Offline and active gradient-based learning strategies in a pushing scenario

Sergio Roa and Geert-Jan Kruijff
German Research Center for Artificial Intelligence / DFKI GmbH
{sergio.roa,gj}@dfki.de[1]

Abstract.
When operating in the real world, a robot needs to accurately predict the consequences of its own actions. This is important to guide its own behavior, and in adapting it based on feedback from the environment. The paper focuses on a specific problem in this context, namely predicting affordances of simple geometrical objects called polyflaps. A machine learning approach is presented for acquiring models of object movement, resulting from a robot performing pushing actions on a polyflap. Long Short-Term Memory machines (LSTMs) are used to deal with the inherent spatiotemporal nature of this problem. An LSTM is a gradient-based model of a Recurrent Neural Network, and can successively predict a sequence of feature vectors. The paper discusses offline experiments to test the ability of LSTMs to solve the prediction problem considered here. Cross-validation methods are applied as a measure of convergence performance. An active learning method based on Intelligent Adaptive Curiosity is also applied for improving the learning performance of learners trained offline, generating a combination of learners specialized in different sensorimotor spaces after the knowledge transfer.

1 Introduction

Robots need to learn in continuously changing environments. One way to learn from the world is by interacting with objects present in it. This work is inspired by the fact that humans and animals in general are able to properly adapt to a dynamic environment. Theories of cognitive development like the theory of affordances [5] attempt to explain how creatures are able to acquire sensorimotor skills when they are faced with the different features found in the environment. For instance, surfaces afford posture, locomotion, collision, manipulation, and in general behaviour [5]. Particular objects can afford sliding, flipping, rolling behaviour. A consequence of this is that the creature should then properly predict the consequences of actions on a surface given its own body configuration. We consider here a special case of affordances learning in robots, namely that of predicting consequences of pushing simple geometrical objects called *polyflaps*. Polyflaps have been proposed to design simple learning scenarios. A polyflap is a polygon (concave or convex) cut out of a flat sheet of some material (e.g. cardboard) and folded once (anywhere) to produce a 3-D object [17], cf. Fig. 1. By combining different objects and performing different actions, we can steadily increase the complexity of the learning environment.

[1] The research reported of in this paper is supported by EU FP7 IP "CogX" (ICT-215181)

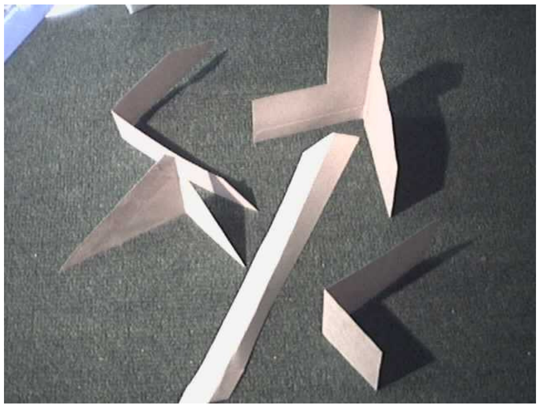
Figure 1. Polyflaps, http://www.cs.bham.ac.uk/∼axs/polyflaps/. Used here are polyflaps of the shape (bottom-right corner)

In this paper we discuss a learning scenario where a simulated robotic arm interacts with a polyflap. In the implementation we use the NVidia® PhysX™ library that allows us to perform realistic physical simulations and to obtain 3-dimensional feature vectors, so that we can easily re-adapt our algorithms to real scenarios. Although providing an idealized scenario, these experiments are necessary to establish a base line from which we can start facing noisy and incomplete features, where learning machines should be able to generalize and present outcomes in the presence of uncertainty. The learning machines we use are able to process spatio-temporal features. Specifically, we use the Long Short-Term Memory (LSTM) [7, 6] model of an Artificial Neural Network. The main objective is that the robot arm pushes the object and predicts a sequence of polyflap poses encoded as rigid body transformations during a certain time interval following the pushing action. To reduce the space- and time complexity of the problem, we select a discrete set of possible actions and starting positions for the arm to start the pushing movement. This reduction of dimensionality affords us also to evaluate and analyse more easily and carefully the learning algorithms and its corresponding results. In general, *sliding* and *flipping* affordances are obtained by applying pushing actions. The experiments show that the machines are able to model a sort of regression function that fits the data very accurate. This fact is also crucial from the point of view of dimensionality reduction, since the use of a learning machine together with its generalization abilities can highly reduce the need of storage space. Moreover, the inherent recurrent topology of these networks affords the reduction of space needed for storing spatio-temporal information.

