Learning Android Control using Growing Neural Networks

Heni Ben Amor, Shuhei Ikemoto, Takashi Minato and Hiroshi Ishiguro
Department of Adaptive Machine Systems
Osaka University,
Osaka, Japan,
Email: {amor,ikemoto,minato,ishiguro}@ed.ams.eng.osaka-u.ac.jp

***Abstract*— Fixed sized neural networks are a popular technique for learning the adaptive control of non-linear plants. When applied to the complex control of android robots, however, they suffer from serious limitations, such as the moving target problem i.e. the interference between old and newly learned knowledge. To overcome these problems, we propose the use of growing neural networks in a new learning framework based on the process of consolidation. The new framework is able to overcome the drawbacks of sigmoidal neural networks, while maintaining their power of generalization. In experiments the framework was successfully applied to the control of an android robot.**

I. Introduction

Neural networks have proved to be powerful and popular tools for controlling robots with many degrees of freedom. Particularly, their ability to learn and capture non-linear dependecies between control variables has contributed to this popularity. For example, using a process called "Feedback-error learning" [2] the neural network can learn to predict the effect of issued motor commands. After learning, the network represents an inverse model of the robot's dynamics which enables it to generate optimal actions for control by cancelling out any undesired effects.

Recently, some critics arouse about the use of sigmoidal neural networks (neural networks with sigmoidal kernel) for robot control. For example, Vijayakumar and colleagues [6] identified the following drawbacks of sigmoidal neural networks:

- The need for careful network structure
- The problem of catastrophic forgetting
- The need of complex off-line retraining

The first point refers to the complex relationship between the network structure and it's ability to learn particular functions. It is well known in the machine learning community that particular functions (for instance XOR) cannot be learned if the network structure is not appropriately chosen. The problem of catastrophic forgetting refers to the interference between old and newly learned knowledge. When been sequentially trained on two problems *A* and *B*, it is often found that the network will have forgotten most of it's knowlede of problem *A* after training on problem *B*. Due to this drawback it is particularly difficult to employ neural networks in incremental learning applications. However, if we really want to achieve lifelong learning robots [5], it is of premordial necessity that robots are able to aquire new knowledge without forgetting previously learned information. According to [6] complex off-line learning is needed to overcome the problem of catastrophic forgetting, which "from practical point, (this approach) is hardly useful". Due to these identified drawbacks Vijayakumar et al. propose the use of learning techniques based on local receptive fields, in which the above problems do not seem to appear. However, due to their local nature such techniques have limited ability to generalize.

In contrast to this, we try to profit from the sigmoidal neural networks ability to generalize, while minimizing the risk of forgetting knowledge. We do this by adressing the above three problems and presenting ways to solve them. In the following Section II we will introduce the process of consolidation and present a computational framework for motor learning based on consolidation. Then in Section III we introduce the concept of growing neural network. In Section IV we present results from a first implementation of this framework on an android arm.

II. Consolidation Learning

All problems mentioned above in relation to sigmoidal neural networks can interestingly also be found in humans. For example in [3] it was demonstrated that two motor tasks may be learned and retained, only if the training sessions are seperated by an interval of approximately 6 hours. Otherwise, learning the second task leads to an unlearning of the internal model of the first task. However, if sufficient temporal distance is taken between the tasks, then the corresponding motor skills are retained for long time (at least 5 months after trainig). The reason for this is a process called *consolidation*. During training, the acquired information is first stored in the prefrontal regions of the cortex. In this phase the information is still fragile and sucseptible to behavioral interference. Then, approx. 6 hours after trainig, consolidation shifts the information to the premotor, posterior and cerebellar cortex in which the information is stored for a long period of time without being disrupted. In this process the brain engages new regions to perform the task at hand. Thus the number of neurons needed for a particular task is adaptively figured in the consolidation process.

Although humans share similar problems with sigmoidal neural networks, we are still able to learn a variety of different tasks, and recall information which was acquired a long time ago. Obviously, this raises the question whether a computational implementation of this process (or a simplified version of the latter) can make sigmoidal neural networks attractive again for robot control.

A. A Computational Framework for Consolidation Learning

To answer the above questions we propose a new computational framework for motor learning in which consolidation can be easily incorporated. Figure 1 shows the components and the working of this new framework.

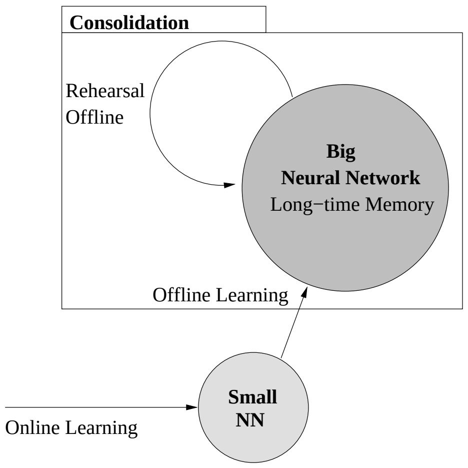

Fig. 1. Consolidation learning framework. A new problem is first learned by a small neural network. The information gained from this NN is then used to teach a big neural network in an offline process involving rehearsal of old information.

