechanism, which would allow the agent to obtain information about the success in the goals' achievement. With this purpose we *"restrict"* the agent in the achieving not fully restored goal. If the agent was trying to perform an action that would lead to the achievement of the unrestored goal, we blocked the change of the bit and the state vector of the environment remained the same. Thus, the agent should develop a correct interpretation mechanism of such a situations in the course of evolution.

Agent's behavior in the environment is controlled by a neural network. The structure of this neurocontroller is determined by a processes of evolution, development and learning. The controller consists of three types of neurons: input, output and interneurons. Number of input and output neurons is fixed and determined by a dimension n^{env} of the hypercubic environment (within this study we used environments with 8 bits dimension). Input neurons are used to transmit the information about current state vector into the network, so input layer has one neuron for each bit. Output neurons encode action that agent performs on a current discrete time step. Each pair of output neurons is responsible for turning on or off a particular bit of the state vector. Current agent's action are selected according to the pair of the most active output neurons. Thus, the number of output neurons is determined as:

$$n^{out} = \underset{n}{\operatorname{argmin}} \left[\binom{2}{n} \geq 2n^{env} \right] \quad . \tag{1}$$

Interneurons of a neurocontroller are organized in a layered topology, which in general case has no restriction on its connectivity structure. This allows to form both direct and recurrent effector synapses. As usual excitation propagates by recurrent effector synapses with unit time delay.

Evolutionary algorithm

Genotype of an agent is a triple (network with two types of connections):

$$G = (P, EC, PC) \quad , \tag{2}$$

where $P = \{P_\alpha\}$ is a set of network's neuronal pools, $EC = \{ec_{\gamma\alpha}\}$ is a set of of effector connections between pools, $PC = \{pc_{\gamma\alpha}\}$ is a set of predictor connections. During the development process this genotype are used to construct neuromorphic system that will control agent's behavior and provide learning. Each neuronal pool P_α in the genotype corresponds to a group of neurons with similar connectivity topology and has the following parameters: *a)* pool's size c_α, i.e. the number of neurons that will be formed from this pool during development; *b)* mean m_α and standard deviation σ_α of the biases of the neurons, that corresponds to a particular pool.

Each effector connection $ec_{\gamma\alpha}$ between pools α and γ has the following parameters: *a)* mean $m_{\gamma\alpha}$ and standard deviation $\sigma_{\gamma\alpha}$ of the weights of the synapses, that will be formed between neurons belonging to the corresponding pools; *b)* probability of the synapse development $p^{\text{dev}}(ec_{\gamma\alpha})$.

Predictor connections $pc_{\gamma\alpha}$ have only one attribute – the probability of connection development $p^{\text{dev}}(pc_{\gamma\alpha})$ (by analogy with effector connections).

Population of an agents with controller, resulting from a translation of a described genomes, are placed in the environment to determine their effectiveness. The success of each agent is calculated as the total amount of rewards, which it could collect for a fixed amount of time. Reproductive success of the agent, i.e. the probability to be selected as the parent for the next population, is directly proportional to the agent's effectiveness. Each selected parent goes through a number of mutations, among which are mutations of all numerical parameters of the genome introduced above, to produce an offspring. We used *duplication pool* mutation described in detail in the paper (Lakhman and Burtsev, 2013) as the main structural mutation. This mutation implies formation of a two pools with the same connectivity structure and half the size from a single "parent" pool. We also used addition and deletion of both types of connections as additional structural mutations. Full list of numerical parameters of mutations that have been used for the simulation is provided in the Appendix A.

Development algorithm

Developmental algorithm used in this paper consists of three phases: construction of a complete neuromorphic network, simulation of a stochastic version of a network isolated from the model environment, selection of the network's structural elements according to the numerical indicators obtained during the simulation.

