

Essay

Introduction

Humans seem to have the best of two worlds; fast, instinctive reactions and deep logical thinking done over long time. We can catch a thrown ball with only a few hundred milliseconds (Kenward and Nilsson 2011) to react. We read each others faces and predict emotions, judge intent by the sound of a voice, we ride bikes and even drive cars on instinct, without much mental effort. At the same time, we have the ability sit down and do complex mathematics, construct intricate logical arguments or write complicated fictional stories with, requiring much mental effort, and in hindsight we fret over our social awkwardness and our shortcomings. In busy or crowded places, we can easily focus our attention on some arbitrary detail, like a familiar sound, a particular hair color or our own car, while discarding otherwise attention-grabbing impulses. This is the foundation of psychology theories within cognitive control, where humans have two modes of thought, System 1 and System 2, or fast and slow, one instinctive and one rational. Above, some examples of the two types of system are given. Instinctively, the slow rational part of us is viewed as the greater of these two systems, but 95 % of a normal day, a person relies solely on the fast system, while the slow system is running in the background, using little effort. The strength of the fast system comes from its ability to be trained by the slow system, it can learn more complex tasks as we grow and take over new tasks. For example, many experience "driving on autopilot", not remembering the drive to work or being completely lost in thought while driving. This is the fast system at work, where the drive took low mental effort and the act of driving the car was not in the forefront of the mind, but this must have been learned. To begin with, driving was a demanding task and required a great deal of attention. Here, the slow system was involved, constantly monitoring the fast system, correcting and coaching. Grandmasters in chess can rely on their fast system to judge the state of a chessboard almost instantly, but no one is born with this ability. Thus it is quite apparent that the fast system must be trainable by the slower more rational system. Kahneman offers a good overview of these two systems and their interaction between each other:

System 1 (fast system) runs automatically and System 2 is normally in a comfortable low-effort mode, in which only a fraction of its capacity is engaged. System 1 continuously generates suggestions for System 2 (slow system): impressions, intuitions, intentions and feelings. If endorsed by System 2, impressions and feelings turn into beliefs, and impulses turn into voluntary actions. When all goes smoothly, which is most of the time, System 2 adopts the suggestion of System 1 with little or no modification. (. . .) When System 1 runs into difficulty, it calls on System 2 to support more detailed and specific processing that may solve the problem of the moment. (. . .) System 2 is activated when an event is detected that violates the

model of the world System 1 maintains, or when System 2 detects an error about to be made

This division of labor is highly efficient, as the fast system is sufficient for the majority of tasks encountered, it performs well in familiar situations and provides good short-term predictions. However, the slow system is imperative, without it, humans would be inflexible, solely reactive, unable to learn new skills to adapt well to new environments and situations. Since the beginning of the 2010s machine learning and deep learning have continued to prove its unparalleled performance in a wide range of applications, but the algorithms developed are only able to retain high performance in narrow fields. A trained artificial neural network that is applied to a new task, is prone to "catastrophic forgetting", where the network loses its performance in previously mastered tasks in favor of new tasks. Therefore, for each application, a new algorithm must be trained with an accompanying dataset, which can be very data inefficient. The concept of meta-learning, "learning to learn", tries to avoid this by learning a general structure for all the tasks it will be exposed to. Another approach could be to retain some inherent information about each task the algorithm is exposed to, so when the task is mastered, the network's state can be saved to a knowledge database. When this task is later encountered, the state of the network can be used to solve the task quickly. Another approach, very recently proposed within the field of reinforcement learning, is to move away from model-free algorithms. In addition to learning the tasks, the network tries to learn a model about the world, then use this model as well to solve the task. When a new task is encountered, the network still retains information about the world and its inherent properties (such as gravity, inertia and such), given that the task is within the same domain. This does not address the problem of catastrophic forgetting, but alleviates the problem by exploiting the fact that the world doesn't change and is unnecessary to relearn. In this project, we attempt to combine the use of consolidating knowledge into predispositions and model-based algorithms, into a model that models the fast and slow systems described earlier.

Relevant methods

Machine learning

Heuristic algorithms use rules to solve a task. For example given an input, if this input is above a certain threshold, apply this operation and return the result. Machine learning algorithms differ from heuristic methods by not using the rules to solve a task, but attempting to learn to solve the task. The three types of machine learning differ in how they learn; in supervised learning the algorithm trains on a labeled dataset where the information we want is already known. In unsupervised learning the algorithm tries to find patterns and connections in large unlabeled datasets, modeling probability densities over the inputs. In reinforcement learning, we don't have a labeled ground truth, but we can infer how good a prediction is, usually by a reward function. This method has been shown to achieve great success on games and simulations with explicit reward

structures.