The characteristics of these learning machines are appropriate for autonomous development of robots [10, 14]. Robots should be able to autonomously acquire sensorimotor skills by interaction with the environment. Thus, machines that are able to learn in an online and active manner need to be used. Neural Networks in general are useful for these tasks, since their weights can be updated efficiently by using one forward and one backward pass when we use gradient-based methods. However, one has to be careful with the problem of overfitting data (bias-variance tradeoff). Therefore, a sufficiently big set of samples and iterations are needed in order to generalize sufficiently well a dataset.

We tested the topology of the neural network in order to find a good compromise between computational complexity and generalization ability. For that purpose, we extracted n-fold crossvalidation sets and analyzed the average sum of squares error for all training epochs. The problem that we tackle can be regarded as a time series prediction problem approached by regression techniques. Therefore, the sum of squares error is a good performance estimation. The experiments show that the machines are able to accurately predict a feature vector, given a history of precedent feature vectors that together form a sequence. After offline training, we applied an active learning technique based on the Intelligent Adaptive Curiosity algorithm [10, 14], by including an additional set of actions in order to test the autonomous generation of different regions in the sensorimotor space that allows an active selection of samples via maximization of a measure of learning progress and multiple learners specialized in each region. We also show that the generalization is improved by the set of machines (which are only “biased” for their corresponding regions).

This paper is organized as follows. In the next section, we present a current state of the art related to affordances learning in robots and recurrent neural networks. In section 3, we describe the learning scenario and the features we used for training LSTMs. In section 4 we present the offline learning mechanism and architecture employed. In section 5 we show and explain experimental results for offline experiments with LSTMs. In section 6 we explain the active learning mechanism and results and in section 8 we present some concluding remarks and planned work.

2 Related Work

Affordances learning has been introduced in the field of robotics in recent years. The reason is that aiming to autonomous behaviour in robots requires an inspiration from biological cognitive systems, which are very successful on acquiring sensorimotor skills by their own means. As a cognitive science theory, the field was introduced by the perceptual psychologist J.J. Gibson [5]. An affordance in this sense is a resource or support that the environment offers an agent for action, and that the agent can directly perceive and employ. From the robotic perspective, this concept implies that the robots should be able to predict consequences of actions given certain object features and robotic embodiment.

In the field of robotics, a compilation of works related to affordance-based robot control can be found in [15]. A similar approach to the one presented in this work is described in [12]. In that work, labels of object/action pairs and 11 features encoding the action performed and the object behaviour are used to train Self Organizing Maps. In this way, they cluster this space and map the features to the target function represented by such labels. Pushing actions were performed on different objects in a real environment. Other approaches have used also similar features and learning methods and have studied different kinds of affordances [2].

Perception of affordances has also been addressed with reinforcement learning techniques. In [11], the robot performs different learning stages starting from recognizing affordances and finally accomplishing some task given the affordances that the robot has already acquired in earlier stages. In that work, liftable vs. non-liftable objects are recognized and Markov Decision Processes are used for the goal-based task. In [10, 14] the robot autonomously enters different stages of development by interacting with objects or performing some action, which is selected according to a measure of “interestingness”. Thus, robots are intrinsically motivated to perform actions that offer an opportunity to learn according to an estimation of learning progress calculated from prediction error histories.

However, we can consider that these aproaches use a kind of short-term memory or mapping approach that does not take into account the spatio-temporal processing of data when an action is performed in a given time interval. Moreover, in some approaches there is an explicit labelling of the recognized affordances or the robot has no means to evaluate the accuracy of its predictions. In order to evaluate the abilities of learning machines in processing a series of features like rigid body transformations that gives us a more accurate assessment of the object poses and behaviour, we are using recurrent neural networks that are known to process sequences and obtain proper generalizations by infering regression functions.

A simulated scenario using also polyflaps is described in [9]. The authors formalise the learning problem in a probabilistic framework. Explicit 3D rigid body transformations are predicted by that models and they are tested against novel objects similar in shape to polyflaps.