The most striking feature of the proposed consolidation framework is the separation between short-time memory (represented by the small neural network) and the long-time memory (represented by the big neural network) of the system. Each new task or subtask is first learned online by the small neural network. Learning is continued until the network becomes an expert in that particular task. Once this online phase is finished the expert network teaches newly acquired knowledge to the big neural network in the consolidation phase. This process is done offline. To ensure that the network has enough memory capacity to learn the new knowledge, learning is performed using a growing neural network technique (see Section III). At the same time, old knowledge contained in the big neural network is rehearsed in order to avoid catastrophic forgetting.

Algorithm 1 illustrates the coarse flow of consolidation learning. Here, the set $T = \{t_1, ..., t_k\}$ refers to the k subtasks to be sequentially learned. The number β is a biasing parameter, which biases learning towards new or towards old information. If β equals zero, then the network will mainly try to remember old knowledge without learning new information. Otherwise, if β equals one, then the network will eventually forget all previous knowledge. Setting this value to 0.5 is a results in a good tradeoff. Finally, the number n represents the size of the training set to be used during consolidation.

Algorithm 1 Pseudocode for consolidation learning $i = 0$

```
1: Learn new task $t_i$ unsupervised learning (online)
2: Create set $A$ with $n * \beta$ supervised data from small neural network
3: Create set $B$ with $n * (1 - \beta)$ supervised data from big neural network
4: Combine supervised data $A \cup B$
5: Learn $A \cup B$ by using the big neural network and growing technique (offline)
6: $i \leftarrow i + 1$
7: GOTO 1
```

The most critical point of this algorithm is the reinstatement, e.g. the creation of supervised data used for rehearsal. One way to create the supervised data would be to store the old input vectors which were previously used for learning. By computing the output of the big neural network for each of these input vectors, we get input-output pairs. These pairs can then be used as a training set during rehearsal. The drawback of this approach is that it requires storing input data of all training phases so far. Another approach to reinstatement is to use random input vectors when computing the rehearsal training set from the big neural network. On one hand, this has the advantage of getting rid of additional data to be saved. On the other hand, it might detoriate the performance of consolidation. A random input might lie in an area of the input space, for which the long-time memory did not get any training data so far. In such a case, the rehearsed pattern might conflict with a newly learned pattern from the small neural network. While in this paper we are mainly using the former approach (stored input data), we are currently investigating the use of randomly reinstated information for rehearsal.

B. Advantages of Consolidation

In contrast to the considerations of [6], the approach using offline consolidation is found to be highly effective. It enables us to benefit from the sigmoidal neural networks power of generalization. The consolidation framework also enables us to draw a neat line between the problem of learning new information and its incorporation into previously acquired knowledge. In the proposed framework, however, the incorporation step is done in an offline process. As such it can be much faster, as it is not dependent on the robot (or other time consuming ressources) anymore. Another interesting point is the fact that the small neural network acts as filter between the environment and the long-time memory. Only when the online phase is finished and a good local approximation of the current task is found, the long-time memory is retrained. As a result the big-neural network is freed from performing random walk cycles, which are needed for exploring a new task. Finally, by mimicking the human way of motor learning, consolidation makes the process of robotic motor learning more realistic from a cognitive science point of view.

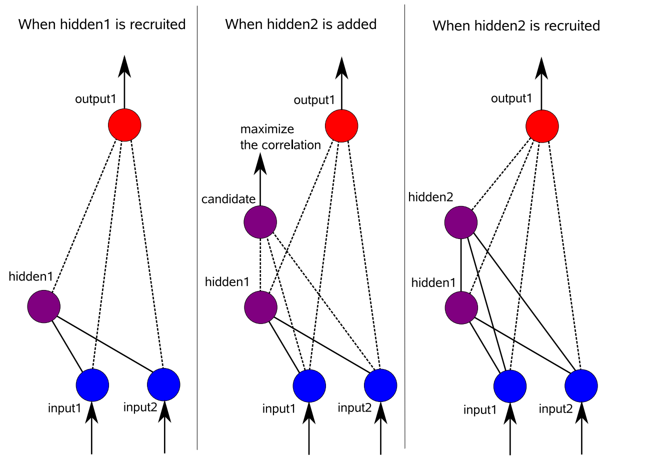

Fig. 2. Different desired and executed trajectories of the android arm after consolidation learning is finished. Time is given in microseconds.

