Essay - Predictive and Alert systems

Håkon Tønnessen

Contents

Introduction

Humans seem to have the best of two worlds; fast, instinctive reactions and deep logical thinking done over a long time. We can catch a thrown ball with only a few hundred milliseconds (Reid et al. 2020) to react. We read each other's 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, particular hair colour or our car in a crowded parking lot, 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 experiences "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 at 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, if you drive a new car or in a new situation you use a lot more effort when driving. 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 (2012) 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 labour is highly efficient, as the fast system is sufficient for the majority of tasks encounter, 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 favour 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 networks 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 work, 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 attempts 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 labelled dataset where the information we want is already known. In unsupervised learning the algorithm tries to find patterns and connections in large unlabelled datasets, modelling probability densities over the inputs. In reinforcement learning, we don't have a labelled ground truth, but we can infer how good a prediction is, usually by a reward function. This have method has shown to achieve great success on games and simulations with explicit reward structures.

Previous work

Attempts at implementing the division of labour modelled have been done, the most relevant from Li et al. (2016) where a proof-of-concept was shown. 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.

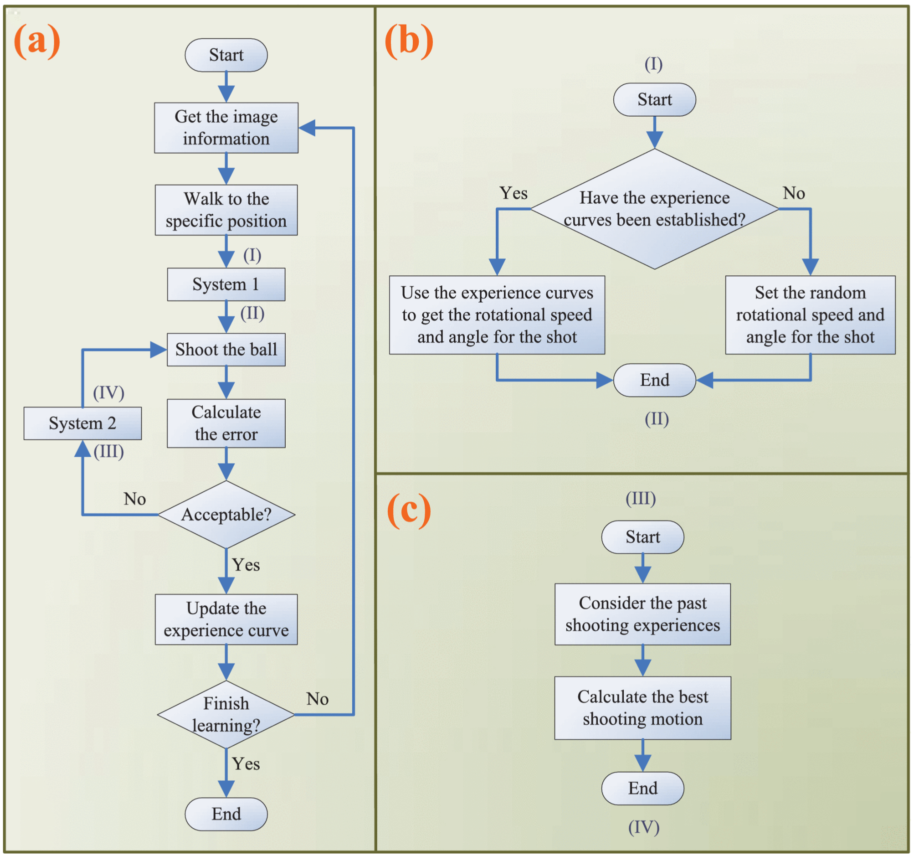

(a)
Start
Get the image information
Walk to the specific position
(I)
System 1
(II)
Shoot the ball
Calculate the error
System 2
(IV)
(III)
No
Acceptable?
Yes
Update the experience curve
Finish learning?
No
Yes
End
(b)
(I)
Start
Have the experience curves been established?
Yes
No
Use the experience curves to get the rotational speed and angle for the shot
Set the random rotational speed and angle for the shot
End
(II)
(c)
(III)
Start
Consider the past shooting experiences
Calculate the best shooting motion
End
(IV)

Figure 1: Describes the system proposed by (Li et al. 2016)

System 1 was set two 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 attempted is remembered), long-term memory (all attempts is being remembered) and peek-end rule (the best, last attempts is remembered), another theory proposed by Kahneman (2012). The remembered attempts are then used with a given correction rule to improve the coefficients of the two polynomials determining the 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.