Previous work

Attempts at implementing the division of labor modeled have been done, the most relevant from Tzuu-Hseng S. Li et. al., where in 2016, a proof-of-concept was shown (Li et al. 2016). In this paper, they demonstrated a humanoid robot who's task was to throw a ball into a basket, using the model of two systems.

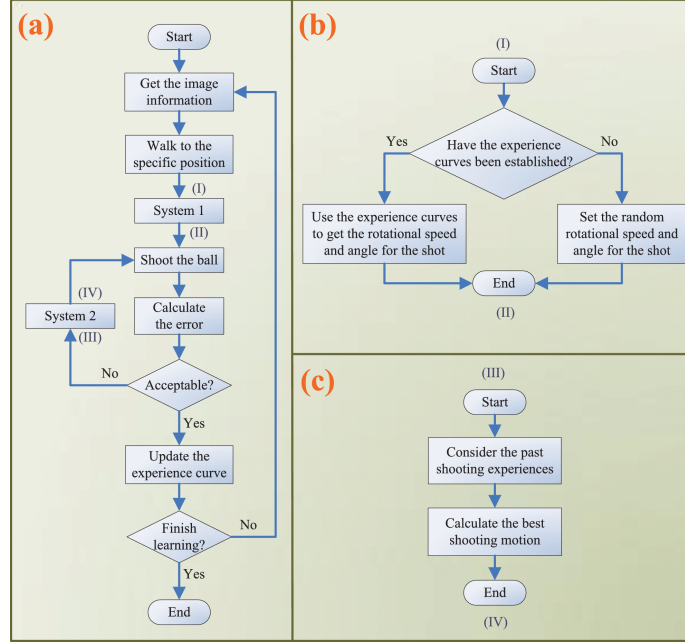


Figure 1: Describes the system proposed by (???)

System 1 was set to be two polynomials, one for rotational speed and one for the angles of the arm used to throw the ball. These polynomials were used to fit experience curves for each polynomial, based on distance and angle from the robot to the basket. The authors used 3 different methods for System 2, short-term memory (only the last shooting attempt is remembered), long-term memory (all attempts are remembered) and peek-end rule (the best, last attempt is remembered), another theory proposed by Kahneman. The remembered attempts are then used with a given correction rule to improve the coefficients of the two polynomials determining rotational speed and angle. This setup showed promising results, cutting down the learning time from 90 minutes to 20 minutes based on this setup, they especially note the usefulness of anchors and peak-end rule.

Later, in 2019 the authors published a new paper (Li et al. 2019), this time

proposing a Deep Belief network (DBN). The DBN is a simple neural network of stacked restricted Boltzmann machines, used to suggest angles for the entire arm and velocity of the joint between the overarm and the shoulder.

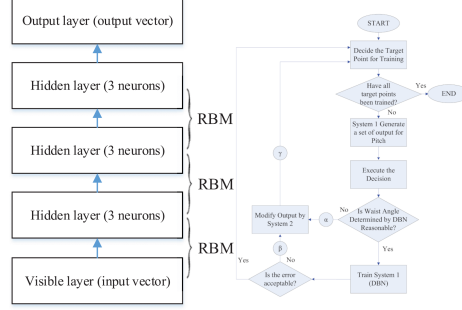


Figure 2: Overview of system proposed by (Li et al. 2019)

This replaces the two polynomials as System 1, while a population-based intelligent optimization (PSO) algorithm is used as System 2. PSO originates from swarm intelligence and uses particles to search for the best solutions in D-dimensional space. Here, it is modified to also use inertia weighting to for balancing both the local and global search space, to avoid getting stuck in local maxima. The two networks are then trained on alternating examples, until the DBN achieves a set accuracy level. This paper shows some faster convergence with DBNs compared to ANN (unspecified by the paper), while still achieving 100 % accuracy. The PSO was not very much used, it only intervened 5 times throughout the training process.

(Hafner et al. 2019) showed a novel method for mitigating the catastrophic forgetting problem. They moved away from model-free algorithms, where the goal is to use brute-force to find solutions, building a policy network that react to the in-puts with no inherent understanding of the world. Planning based algorithms, like PlaNet, learns stochastic and deterministic kinematics of the world, creating a model of the world and use this to plan ahead. This planning allows the algorithm to find better solutions with far less data.

Experiments done by (Hafner et al. 2019) find that it achieves similar performance as leading reinforcement learning algorithm with up to 100 times less episodes. The team trained the network on six different tasks within the same domain (OpenAI Gym), where the network had already learned rudimentary understanding of gravity and dynamics from the first task. This knowledge was then reused when learning the new tasks to achieve high performance with, on average, 50 times more efficiency than comparable algorithms. Additionally, (Hafner et al. 2019) used encoders to create compact latent states from images, instead of using the images directly. Due to its more compact states than images, the network is able plan/predict longer.