III. Growing Neural Networks

Instead of having a human determining the number of hidden neurons and layers, growing neural network methods determine the network structure adaptively according to the problem difficulty. In doing so, they also reduce the effect of catastrophic forgetting, as new neurons are allocated everytime new data is learned. This adaptive memory capacity removes the necessity of overwriting previously learned knowledge. Still, if used in a naive way, growing neural networks run into the danger of overwriting acquired knowlede. In this section, we will first introduce two of the most successfull existing techniques used for growing neural network architectures.

A. *Cascade Correlation*

In this chapter, we summarize the Cascade-Correlation learning architecture, as formulated in [1]. Cascade-Correlation is a learning method for growing neural network architectures. Cascade-Correlation starts learning with minimal structure, and then adds new hidden unit while maximizing correlation between output of the new hidden unit and outputs of output units. Maximizing correlation indicates maching the direction of primary error components and output of new hidden unit each time.
Additionally, we freeze its input weights when it is recruited to the network. Eventually, the neural network obtains deep structure with constructed hidden units which have permanent input weights and receive input connections from all previous layer's units. Thus, hidden units become permanent feature-detectors in the network, and the newly added unit indicates a new detector that is of higher order than previously added ones.
For example, Figure 2 illustrates the structure of the neural network when 2 hidden units are added. The cascade-correlation can solve two principal problems of backpropagation learning, namely the “Moving Target Problem” and “Step Size Problem” because of specialized and fixed hidden units and its deep structure.

B. *Neuroevolution*

The term Neuroevolution refers to evolving the topology and weights of a neural network using a genetic algorithm. One of the most successful algorithms for neuroevolution is the NEAT algorithm (Neuro Evolution on Augmenting Topologies) [4]. NEAT first starts with a population of minimal structure netoworks, e.g. networks with zero hidden neurons. Each network is represented by a genome which holds all necessary information about structure and weights. In each generation mutation operators acting on the genomes successively add new hidden neurons and connections between neurons. At the same time, mutation also changes the weights of the network. Additionally, a crossover operator makes sure that two “good” networks can mate in order to produce an even better child. Using historical markings NEAT is able to crossover two genomes which have entirely different structure. Finally, a speciation operator protectes promising genetic material from vanishing from the population.

IV. Experiments & Result

A. *Android Robot*

In this chapter, we introduce the robotic platform for our experiment and explain it's characteristics. The employed platform is an android; a humanoid robot with a realistic human-like appearance and skin. The android is developed by the company Kokoro Ltd. Only upper-body is actuated and has 43 DOF. Each joint of the android is actuated by pneumatic actuators e.g. a pneumatic cylinder and motor. In our experiments we only use the Arm of the android which has 5 DOF. The experiment which is assembled by the arm, airflow control valves, regulator, and compressor, is shown in Figure 3. In the beginning, the compressor generates high pressured air to actuate the Arm. Secondly, the pressure oscillation and extra moisture which are contained in high pressured air, are removed by the regulator. Finally, the air which is controled by airflow control valves actuates the arm. The compressibility of the air can contribute to safe operation of the android. This advantage is important for a communication robot which aims at interacting with human partners. The arm has strong nonlinearity which is caused by pneumatic actuators and construction of joints. The reasons of nonlinearity which comes from the pneumatic actuators, can be explained by the compressibility of air and the friction of sliding parts in the actuators. Usually actuators in robots are used in combination with a decelerator, to cancel the nonlinear term such as coriolis effect in the robot's dynamics. In contrast, pneumatic actuators are used without decelarators and can therefore not cancel out the nonlinear term. As a result its control is difficult.

B. *Experiment*

In order to evaluate the performance of the new consolidation framework, the following experiment is set up. Three different reference trajectories are incrementally used for learning. The robot arm has to learn to follow these reference trajectories by minimizing the error. In order to have easy analysis of the results, only 1 DOF was used for the experiment.

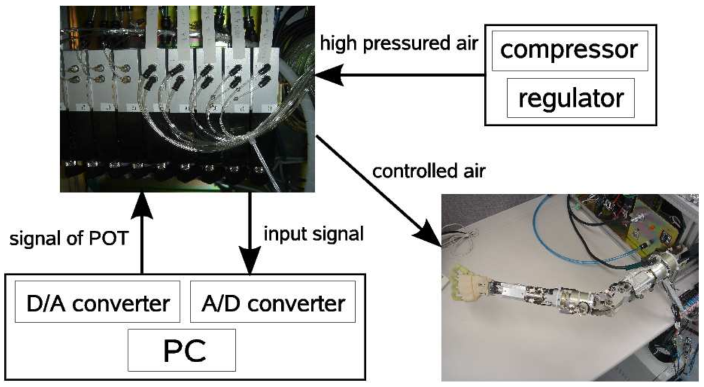

Fig. 3. The setting of the experiment for testing consolidation: An android arm is actuated by air pressure coming from a compressor.