Formation of a neurocontroller During the process of constructing a complete network each neuronal pool is translated into a fixed number of neurons determined by the size c_α of the corresponding pool P_α. Bias b_i of each neuron v_i is determined according to the normal distribution: $b_i \sim \mathcal{N}\left(m_\alpha, \sigma_\alpha^2\right), \ \alpha : v_i \in P_\alpha$. Effector synapses between constructed neurons are determined in accordance with the structure of effector connections in the genome. $\forall \alpha, \beta, i, j :$ $v_i \in P_\alpha, v_j \in P_\beta \rightarrow w_{ji} \sim \mathcal{N}\left(m_{\beta\alpha}, \sigma_{\beta\alpha}^2\right)$, if $ec_{\beta\alpha} \in EC$ with probability $p^{\mathrm{dev}}\left(ec_{\beta\alpha}\right)$; and $w_{ji} = 0$ otherwise, i.e. the synapse has not developed. Thus effector synapse between neurons is formed with some probability $p^{\mathrm{dev}}\left(ec_{\beta\alpha}\right)$ if corresponding connection is present in the genome. Thereby every neuron in the controller has unique connectivity structure both in terms of topology and weights distribution. Development of predictor synapses occurs similarly with the only difference being that the predictor synapses do not have weights.

Model of evolution of networks, that consist of neuronal pools and subsequently are translated into a neural networks, was first proposed in the framework of *Enforced Sub-Populations* evolutionary algorithm (Gomez and Miikkulainen, 1999). However, according to this algorithm only one neuron was selected from each pool in the development. In this case, this mechanism was used to decrease the coadaptation of different neurons, that lead to development of unique specializations of neurons. In the proposed model a similar approach is used to generate a number of variations of the same specialization within single network.

Neurons' selection The second main phase of a developmental process is a simulation of a stochastic version of a network. During this phase several numerical characteristics are calculated that will be used for to separate neurons into two groups: active and silent. Active neurons will be used within the neurocontroller in the beginning of agent's life, therefore forming primary repertoire of behavior. Nonactive or silent neurons form required for further learning set of various local modifications of the network. To determine which neurons are considered active immediately after the "birth" of the agent we utilized competitive principle proposed in the theory of neuronal group selection (Edelman, 1993). Under this approach selection of neurons during the developmental process occurs on the basis of their level of activity.

Developmental process takes place in the isolation from the model environment for a fixed amount of discrete time T^{sys}. At each time step each neuron of the network exhibit spontaneous activation with a fixed probability p^{spon}. Outputs of the remaining neurons are calculated in the standard way using truncated positive sigmoid activation function (truncation implies zero output in case of the negative neuron's potential). Spontaneous activations of neurons are designed to provide signal flow in the network in the absence of information about the external environment. During the simulation period we are calculating the total signed value of the potential received by each neuron. Within each pool fixed fraction p^{act} of the neurons are selected to become active and remaining neurons become silent. The probability for the neuron to be selected is directly proportional to the corresponding value of accumulated potential.

Selection process are also applied to predictor synapses, which role will be explained in detail in the next section. Basically this type of synapses are used to predict future activations of neuron. If there is a predictor synapse between neurons v_i and v_j then pre-synaptic neuron's activity predicts that post-synaptic neuron will be active on the next time step (and vice versa in case of absence of activity). Thus development process is necessary to select the individual synapses that form a statistically significant prediction. Predictor synapses between active neurons, whose average prediction rate is less then some threshold value L^{sig}, e.g 0.5 in the current study, are being deleted from the network. Predictor synapses connecting silent neurons wit active ones will be selected in the learning process.

It should be explicitly noted that the outputs of the silent neurons during the agent's life are always set to zero, regardless of the value of potential they receive by incoming effector synapses.

Learning algorithm

Developmental algorithm allows to generate controller which provides innate behavior. Nevertheless translation of the genotype into phenotype con not be expected to generate ideal behavior (due to stochasticity of the algorithm). But more importantly the primary repertoire of behavior might be partially irrelevant under even slightly changed environment. These circumstances explain the importance of a learning algorithm.