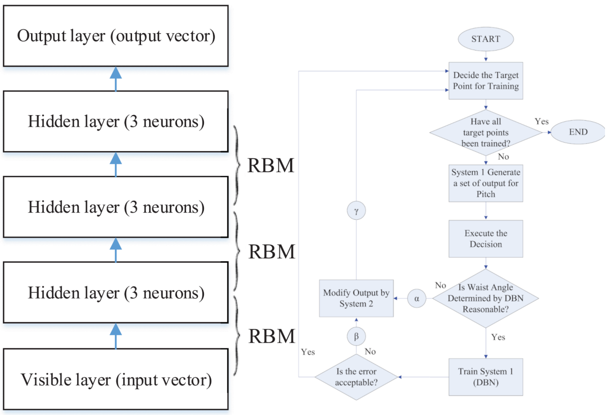

Output layer (output vector)
Hidden layer (3 neurons)
Hidden layer (3 neurons)
Hidden layer (3 neurons)
Visible layer (input vector)
RBM
RBM
RBM
START
Decide the Target Point for Training
Have all target points been trained?
Yes
END
No
System 1 Generate a set of output for Pitch
Execute the Decision
Is Waist Angle Determined by DBN Reasonable?
No
α
Modify Output by System 2
γ
Yes
Train System 1 (DBN)
Is the error acceptable?
Yes
No
β

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, called PlaNet, 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 reacts to the inputs 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.

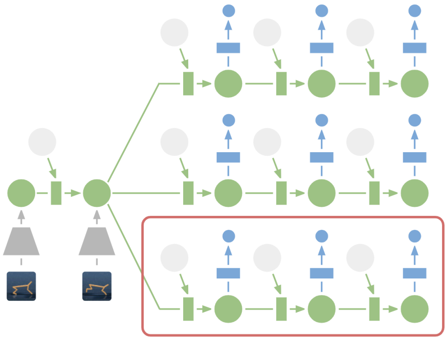
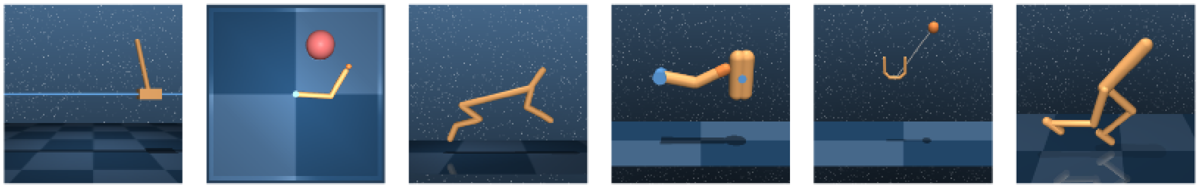


Figure 3: PlaNet planning method. Here three different predictions are presented, the red box around the one with highest reward

Experiments done by Hafner et al. (2019) find that it achieves similar performance as leading reinforcement learning algorithm with up to 100 times fewer 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 efficient than comparable algorithms. Additionally, Hafner et al. (2019) used an encoder network on the input images to create compact latent states, instead of using the images directly. Due to its more compact states than images, the network can have a higher data input and plan longer.