After sequentially learning the three reference trajectories, the resulting neural network is tested for generalization and forgetting. For this, a validation set is created including the three reference trajectories and two new (not learned) trajectories. The tests were performed using the following different learning methods: Feedback-error Learning using static neural network of 20 neurons, NEAT and consolidation learning.

C. Results

In Figure 4 we see the the trajectories performed by the neural network learned through consolidation in comparison with the reference trajectories. It can be seen that the network learned to follow the trajectories without forgetting. The network was able to solve trajectories that have not been learned. Thus, it was able to generalize from given knowledge. Compared with the results of simple Feedback-error learning (see Figure 5), the trajectories also seem much smoother. Table I shows the mean squared of each method on the validation set. We can see from the table, that NEAT performed slightly better than Feedback-error learning. The reason for this might stem from the fact that NEAT employs populations of solutions. This provides the possiblity, that information about an old task is still contained in one of the population members.

Method	MSE
NEAT	0.055
Feedback-error Learning	0.056
Consolidation Learning	**0.044**

TABLE I

MEAN SQUARED ERROR MADE ON VALIDATION SET.

V. CONCLUSION

In this paper we presented a new computational framework for motor learning using growing neural networks. The framework is based on a process called consolidation which is inspired by neurobiological findings in humans. The framework is evaluated on an android robot and found to be successfull in avoiding catastrophic forgetting and achieving high generalization. However, the current implementation

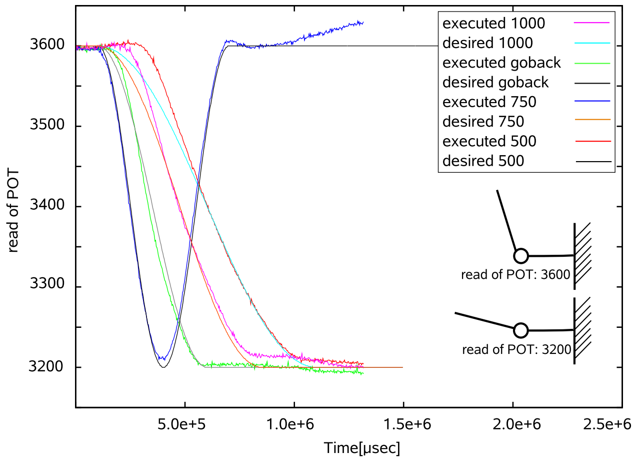

Fig. 4. Different desired and executed trajectories of the android arm after consolidation learning is finished. Time is given in microseconds.

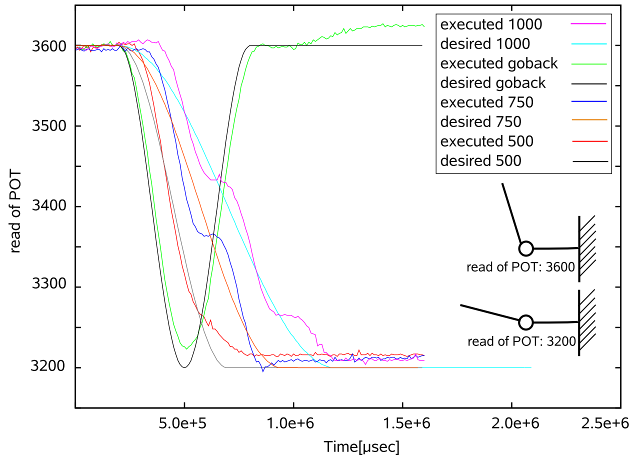

Fig. 5. Different desired and executed trajectories of the android arm after Feedback-error learning with static network of 20 hidden neurons.

assumes that input vectors used during learning are stored for future rehearsal. Depending on the learning task this can take large amounts of disk space. To improve this, we are currenlty investigating the reinstatement of rehearsal patterns through random input vectors.

REFERENCES

[1] S. Fahlman and C. Lebiere. The cascade-correlation learning architecture, 1990.
[2] M. Kawato. Feedback-error-learning neural network for supervised motor learning. *Advanced Neural Computers*, pages 365–472, 1990.
[3] R. Shadmehr and H. Holcomb. Neural correlates of motor memory consolidation. *Science*, pages 821–825, 1997.
[4] K. O. Stanley and R. Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10:99–127, 2002.
[5] S. Thrun. *Lifelong learning algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.
[6] S. Vijayakumar and S. Schaal. Fast and efficient incremental learning for high-dimensional movement systems. In *Proc. IEEE Int'l Conf. Robotics and Automation*, pages 1894–1899. IEEE Press, Piscataway, N.J., 2000.