It is necessary to solve two basic questions when designing a learning algorithm: *1)* at what point learning should begin; *2)* how to perform learning if it is needed.

Mismatch detection To solve the first problem it is necessary to develop mechanism for detecting the moments in which additional learning is needed. These moments could be regarded as behavioral stages in which agent performs actions that earlier led to adaptive result but at the current time they are not successful. To address this problem in accordance with the theory of functional systems (Anokhin, 1974) some *functional system* generates both the program of actions and so-called *acceptor of action results* (AAR). This functional system for our purposes could be considered as a population of neurons that responsible for the implementation of a particular behavior. In the simplest case AAR corresponds to the expected changes of the environment in response to the chosen action. If AAR is not consistent with observed physical parameters of the environment, obtained immediately after the performance of the planned action, then system "concludes" that additional learning is needed. At such moments a functional system is in so-called *mismatch state*. These functional systems are organized into the network structure and, thus, the "environmental" parameters for the particular system could consist of both real environment's parameters and signals from another systems.

In the current study we formalized this mechanism for evaluation of goal-directed behavior for each individual neuron in the agent's neurocontroller. Several theoretical papers concluded that the biological neurons are able to provide similar functions (Fiorillo, 2008).

Neuron's "environment" consists of the signals from other neurons. Basic neuron's actions are being in the excited or ground state at a given time step. Formally, the neuron is in the excited state when its output is greater than zero (according to the truncated sigmoid activation function). Each neuron forms prediction about its own future activity based on input predictor synapses from other neurons. Each presynaptic neuron makes its contribution to the total prediction in accordance with the following scheme: if the presynaptic neuron was excited at time step t then it predicts that the postsynaptic neuron will be excited at time step $t+1$, and vice-versa if the presynaptic neuron was in the ground state. Thereby each neuron could calculate simple probability distribution of its activity on the next time step based on predictor signals:

$$\begin{array}{c} p^{\mathrm{exc}}\left(v_j,t\right) = \frac{\left|\left\{v_i \| ps_{ji} \in PS, o_i(t-1)>0\right\}\right|}{\left|PS_j^{\mathrm{act}}\right|} \\ p^{\mathrm{sil}}\left(v_j,t\right) = 1 - p^{\mathrm{sil}}\left(v_j,t\right) \end{array}, \qquad (3)$$

where $p^{\mathrm{exc}}\left(v_j,t\right)$ is the probability of neuron v_j of being active at time step t, PS is a set of network's predictor synapses, o_i is the output of neuron v_i, PS_j^{act} is a set of incoming predictor synapses of neuron v_j that are coming from active neurons. Neuron makes prediction based on this distribution only if one of these probabilities exceeds a fixed threshold $L^{\mathrm{pred}} \in (0.5, 1]$. As follows from Eq. 3 only active neurons are involved in the formation of prediction.

Described procedure assumes two types of the possible mismatch situations: I-type mismatch implies the absence of the neuron's activity when it was predicted and II-type mismatch implies the presence of the activity when it was not predicted.

Neuronal learning According to the systems-selection theory (Shvyrkov, 1986) learning at the neuronal level corresponds to the formation of neuronal specializations for the problem situations. Such a specialization occurs via the selection of neurons from the "reserve" of low active cells or, in our case, from silent neurons.

Due to the introduced mechanism for mismatch detection one could effectively locate a specific place in a neuromorphic controller where it is necessary to make modifications during learning. Here we give the learning procedure for I-type mismatch. In this case we find in the neuronal pool of mismatched neuron such a silent neuron, which in this behavioral situation gets maximum positive potential from the presynaptic neurons. Found silent neuron is added to the set of network's active neurons. We also transform the set of incoming synapses of this neuron and leave only those that allow it to effectively recognize the current situation. This implies the deletion of synapses from neurons that have not been active on the current time step. Additionally we add a strong excitatory synapse $w \sim \mathcal{U}(0.5, 1)$ from activated neuron to the mismatched one. This synapse could potentially lead to the elimination of the mismatch on the postsynaptic neuron in the similar behavioral situations in the future. We also add additional predictor synapses from the neurons that are predicted activation of the mismatched to the set of incoming synapses of the activated neuron.