Long Short-Term Memory machines have been used for problems like time-series prediction, sequences classification, phoneme classification, reinforcement learning, among others [7, 6, 1]. They are appropriate to handle long-term dependencies in data sequences. Therefore, they seem to have a high potential to be used in learning tasks where compositionality and conditional dependencies of events or states is encountered through a relatively broad time period.

The work described in this paper is a follow-up of the one presented in [13].

3 Learning Scenario

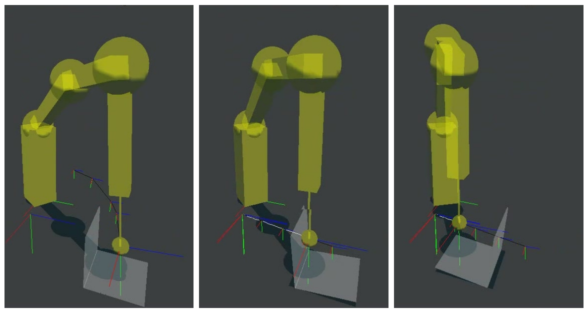

Figure 2. Learning scenario with a polyflap

The learning scenario is shown in Fig. 2. The simulated arm corresponds to a Neuronics® Katana 6M$^{\text{TM}}$ arm with a ball as a simple finger. In order to simulate a pushing action we apply a linear trajectory over a specified time period until it reaches the desired pose. The arm has 6 joints, including the last joint for the finger which is static. The representation of object poses are in Euler angles with respect to a reference frame which is the origin in the scene (6-D pose).

The features corresponding to the arm are a starting 6-D pose vector for the end-effector $\mathbf{e}_0$, and a real value denoting a direction angle Θ ranging from 60 to 120 degrees, parallel to the ground plane in the direction to the center of the standing polyflap side. Together, these features form the motor command feature vector denoted as $\mathbf{m}$. The values are all normalized to obtain vectors with mean 0 and standard deviation 1.0. A 6-D pose vector corresponding to the polyflap pose is denoted as $\mathbf{p}_t$ at time t. The pose $\mathbf{p}_0$ is fixed for all experiments.

Then, the concatenation $\mathbf{f}_0 = [\mathbf{m}\ \mathbf{e}_0\ \mathbf{p}_0]$ represents the feature vector to be fed initially to the neural network. The subsequent feature vectors fed to the machine have the form $\mathbf{f}_t = [\mathbf{0}\ \mathbf{e}_t\ \mathbf{p}_t]$, where the size of $\mathbf{0}$ is the size of $\mathbf{m}$. This representation affords the learning machine to attain a better convergence.

During the execution of the arm path, we obtain a series of poses $\langle \mathbf{p}_t, \mathbf{e}_t \rangle$ to construct a feature vector $\mathbf{f}_t$. We extract then n polyflap and effector poses and finally we build a sequence set $S = \{\mathbf{f}_{t=1}^n\}$. So, a particular sequence set (an instance) is used in each iteration of the experiment to be fed to the LSTM in $n + 1$ steps. For the time step t, a training tuple $\langle \mathbf{f}_t, \mathbf{t}_t \rangle$ is used for the neural network learning procedure, where the feature vector $\mathbf{f}_t$ represents the input vector and $\mathbf{t}_t = \mathbf{p}_{t+1}$ the target (predicted) vector encoding the predicted polyflap pose.

This representation then encodes the rigid body transformations of polyflap and effector through these n steps and also encodes the given robot control command that performs the pushing movement. In order to discretize and reduce the dimensionality of the task, we only used a discrete number of different starting positions for the arm to start the pushing movement.

4 Offline Learning method

The learning process used for training LSTMs with the features described in section 3 is described here. As mentioned in the previous section, a dataset $\mathcal{D}$ containing a certain quantity of sequences S_i is obtained and we perform offline experiments with these data.

A LSTM machine is usually composed of an input layer, a hidden layer and an output layer. In general, recurrent neural networks can have recurrent connections for all their neurons. In particular, in this work we only use recurrent connections for the hidden layers. We also made preliminary experiments with networks with no recurrent connections and we found less performance. The LSTM [7, 6, 1] architecture was developed in order to solve some learning issues in recurrent neural networks related to long-term dependencies learning. These problems sum up to the problem that errors propagated back in time tend to either vanish or blow up. This is known as the problem of vanishing gradients.