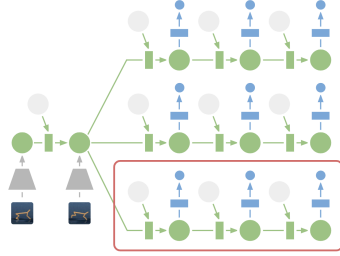


Figure 3: PlaNet planning method. Here three different predictions are presented, the red box around the one with highest reward

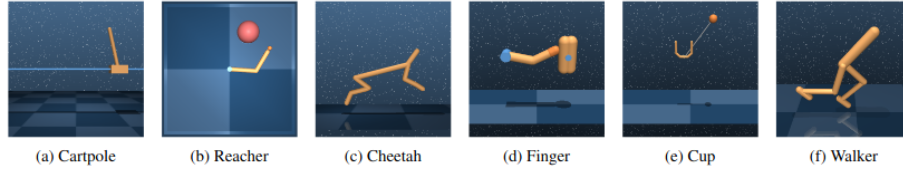


Figure 4: PlaNet tasks

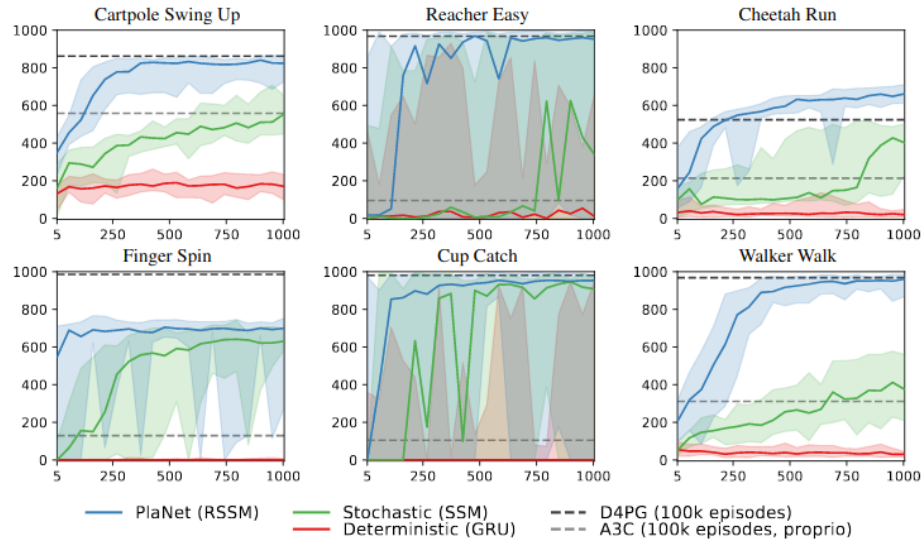


Figure 5: Results for each tasks

Peter Norstein, in his master thesis, created a novel system modeling the fast and slow system with reinforcement learning algorithm called TIDE. TIDE, short for Task Identification During Encounters, identifies a task based upon a task descriptor and tries to identify the task. If this task has been encountered before, the algorithm or neural network used to solve this task is used, but if its a new task thats yet to be solved, the system finds a solution to the task, associates it with the task identifier and stores for future use.

In this work, the fast system is represented by the pretrained or already found solution being applied to the task, while the slow system is represented as the process of finding a solution and associating it with the task identifier. This system tries to overcome the issue of catastrophic learning and solving more general problems.

Experiments show that TIDE perfoms well when the task descriptors have some structure, giving similar tasks similar task descriptors. When the descriptors are random, it perform worse than just learning the tasks when a new one is encountered (just going to the slow system immediately). This shows promise, as the experiments are done on the Multi-armed bandit problem, a simplistic environment. Here, a simple algorithm can achieve good results, but a complex system, like TIDE, should scale better with more complex tasks and/or more challenging environments.

The TIDE system of identifying tasks based upon the task descriptors where trained both using a supervised learning method and using a reinforcement learning method. In the supervised learning, TIDE is trained with a task descriptor and a ground truth task labels. Not allowing TIDE to have access to ground truth labels, but training with the rewards from the agent, changes this to a reinforcement learning problem.

Supervised learning significantly boost performance, but this method might not be possible to implement for example where there is no ground truth labels, reinforcement learning can provide comparable performance.