Learning for II-type mismatch occurs in exactly the same manner except that we add strong inhibitory connection between activated and mismatched neurons in order to avoid mismatch in the similar behavioral situations.

The presented learning procedure is performed for all mismatched neurons on each time step resulting in a set of local modifications of connectivity structure. This scheme fits into the general learning approach in which learning is considered in terms of the traditional evolutionary mechanisms: generation of diversity and selection (Burtsev, 2008). The activation of the silent neurons could be interpreted as a local mutations of neurocontroller, which can potentially lead to successful behavior. Nonetheless, learning terminates only if there is no mismatch between organism's actions and environment detected at the neuronal level.

Experimental results

As the first step we have studied how learning algorithm affects efficiency of an agents from a statistical point of view. We have randomly generated 10 environments with different goals structure and run evolution with and without learning (only development process) 5 times for each environment. Then we detected the best population in terms of average accumulated reward in each run. For all agents in the best population we have performed development process for 5 times. Finally, we run these 5 different controllers for each agent from all $2^{n^{\mathrm{env}}}$ initial states of the environment (in our case of 8-bit environment there were 256 initial states). On the Fig. 2A we present averaged results of the described analysis procedure. We have not found statistical difference between efficiency of the agents evolved with and without learning – t-test showed $p\mathrm{-value} = 0.23$. However, shut-

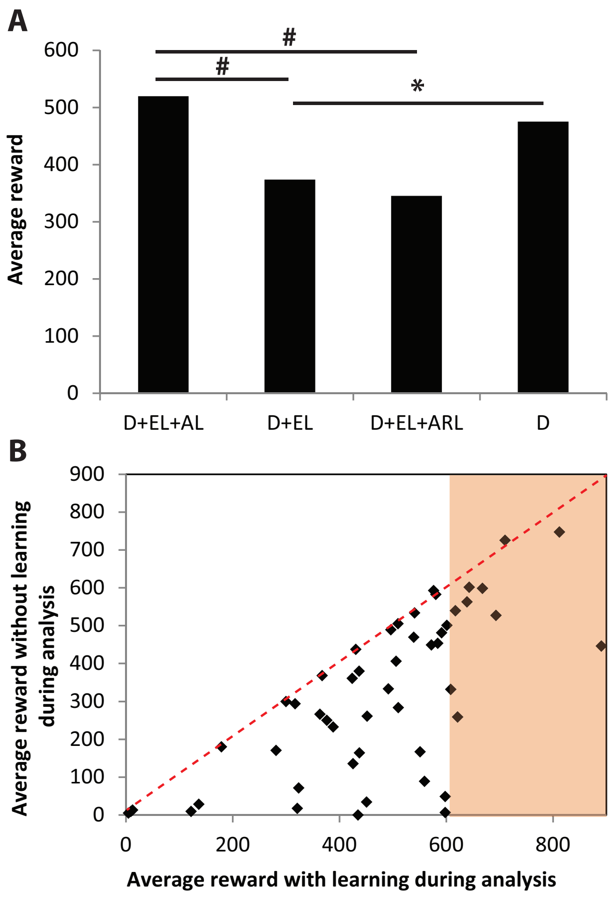

Figure 2: Efficiency analysis. **A)** Efficiency of the different types of agents (D – development, EL – learning during an evolution, AL – learning during an analysis phase, ARL – "random learning" during an analysis phase). # denotes paired samples t-test with $p-$value < 0.001, $*$ denotes t-test with $p-$value $= 0.012$. **B)** Comparison of the efficiency of the learning agents and the same agents without learning. Each dot represents averaged value for the one evolutionary run. The best 20 runs are indicated.

down of learning only in the analysis phase significantly decreases the efficiency that reveals the importance of learning for these agents. We provide different representation of this phenomenon on the Fig. 2B. As one can see several runs do not suffer from learning's shutdown, however this doesn't mean that learning didn't play significant role in these runs during the evolution.