LSTM's solution to this problem is to enforce *constant* error flow in a number of specialized units, called Constant Error Carrousels (CECs), corresponding to those CECs having linear activation functions not decaying over time. CECs avoid to transmit useless information from the time-series by adding other input gates that regulate the access to the units. Thus, they learn to open and close access to the CECs at appropriate moments. Likewise, the access from the CECs to output units is controlled by multiplicative output gates and they learn in a similar way how to open or close the access to the output side. Additionally, forget gates [3] learn to reset the activation of the CECs when the information stored in them is no longer useful, i.e., when previous inputs need to be forgotten. The combination of a CEC with its associated input, output and forget gate is called a memory cell, as depicted in Fig. 3. Other additions are peephole weights [4], which improve the LSTM's ability to learn tasks that require precise timing and counting of internal states, and bidirectional connections [16].

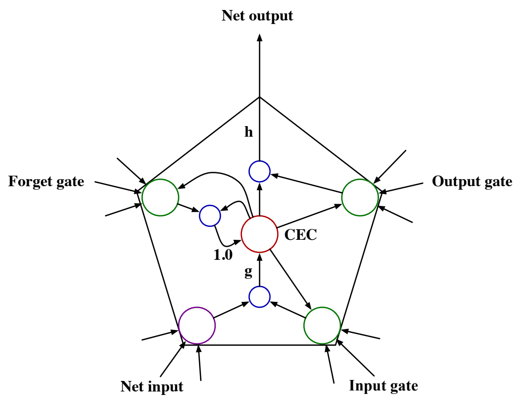

Figure 3. LSTM memory block with one cell. The internal state of the cell is maintained with a recurrent connection of fixed weight 1.0. The three gates collect activations from inside and outside the block, and control the cell via multiplicative units (small circles). The input and output gates scale the input and output of the cell while the forget gate scales the internal state. The cell input and output activation functions (g and h) are applied at the indicated places [6].

In this work, we used 10 memory blocks in the hidden layer, which was found to be a good compromise between computational complexity and convergence.

When some input vector is fed to the network, the forward pass is calculated as follows. Let us denote an output neuron (unit) activation y^o, an input gate activation y^{in}, and output gate activation y^{out} and a forget gate activation y^f. Then, for the time step t each of them are calculated in the following standard way:

$$y^i(t) = f_i(\sum_j w_{ij} y^j(t-1)), \tag{1}$$

where w_{ij} is the weight of the connection from unit j to unit i, and f the activation function. In this paper, we only consider one CEC activation (one cell) for each memory block. The CEC activation s_c for the memory cell c is computed as follows:

$$s_c(t) = y^{f_c}(t) s_c(t-1) + y^{\text{in}_c}(t) g(\sum_j w_{cj} y^j(t-1)), \tag{2}$$

where g is the cell input activation function. The memory cell output is then calculated by

$$y^{s_c}(t) = y^{\text{out}_c}(t) h(s_c(t)), \tag{3}$$

where h is the cell output activation function. The backward pass is a steepest (gradient) descent method which updates the weights of the different types of units. Consider a network input $a_j(t)$ to some unit j at time t. In general, the gradient is defined as:

$$\delta_j(t) = \frac{\delta E}{\delta a_j(t)}, \tag{4}$$

where E is the objective (error) function to be minimized and used for training. For a detailed explanation of the backward pass equations for each unit type cf. [6]. Since we are dealing with a regression problem, we consider the sum of squares error as a performance measure. The error function is defined as:

$$E_t = \frac{1}{2K} \sum_i (y_i - y_i')^2, \tag{5}$$

where K is a normalization factor which depends on the size of each sequence n_i and the total number of sequences in the dataset k. y_i is the output unit activation and y_i' is the expected value. The learning process is described in the Algorithm 1.

```
Data: A dataset $\mathcal{D}_1$ containing $k$ sequences of variable size $n_i$
      for training. A dataset $\mathcal{D}_2$ containing $z$ sequences of size
      $n_j$ for testing.
Result: An LSTM machine after error minimization.
Nr. of epochs $ep = 0$.
repeat
    for $i$=1 to $k$ do
        for $j$=1 to $n_i$ do
            Input: Present training tuple $\langle \mathbf{f}_{ij}, \mathbf{t}_{ij} \rangle$ ($j$th forward
                   pass step).
        end
        Calculate error $e_i$ associated to current training
        sequence $S_i$.
        Backward pass.
    end
    Evaluate error $E_t$ with the test set $\mathcal{D}_2$.
    Epoch $ep = ep + 1$.
until No new network found with lowest error after 20 epochs ;
```

Algorithm 1: Offline learning process

For the purpose of calculating the number of training sequences that are necessary so that convergence improves, we generated n-fold cross-validation sets. We split a dataset $\mathcal{D}$ into n disjoint sets of equal size that are used for testing. We used the remaining data for training n different networks.

