ID: 2018A8PS0844H
Name: Maddha Abhinava
CS F407 Assignment-I

Title of Study: Reinforcement Learning and Neural Networks for PID Control

Reinforcement Learning and Neural Networks for PID Control

Abstract: In recent times in the industry, the process control techniques have made great progress, which is an essential part of the instrumentation field of industries. Among various control methods Proportional Integral Derivative (PID) controller, a common feedback component is very widely used in industrial control. It is a very practical control technique due to its versatility, high reliability, standard structure, ease of operation and robustness in terms of performance under an extensive range of operational conditions. Various controller design methods like adaptive control, neural control, fuzzy control etc., have been the topics of research in this field of study. Artificial Intelligence has become revolutionizing, especially to the industries. Research in AI has allowed us compete with human intelligence on a whole new level. Research in Machine Learning, Deep Learning and Reinforcement Learning is being fueled by the increase in availability of data which can be used to train agents to reach the level of human intelligence. In this report we go from exploring AI techniques broadly, to focusing on applying Reinforcement Learning algorithms and Deep Neural Networks to a few fields like games(since RL and games have a long standing, mutually beneficial history) and then PID Control. Defining the Reward function is the main and most crucial task while designing the model that is adaptive to continuous action spaces which is the apparent scenario in most practical control environments. This report concludes with the observations from the research papers considered and a proposal of applying the concepts of Temporal Difference Learning techniques along with Neural Networks to supervise PID control in industrial plants, in order to broaden the scope of application of PID control and apply it to any system, to handle non-linear systems and continuous action spaces.

Table of Contents

Introduction

Artificial Intelligence is the field that aims to understand and construct intelligent entities. It is a universal field encompassing subfields like Machine Learning, and is applied to a range of other fields. The Turing Test proposed by Alan Turing (1950) is used to testify whether a computer system has a human level performance in cognitive tasks, enough to be able to trick an interrogator into believing it's a human, by looking for the following capabilities in the system:

a. Linguistic understanding: The ability to understand, interpret and reply using natural languages by translating it from vocal form to written, processing it and translating to various other languages. This includes: Speech Understanding, Information retrieval, Natural Language Processing, Answering questions and Language Translation domains of AI.
b. Adaptivity: The ability to learn from previous experiences and apply that knowledge to new situations- Cybernetics, Concept Formation domains.
c. Problem solving: The ability to model the problem statement, and find the solution while knowing and extracting additional information necessary for the execution, and involves: Interpretation, Interactive Problem Solving, Automated Program writing and Heuristic search.
d. Perception: The ability to perceive the environment from the sensors and relating it to an internal model of the world, thus defining a set of assosciations between the external objects in the environment- Pattern recognition, Computer Vision, Scene analysis.
e. Modeling: The ability to predict the relationship between external objects by modeling an internal representation and transformation rules through- Representation and Problem-solving Systems, Hobot World Modeling.
f. Robotics: The physical execution of the above areas on a structure to be able to move the objects around- Exploration, Transportation (Navigation), Industrial Automation, Security, Agriculture, Mining, Construction, Military.

Machine learning (ML) is a branch of AI, and defined as the study of algorithms that allow computer programs to automatically learn through experience, without being explicitly programmed. ML includes three types of learning-
Supervised learning, Unsupervised learning and Reinforcement learning.

In the domain of artificial intelligence, Reinforcement Learning (RL) is one of the common learning forms especially used in games and robotics, following the concept of hit and miss mechanism, which is very close to how humans learn. It is a form of goal-oriented learning from the interaction between the agent and its environment wherein, an agent learns to act in the environment to achieve specific goals and maximize a certain reward function. Although both Supervised learning as well as Reinforcement learning map input and output in a similar way, SL uses loopback of a correct set of actions, whereas reinforcement learning uses rewards (positive

or negative) as signs for optimal behavior and penalty for the opposite. Reinforcement Learning has different goals compared to Unsupervised Learning. The goal in unsupervised learning is to cluster data, whereas in reinforcement learning it is to find the optimum action model/sequence in order to maximize the cumulative reward received by the agent.

The different methods of Reinforcement Learning are-
Value-based learning, Policy-based learning and Model-based learning.
The learning models of RL are-

1. Markov Decision Process (mathematical framework to describe an environment)
2. Q-learning (model-free approach to build an agent).

The different algorithms used in RL are-

1. Q-learning- model-free, off policy method to estimate q values based on another policy
2. SARSA- model-free, on-policy method that uses its own policy to estimate values
3. Deep Q-learning- use Neural Networks to estimate q values
4. DDPG (Deep Deterministic Policy Gradient)- model-free, off-policy, actor-critic algorithm that learns policies in high dimensional, continuous action spaces).

Basic terms used in Reinforcement Learning**:**

1. Markov Decision Process:
   A Markov decision process is a discrete-time, stochastic control process that provides a mathematical basis to model the decision making in situations where outcomes are partially arbitrary and partially under the control of a decision-maker.
   For RL an MDP (Markov Decision Process) is taken as a tuple (S,A,R,T) where:
   • S :the set of all states $s_t \in S$
   s_t is current state at time step t
   • A :the set of all actions $a_t \in A$
   a_t is action taken at the current time step t
   • R : mapping $S \times A \times S \rightarrow IR$:the reward function that outputs reward R as a function of s_t, a_t, s_{t+1}.
   • T : mapping $S \times A \times S \rightarrow [0,1]$:the transition function, T(s,a,s′) which gives the probability of reaching s_{t+1} after performing a_t in s_t.

0. Discounted Reward:
   'G_t' discounted reward, is the total reward(discounted using $\gamma \in [0,1]$) that an agent earns by following a whole trajectory starting from t.