Original ideas

While some progress has been made in this field, the implementations have all been very restrained; some things are already known about the problem in different tasks in the same domain need to approximate two polynomials or we need a neural network to predict these 4 variables), they rely heavily on highly accurate sensors and due to this isn't directly transferable to other fields. If you would want to implement any of these described methods, it would require a lot of adjusting and customization to make the implementation fit the problem before us. We need to be aware of what this method seeks to improve, what is its main advantage? In this machine learning setting, its cutting heavily down on the compute cost of more advanced models, by learning or creating a simpler model that works for 90 %of the time and can be further improved by the more advanced model. This does not restrain us to use polynomials or

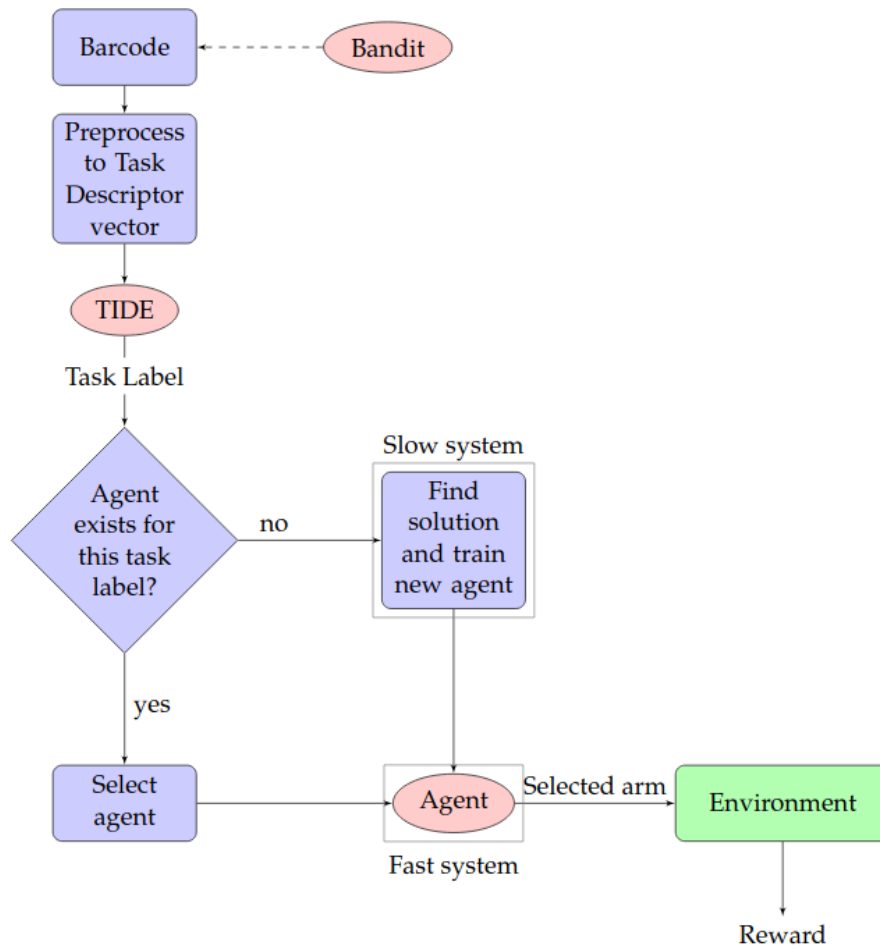


Figure 6: Overview over system proposed by Peter Norstein

using a set of known variables for the specific problem in question, but new advances within Reinforcement Learning and Deep Neural Networks allows us to create system that is agnostic to its environments and, more recently, to its own body/actor. If we were able to train an advanced Deep Neural Network, perhaps a Reinforcement Learning setup, and then simplify or approximate this system while still reaching 90 % accuracy of the full system and have the ability to be trained further, we could implement this as a dual system comprising of two neural net-works of different complexity. Preferably we would want to infer the complex model (System 2) after we infer the simpler model (System 1), but perhaps our main contribution can come in this area. If we can create a system that uses System 1’s confusion matrix, as well as some probability, to infer System 2 as well, we could even have a small additional neural network that learns when System 1’s suggestion should be verified by System 2, so that we can start to infer System 2 while we also infer System 1 and then wait for the System 2’s prediction. This could speed up cases when we need to infer System 2 as well. I would like to explore this area by using architectures that are known to be performant , but expensive and see if one can derive a simpler model that can act as System 1. If this is successful, then we can explore the logic and controls for switching and infering between these two systems. I propose to first explore the possibility to simplyfy? artificial networks and their structure drastically while maintaining an accuracy/precision well within reason, 90-70 %, but cut-ting down on infer times drastically. After this, I propose to use some kind of simulator or video game and create an artificial neural network, like an autopilot for a car simulator, to then see if this duality of models has an advantage and if we can create some smart system for predicting a hard case for System 1.

Hafner, Danijar, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. 2019. “Learning Latent Dynamics for Planning from Pixels.” In *International Conference on Machine Learning*, 2555–65.

Li, T. S., P. Kuo, C. Chang, H. Hsu, Y. Chen, and C. Chang. 2019. “Deep Belief Network–Based Learning Algorithm for Humanoid Robot in a Pitching Game.” *IEEE Access* 7: 165659–70.