5 Experimental results for Offline Learning

In order to test the convergence of LSTMs we used 10-fold cross-validation sets for three different dataset sizes, namely 100, 200 and 500. That allowed us to estimate the approximate number of samples that are needed to learn with high precision the prediction task.

In Fig. 4 a comparison of the average sum of squares error (SSE) and SSE standard deviation is shown. In this case, the SSE is averaged among all the cross-validation sets. The picture shows that the SSE is considerably reduced when more samples are used, as expected, and likewise the standard deviation of the SSEs.

6 Active Learning

The active learning procedure is based on the work of Oudeyer et al. [10] about Intrinsic Motivation Systems. The general idea of the Intelligent Adaptive Curiosity (IAC) algorithm is that a meta-learning system samples a set of actions and selects one that maximizes the learning progress, which is a measure based on the difference between smoothed current and previous mean error quantities. The learning progress $\mathcal{L}_r$ is associated to a region $\mathcal{R}_r$ in the sensorimotor space. Starting with one region, successive regions are obtained by splitting the sensorimotor space depending on a measure of variance in the dataset $\mathcal{D}_r$ (exemplars used for Region $\mathcal{R}_r$). This division is performed after $|\mathcal{D}_r|$ achieves a certain threshold κ. A dataset $\mathcal{D}_r$ for a Region R_r is split in two datasets $\mathcal{D}_{r+1}, \mathcal{D}_{r+2}$ (for regions $\mathcal{R}_{r+1}, \mathcal{R}_{r+2}$). Let us denote

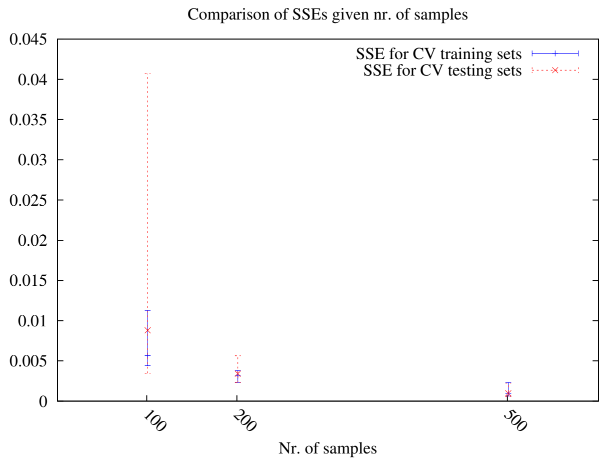

Figure 4. SSEs are reduced when increasing the dataset size.

$$\mathcal{D}_r = \{S_i\}$$

the set of instances in region $\mathcal{R}_r$. Then the split of $\mathcal{D}_r$ defined by the index c with value v_c is performed when the following criterion (Γ) is met:

- all the instances S_i of $\mathcal{D}_{r+1}$ have the cth component of their motor command vector $\mathbf{m}_i$ smaller than v_c.
- all the instances S_i of $\mathcal{D}_{r+2}$ have the cth component of their motor command vector $\mathbf{m}_i$ greater than v_c.
- the quantity $|\mathcal{D}_{r+1}| \cdot \sigma(\{[\mathbf{e}_{ij} \;\; \mathbf{p}_{ij}]_{j=1}^{n_i} \in \mathcal{D}_{r+1}\}) + |\mathcal{D}_{r+2}| \cdot \sigma(\{[\mathbf{e}_{ij} \;\; \mathbf{p}_{ij}]_{j=1}^{n_i} \in \mathcal{D}_{r+2}\})$ is minimal, where

$$\sigma(\mathcal{S}) = \frac{\sum_{v \in \mathcal{S}} \|v - \frac{\sum_{v \in \mathcal{S}} v}{|\mathcal{S}|}\|^2}{|\mathcal{S}|}$$

where $\mathcal{S}$ is a set of vectors.