We have also tested our learning mechanism against simple random activation of silent neurons (Fig. 2A). *Random learning* implies that silent neuron has a fixed probability of activation and inclusion in the network at a given time step. We applied this procedure in the analysis phase to the agents evolved with learning. We examined several activation probabilities in the interval $[0.01, 0.1]$, but show results only for the best value 0.5. Random activations of neurons do not help to increase efficiency comparing to our learning mechanism and even showed some tendency to decrease efficiency (paired samples t-test with p-value 0.08) comparing to the results of the same agents but without any learning.

Our results suggest that evolutionary algorithm "finds way" to make non-learning agents almost as effective as learning ones. We found that the distribution of pools' sizes for the non-learning agents is shifted toward smaller sizes (data is not shown). These agents also have more pools in average than learning agents. This fact means that non-learning agents are in some sense more determined since development algorithm is highly stochastic.

As we have showed earlier (Lakhman and Burtsev, 2013) behavior evolved in the course of evolution in a hypercubic environment consists of two phases: preliminary sequence of actions unique for the particular initial state and a cycle of actions that agent infinitely repeats. We will address the second phase as behavioral policy. We have conducted analysis of the innate and learned behaviors' evolution for the best run (Fig. 3A). For this purpose we performed development for 5 times for the best agents in each generation. Then we evaluated resulting controllers with and without learning. The analysis showed that at certain evolutionary moments efficiency of the learned behavior sharply increases as compared with innate behavior.

The increase in efficiency can occur both while efficiency of the innate behavior is decreasing (Fig. 3B) and while its efficiency remains unchanged (Fig. 3C). On the behavioral level the change of the dominant behavioral policy in the population is underlying such moments. At these moments one can find up to 200 rare modifications of the dominant behavioral policy in the behavior of the learning agents depending on the course of development and learning (data is not shown). The number of observed policies are sharply reduced when we analyze the same agents but without learning. This phenomenon might be an important mechanism

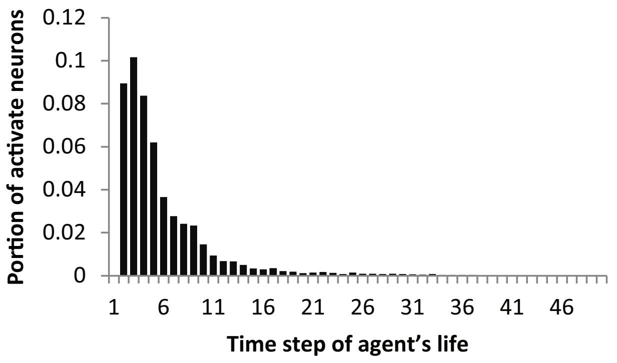

Figure 4: Dynamics of the activations of silent neurons during an agent's life averaged over the best 10 evolutionary runs