(a) Cartpole (b) Reacher (c) Cheetah (d) Finger (e) Cup (f) Walker

Figure 4: PlaNet tasks

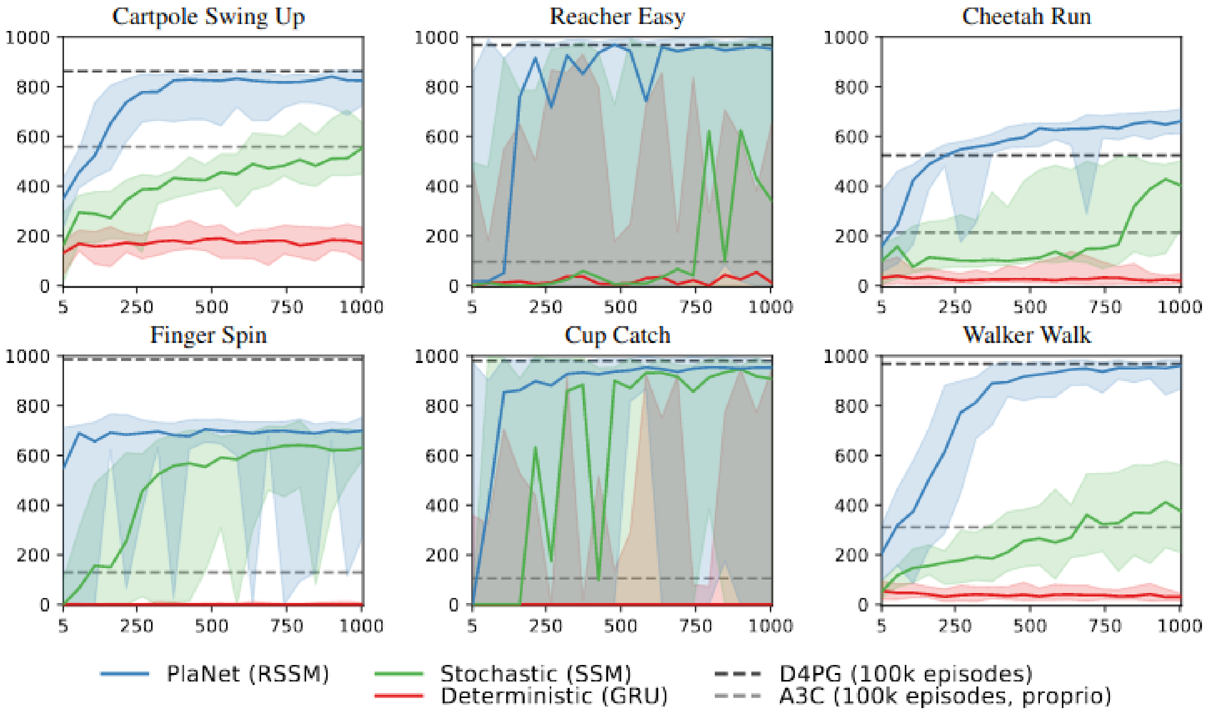


Figure 5: Results for each tasks

Peter Norstein, in his master thesis, created a novel system modelling the fast and slow system with reinforcement learning algorithm called TIDE. TIDE, short for Task Identification During Encounters, uses task descriptors to identify tasks. If this task has been encountered before, the algorithm or neural network used to solve this task is used, but if it's a new task that's 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 performs well when the task descriptors have some structure, giving similar tasks similar task descriptors. When the descriptors are random, it performs 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 was trained both using a supervised learning method and using a reinforcement learning method. In supervised learning, TIDE is trained with a task descriptor and 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.

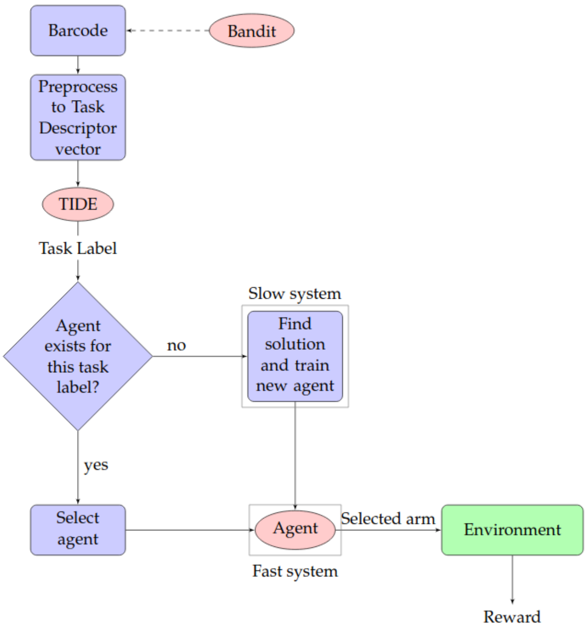

Barcode
Bandit
Preprocess to Task Descriptor vector
TIDE
Task Label
Agent exists for this task label?
no
Slow system
Find solution and train new agent
yes
Select agent
Agent
Selected arm
Environment
Fast system
Reward

Figure 6: Overview over system proposed by Peter Norstein

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

The issue of this method is to create task descriptors that have a structure. We want task descriptors where similar tasks result in similar descriptors but deriving these descriptors, agnostic of environment and task type, is a challenge. This is, however, the biggest hurdle to make this a usable system.

Original ideas

In this thesis, I would like to address the transfer learning problem by utilizing the best from PlaNet and Peter Norsteins master thesis. By combining PlaNets model-based approach to reinforcement learning and TIDEs method of classifying tasks, into a system that is task and environment agnostic, would be the long term goal.

The proposed system could be described as such:

> A task has a structured task descriptor, either by deriving this from some inherent information from the task or these are designed by hand, and TIDE classifies this to the appropriate agent if a pretrained one exists. If this isn't found or TIDE's confidence is too low, a new agent is trained by taking the model from the most similar task found, as described by PlaNet (Hafner et al. 2019). When this reaches good enough performance to solve the task, the agent is stored in the database.

Preferably, this should be done in a 3D simulated environment, since TIDE should have better success with more complex tasks and this is a point for further research for PlaNet. The initial goal is to determine if such a system is feasible and produces good results by just using predetermined task descriptors for each task, then move on to deriving these task descriptor based upon the inherent information of the task. To do this, we would require some information of the task, like its goal, the environment it's set in and such, but these should be able to either be derived from the simulation environment or by some simple human-readable information.

As an example, we could give the desired end state as a pose, with images from the environment and a short description of the task:

> Drive this car home. Park it inside the garage and close the garage door.

Taking more inspiration from PlaNet (Hafner et al. 2019), we could perhaps use an encoder-decoder network to give representations of all this information and learn the appropriate agent. However, this could potentially be a big challenge and is perhaps outside the scope of this thesis.

References

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.

Kahneman, Daniel. 2012. *Thinking, Fast and Slow*. London: Penguin Books.

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.

Li, T. S., P. Kuo, Y. Ho, C. Liu, T. Yu, Y. Ye, C. Chang, et al. 2016. "Robots That Think Fast and Slow: An Example of Throwing the Ball into the Basket." *IEEE Access* 4: 5052–64.

Reid, Brian, Kelley Schreiber, Jason Shawhan, Ethan Stewart, Reuben Burch, and Will Reimann. 2020. "Reaction Time Assessment for Coaching Defensive Players in Ncaa Division 1 American Football: A Comprehensive Literature Review." *International Journal of Industrial Ergonomics* 77: 102942.