Each region stores all cutting dimension and values that were used in its generation as well as in the generation of its parent regions. For the region $\mathcal{R}_r$ a learning machine $\mathcal{M}_r$ is stored, and this machine is inherited by the child regions. The learning process is described in the Algorithm 2.

```
Data: An initial region $\mathcal{R}_0$ which encompasses the whole sensorimotor space.
Result: A set of regions $\{\mathcal{R}_r\}$ with corresponding LSTM machines $\{\mathcal{M}_r\}$.
for $i$=1 to $I$ do
    Choose a motor command action $\mathbf{m}_{r,i} = \arg\max_{\mathbf{m}\in\{R_r\}}\{\mathcal{L}_{r,i}\}$ among all current regions $\{\mathcal{R}_r\}$ by using a near to greedy policy with probability 0.3.
    if $\kappa$ then
        Split region $\mathcal{R}_r$ into $\mathcal{R}_{r+1}$ and $\mathcal{R}_{r+2}$ according to $\Gamma$.
    end
    Calculate error $e_{r,i}$ associated to current training sequence $S_{r,i}$.
    Update the machine $\mathcal{M}_r$ with a forward and backward pass.
    Calculate smoothed mean error $\varepsilon_{r,i+1}$ and $\varepsilon_{r,i+1-\tau}$ with a window parameter $\tau$ and a smoothing parameter $\theta$.
    Calculate the decrease in the mean error rate $\Delta_{r,i+1} = \varepsilon_{r,i+1} - \varepsilon_{r,i+1-\tau}$.
    Calculate the learning progress $\mathcal{L}_{r,i+1} = -\Delta_{r,i+1}$.
end
```

Algorithm 2: Active learning process

7 Experimental Results for Active Learning

In order to test the active learning mechanism, the main idea is to train offline a LSTM with a subset of all possible starting positions producing a partial set of actions and thus a dataset $\mathcal{D}_0 \subset P_0$, where P_0 is the sensorimotor manifold encompassing D_0. Then, we use this machine in the active learning loop allowing additional actions, so that at the end we generate a dataset $\mathcal{D}_1 \subset P_1$, where $P_0 \subset P_1$. The hypothesis is that the algorithm will start producing more frequently actions corresponding to the sensorimotor regions associated to the new actions.

Thus, we first trained offline a LSTM with a subset of possible starting positions for the arm movement and a number of sequences equal to 500. This generates the dataset $\mathcal{D}_0$. When initializing the active learning procedure, we allowed all possible starting positions for the arm movement. Then, we initialize the region R_0 with the already trained machine $\mathcal{M}_0$ that introduces better generalization performance according to the cross-validation sets. We apply a maximum number $I = 300$ of iterations, after which a new dataset $\mathcal{D}_1$ is generated. Then, we merge the datasets into a set $\mathcal{D} = \mathcal{D}_0 \cup \mathcal{D}_1$. We use the set $\mathcal{D}$ to test the errors of the machine trained offline and the ensemble of machines trained via active learning. The results are shown in Table 1.

Table 1. An ensemble trained via IAC against an offline trained machine.

Machine	Avg SSE
Offline	0.4251
Active	0.211991

The unique observation here is that the generalization performance is improved by using the new active learner, which is a expected result. In order to check the hypothesis presented above, we analysed the learning progress of the ensemble of machines created after splitting the sensorimotor space in different regions.

As expected, the algorithm starts to select very frequently actions that are new or "interesting". In Fig. 5, we can observe the frequency of actions generated from each set of starting positions for a window of 20 iterations. For instance, from index ∼150 to ∼250 the new set of actions are more frequent. This result also confirms the generation of different stages of development that the IAC algorithm produces [10]. We make the same observation for a specific region (Fig. 6). In Fig. 7 the curves of learning progress and error for the corresponding region are shown. We can observe that the learning progress curve rises and the error drops.