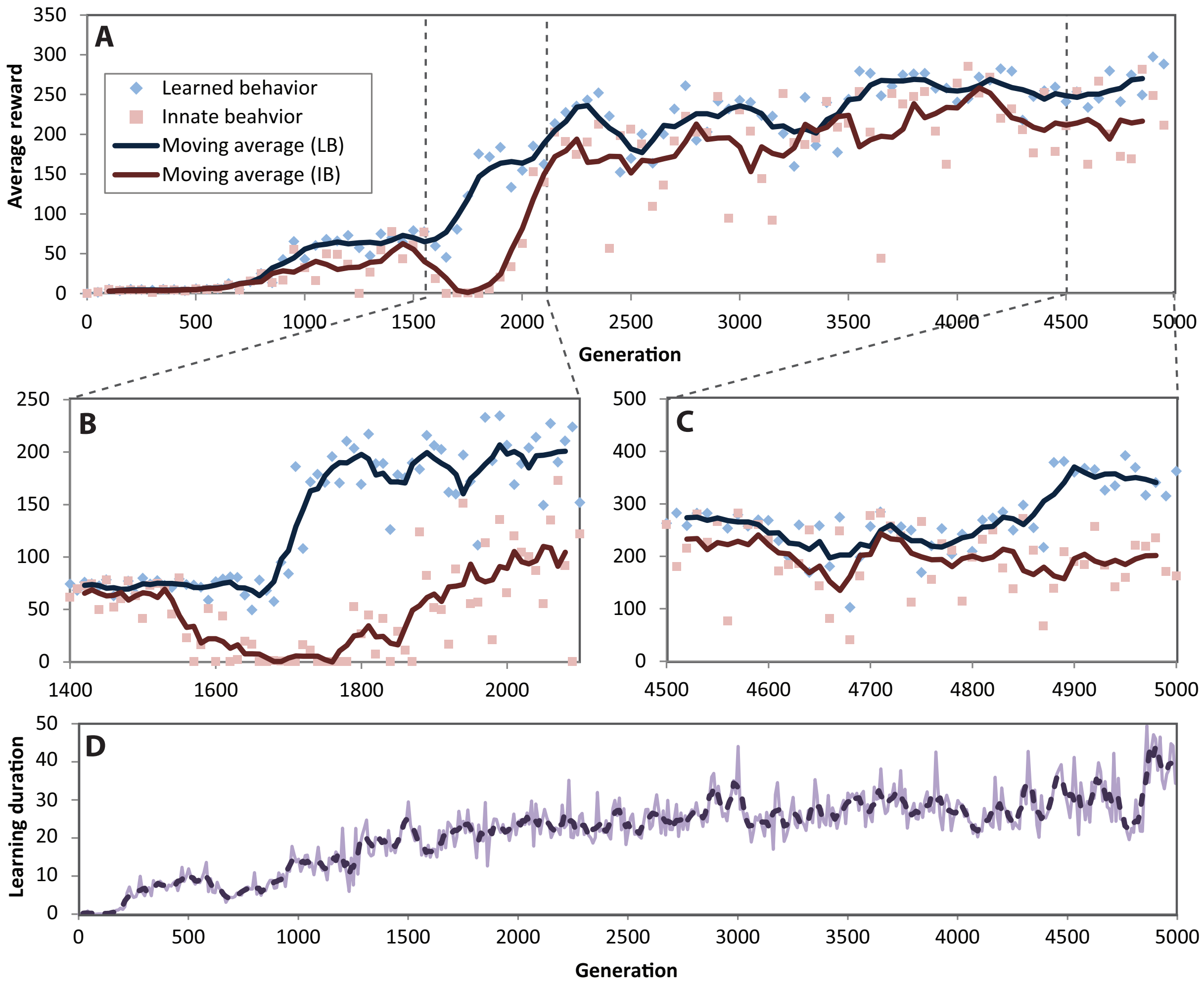

Figure 3: Behavioral evolution. **A)** Dynamics of the average efficiency of the innate and learned behavior for the best evolutionary run. The values are calculated for the best agent of the each 50-th generation. **B-C)** Dynamics of the average efficiency for the moments of increase in the efficiency of the learned behavior. The values are calculated for the best agent of the each 10-th generation. **D)** Averaged dynamics of the duration of learning. Duration of the learning is calculated as the last time step of the agent's life when one can observe activation of silent neuron.

whereby new successful policies are found through learning.

The increase in the efficiency of the learned behavior often occurs together with an increase in the average learning time of neurocontrollers – the time during which silent neurons are activating (starting from 4850-th generation on the Fig. 3D). Dynamics of the activations of silent neurons during the learning process averaged over the best 10 evolutionary runs is shown on the Fig. 4. On average about 50% of silent neurons are being activated lifelong, but most of them are being activated in the early stages of life (up to 10-th time step). However activations of silent neurons could be found even on the 100-th time step for a part of the evolutionary runs.