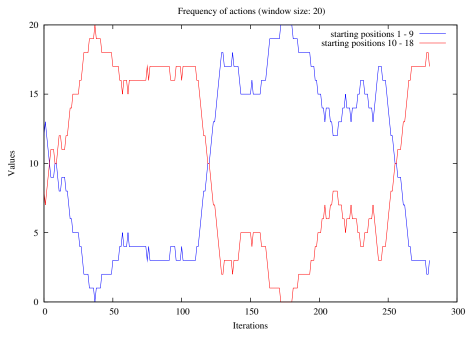

Figure 5. Frequency of actions in the experiment.

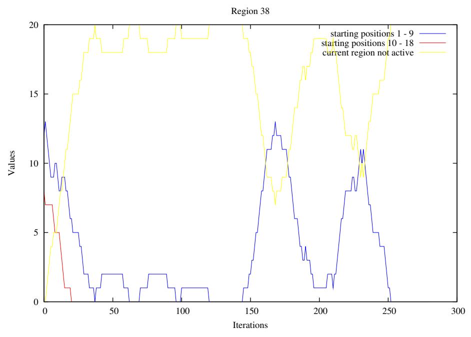

Figure 6. Frequency of actions for a specific region (window size: 20).

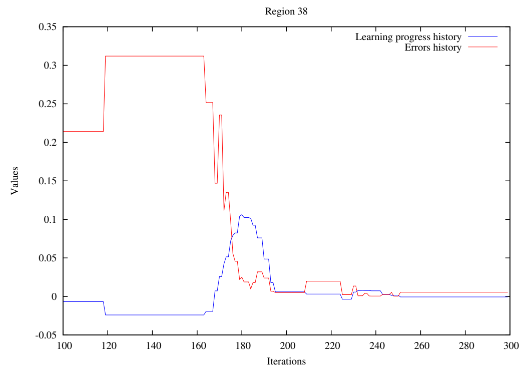

Figure 7. Learning progress increase for a certain region.

In Figures 8,9 the prediction ability of a learning machine over a sequence is illustrated.

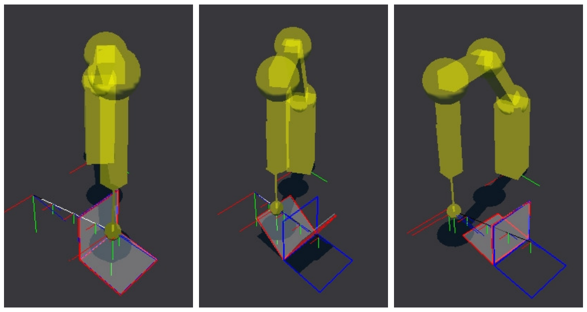

Figure 8. Prediction of the flipping affordance. The blue polyflap is the first predicted polyflap in the current sequence and the red one the last predicted one.

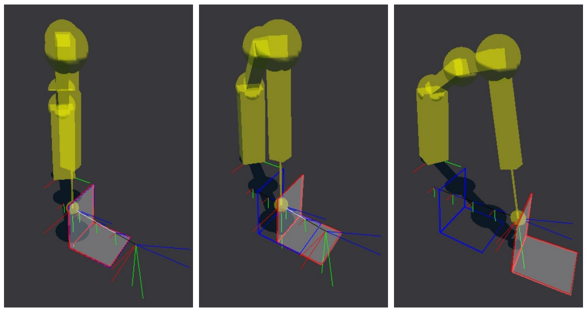

Figure 9. Prediction of the sliding affordance.

8 Conclusions and future work

The experiments shown in this paper demonstrate the ability of recurrent neural networks, in particular Long Short-Term Memory machines to approximate a regression function encoding the trajectory of simple geometrical objects when pushing actions are performed. Therefore, these machines are useful for predicting the affordances of pushing actions. We used 3-dimensional features and realistic simulations that we can then apply to real environments. Sequences of finger effector and polyflap poses were used to feed the LSTMs, showing the capacity of LSTM for prediction in relatively large time periods. The offline experiments showed great accuracy in prediction. The use of an active learning mechanism where machines are specialized in different parts of the sensorimotor space was also tested. The selection of actions is performed via a measure of learning progress that improves generalization.

In this work, the motivation to select an action via active learning is mainly based on the curiosity-driven mechanism introduced by the IAC algorithm. This mechanism forces the robot to select actions that maximize a learning progress measure. This encourages the reduction of error for sensorimotor regions that are still not accurately learned. The effectivity of the IAC-based strategy for assessing learning progress in sensorimotor regions with spatio-temporal features is confirmed. Moreover, machines that take into account spatio-temporal information fit well into the active learning loop. However, the gradient-based method for updating the networks still makes the process slow, so that many iterations are needed to observe high improvements. It is possible to add additional drives or measures for selecting actions in order to have different strategies for accelerating the learning progress. Additionally, alternative algorithms for LSTM training may also be considered.

The CrySSMEx[8] algorithm has been used to analyse recurrent networks as dynamical systems by using a conditional Entropy based method that extracts a probabilistic automaton associated to a machine. This method might be useful for active learning, because it represents uncertainty and predictability during the processing of spatio-temporal features.

REFERENCES

[1] Bram Bakker, ‘Reinforcement learning with long short-term memory’, in *Advances in Neural Information Processing Systems 14*, pp. 1475–1482. MIT Press, (2002).

[2] Paul Fitzpatrick, Giorgio Metta, Lorenzo Natale, Sajit Rao, and Giulio Sandini, ‘Learning about objects through action - initial steps towards artificial cognition’, in *Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3140–3145, (2003).

[3] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins, ‘Learning to forget: Continual prediction with lstm’, *Neural Computation*, **12**, 2451–2471, (1999).

[4] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber, ‘Learning precise timing with lstm recurrent networks’, *J. Mach. Learn. Res.*, **3**, 115–143, (2003).

[5] J. J. Gibson, ‘The theory of affordances’, in *Perceiving, Acting, and Knowing: Toward an Ecological Psychology*, eds., R. Shaw and J. Bransford, 67–82, Lawrence Erlbaum, (1977).

[6] Alex Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, Ph.D. dissertation, Technische Universität München, July 2008.

[7] Sepp Hochreiter and Jurgen Schmidhuber, ‘Long short-term memory’, *Neural Computation*, 1735–1780, (1997).

[8] H. Jacobsson, ‘The crystallizing substochastic sequential machine extractor - CrySSMEx’, *Neural Computation*, **18**(9), 2211–2255, (2006).

[9] M. Kopicki, J. Wyatt, and R. Stolkin, ‘Prediction learning in robotic pushing manipulation’, in *Proceedings of the 14th IEEE International Conference on Advanced Robotics (ICAR 2009)*, Munich, Germany, (June 2009).

[10] P-Y. Oudeyer, F. Kaplan, and V. V. Hafner, ‘Intrinsic motivation systems for autonomous mental development’, *IEEE Transactions on Evolutionary Computation*, **11**(1), (2007).

[11] L. Paletta, G. Fritz, F. Kintzler, J. Irran, and G. Dorffner, ‘Learning to perceive affordances in a framework of developmental embodied cognition’, in *Development and Learning, 2007. ICDL 2007. IEEE 6th International Conference on*, pp. 110–115, (July 2007).

[12] B. Ridge, D. Skočaj, and A. Leonardis, ‘A system for learning basic object affordances using a self-organizing map’, in *International Conference on Cognitive Systems CogSys 2008*, Karlsruhe, Germany, (2008).

[13] Sergio Roa and Geert Jan Kruijff, ‘Long short-term memory for affordances learning’, in *Proceedings of the 9th International Conference on Epigenetic Robotics*, eds., Lola Caamero, Pierre-Yves Oudeyer, and Christian Balkenius, number 146 in Lund University Cognitive Studies, pp. 235–236. o.A., (11 2009).

[14] Sergio Roa, Geert Jan Kruijff, and Henrik Jacobsson, ‘Curiosity-driven acquisition of sensorimotor concepts using memory-based active learning’, in *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*, pp. 665–670, (2008).

[15] *Towards Affordance-Based Robot Control - LNAI 4760*, eds., Erich Rome, Joachim Hertzberg, and Georg Dorffner, Lecture Notes in Computer Science LNAI 4760, Springer Verlag, Berlin, Germany, 2008.

[16] M. Schuster and K.K. Paliwal, ‘Bidirectional recurrent neural networks’, *Signal Processing, IEEE Transactions on*, **45**(11), 2673–2681, (Nov 1997).

[17] Aaron Sloman, ‘Polyflaps as a domain for perceiving, acting and learning in a 3-D world’, in *Position Papers for 2006 AAAI Fellows Symposium*, Menlo Park, CA, (2006). AAAI.