We found that on the early stages of evolution the efficiency of the learning correlates with the number of activated silent neurons (Fig. 5). At these stages behavior of the agents is not fully formed and learning can play significant role. As one can see on the Fig. 5 difference between efficiency of the learning agent and the same non-learning agent correlates with the number of silent neurons, which have been activated during the learning process.

Later in the course of evolution (for example in the 3500-th generation on the Fig. 3) this correlation disappears and roughly the same number of silent neurons are being activated regardless of the difference between learned and innate behavior (data is not shown). The role of learning is also changing: primarily it "corrects errors" that were made in the developmental process. That is developmental process can generate non-learning controllers as effective as learning controllers, translated from the same genome.

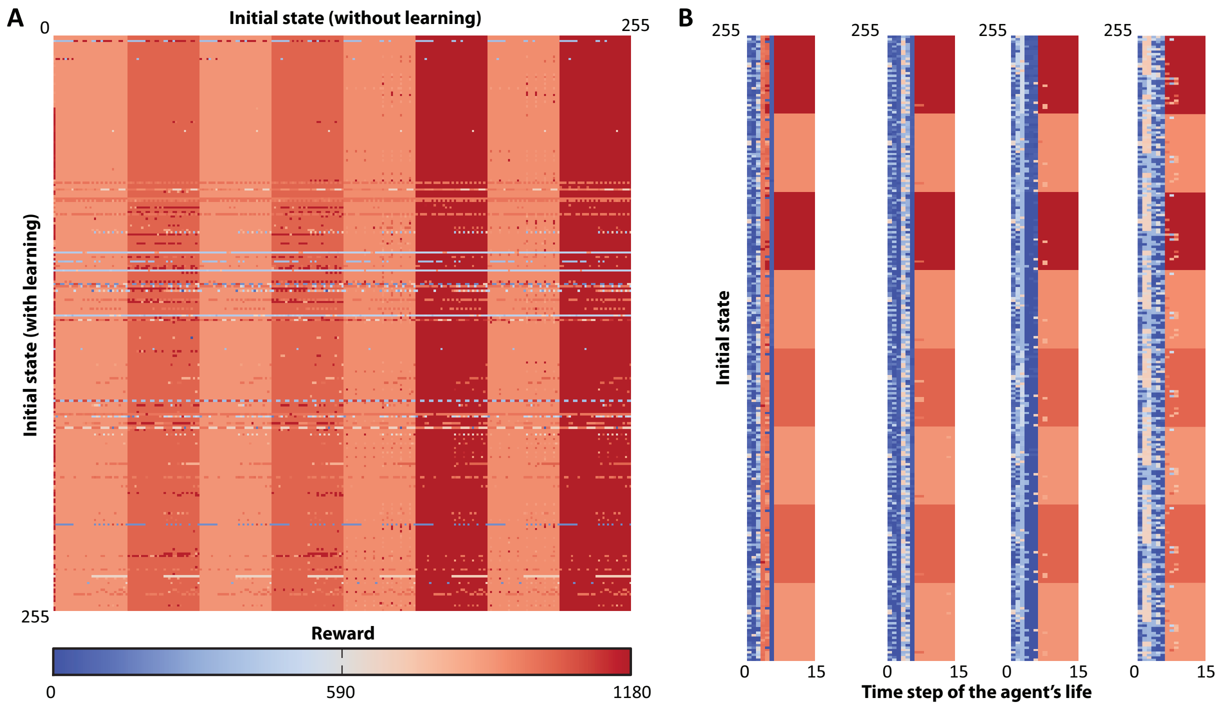

Figure 6: Efficiency of the learning process. **A)** The generality of the learning. First column represents rewards, which the best agent of the 4950-th population on the Fig. 3 with particular generated neurocontroller (see text for details) and turned-on learning acquires starting from all initial states of the environment. Each row in turn represents a rewards, which the same agent acquires with non-learning neurocontroller, which has been obtained after learning from the state corresponding to this row number. **B)** The dynamics of the learning process in terms of the efficiency for a 4 random initial states (see text for details).

We have also investigated the generality of the learning. As shown on the Fig. 6A controllers obtained after learning from one state of the environment are in the most cases also efficient when agent starts life from the different state.

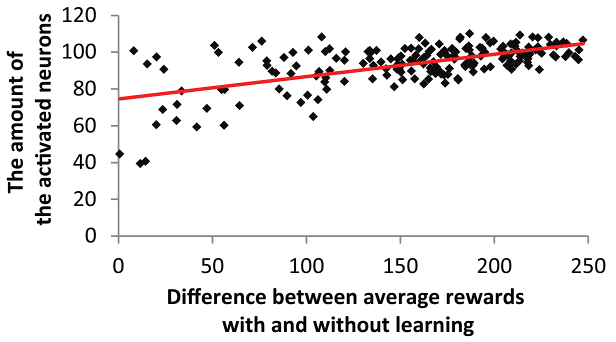

Figure 5: Correlation between the number of activated silent neurons and the quality of learning. Each dot on the graph represents reward averaged over all initial states of the environment for one development of the best agent of 1781-th generation on the Fig. 3. Spearman correlation coefficient $\rho = 0.52$ with $p-$value < 0.001

For this analysis we have chosen the agent from the later stage of evolution (the best agent of 4950-th generation) and have generated neurocontroller (via developmental process) whose reward, averaged over all initial states, when learning is disabled was significantly less than reward when learning is enabled. "Stripes" on the figure correspond to the sets of initial states with different effectiveness levels of the implementation of the same behavioral policy. This means that agent achieves the same goals in the same order, but can be more or less efficient in terms of number of performed actions.

We used the same controller as for the previous analysis to reveal the dynamics of the learning process (Fig. 6B). To do so we started agent's life from some initial state and then after each time step we evaluated efficiency of the controller, obtained so far, by running its non-learning version from all initial states. Our results demonstrate that "effective" learning occurs only at the earliest stage of agent's life during the first 6–10 time steps. These results are also consistent with average dynamics of the activation of silent neurons presented on the Fig. 4. During this early phase learning can even lead to temporary degradation of the efficiency of the intermediate controller. However further learning "stabilizes" behavior and its efficiency significantly increases.

Conclusions

References

Anokhin, P. (1974). *Biology and Neurophysiology of the Conditioned Reflex and Its Role in Adaptive Behavior*. Pergamon, Oxford.

Burtsev, M. (2008). Basic principles of adaptive learning through variation and selection. In Bullock, S., Noble, J., Watson, R., and Bedau, M. A., editors, *Artificial Life XI: Proceedings of the Eleventh International Conference on the Simulation and Synthesis of Living Systems*, pages 88–93. MIT Press, Cambridge, MA.

Edelman, G. M. (1993). Neural darwinism: Selection and reentrant signaling in higher brain function. *Neuron*, 10(2):115 – 125.

Fiorillo, C. D. (2008). Towards a general theory of neural computation based on prediction by single neurons. *PLoS ONE*, 3(10):e3298.

Gomez, F. J. and Miikkulainen, R. (1999). Solving non-markovian control tasks with neuroevolution. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'99, pages 1356–1361, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Lakhman, K. and Burtsev, M. (2013). Neuroevolution results in emergence of short-term memory in multi-goal environment. In *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO '13, pages 703–710, New York, NY, USA. ACM.

Shvyrkov, V. B. (1986). Behavioral specialization of neurons and the system-selection hypothesis of learning. In *Human Memory and Cognitive Capabilities*, pages 599–611. Elsevier, Amsterdam.