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots \quad = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}.$$

0. Policy:
   A policy π is a behavior function that maps states to actions, π(s)=a
0. Value Function:
   The value function represents the quality of a state, telling how good/useful it is for an agent to be in, while following policy π. It is equal to expected total reward for an agent starting from state 's' $V_\pi(s) = E[R_t \mid s_t = s]$
0. Q-Value:

Q Value is a measure of the expected reward overall, assuming the Agent is in state 's' and performs action 'a', following some policy π $Q_\pi(s,a) = E[\, R_t \mid s_t = s, a_t = a]$

0. Optimum Value function:

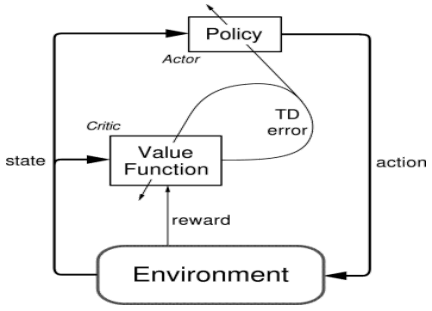
The optimum value function $V^*(s)$ gives the largest expected return achievable by any policy π for each state. $v_*(s) = \max_\pi v_\pi(s)$

The value function corresponding to the optimal policy will hold the following property:

$V^*(s) = \max_a \sum_{s'} T(s,a,s')\Big(R(s,a,s') + \gamma V^*(s')\Big)$ -also called the Bellman optimality for $V^*(s)$.

Actor-critic learning is a common RL learning method used to find the optimal action and expected value simultaneously. Temporal difference is a model free RL method that uses the errors in predictions over successive time steps to drive the learning process of the agent. A2C methods are Temporal Difference methods with an isolated memory construction to explicitly represent/store the policy π irrespective of the value function V. The policy structure is called actor, because it is used to select actions, and the estimated value function is called critic, because it evaluates the actions taken by the agent/actor based on the q value and gives a feedback to the actor.

Actor-Critic is a subtype of Policy Gradient algorithms, which aims to directly learn an optimal policy (a way to act given a state) for an environment, instead of learning intermediate values (like Q-learning). In each time step, the policy takes an action and the critic will adjust the policy according to the state value. Critic will be implemented as one of the value functions (Q value, or cumulative sum of discounted rewards, or Advantage), which will be learnt online using Temporal Difference learning. This value signal in the form of TD error is taken from the critic to operate the agent's learning based on the policy, as suggested by figure below.



After the selection of every action, the critic evaluates the new state to determine whether there has been an improvement or otherwise which is the Temporal Difference(TD) error:

$$V_{t+1}(s_t) = V_t(s_t) + \alpha_t \delta_t \,, \qquad (2)$$

where δ_t is the TD-error, defined as $r_t + \gamma V_t(s_{t+1}) - V_t(s_t)$

where V is the current value function implemented by the critic. Th TD error is used to evaluate the action just selected(a_t) taken in state s_t. If the TD error is positive, the inclination to select

a_t should be reinforced for the future, whereas if the TD error is negative, the inclination should be faded.

Neural networks are artificial systems inspired by biological neurons. They learn to perform tasks from datasets and examples fed to them, without any task-specific rules or prior process modelling information. Basic components of a typical neural network are input layer, hidden layers or computation layers, output layer, perceptrons in every layer, weights assosciated with every perceptron and an activation function. The learning rule forms the basis for weight updates in the network. The different types of NNs are:

- Multilayer perceptron- has three or more layers, applied to nonlinear datasets making use of hidden layers.
- Feed-forward Neural Network- the simplest Artificial Neural Network
- Radial basis function Neural Network- with two layers.
- Convolutional neural network/CNN- uses a variation of multilayer perceptrons, have distributed weights and have invariance characteristics.
- Recurrent neural network/RNN- makes connections between the neurons in a directed cycle/graph enabling it to have temporal behaviour dynamically.
- Long short-term memory neural network- uses RNN architecture without the activation function.
- Sequence to sequence modules- use two RNNs
- Shallow neural networks- produce a vector space from a certain amount of text.

Proportional Integral Derivative (PID) control adjusts a control output based on the error in a measured process variable (PV) as compared to a set point (SP) and a further the value of the controller output u(t) is transferred as the system input forming a closed feedback loop. The Proportional, Integral and Derivative modes are arranged by controller manufacturers into three different controller algorithms or controller structures, Interactive, Noninteractive, and Parallel algorithms. The first few PID controller tuning methods involved observing a loop with proportional action on the edge of steadiness, then decreasing K_p value to get a stable state and calculating integral and derivative terms from the loop oscillation. There are many methods to tune a PID controller. In the Manual tuning method, we adjust the gains by trial and error using the Pole placement (Root Locus) or Loop shaping (Bode plot) methods to arrive at the most convenient set of gains. Such trial and error can be inefficient as they are iterative and time consuming and can damage the hardware. The Ziegler Nichols Open Loop method can be applied only when the unit step response of the plant is an S-shaped curve. Also, it doesn't hold good for I, D and PD controllers. Although the Cohen Coon's method has more convenient tuning rules that work well when the dead time is less than twice the time constant, it is only good to apply for first order systems. The Ziegler Nichols Closed Loop method can be applied only when the tuning processes cannot run in an open loop environment. In this method, the critical gain value ($K_{critical}$ or K_u) and critical oscillation period($T_{critical}$ or T_u) are used to find the looping constants of the controller. This is what we employ into our code to get the minimum and maximum values of the parameters to denormalize them. It tunes the loop for quarter-amplitude-damping response, which overshoots and oscillates quite a bit and can lead to

loop instability. Auto tuning methods using softwares like Matlab and Simulink can be a lot more convenient than the ones stated before in terms of accuracy, in-built PID tuner commands, batch modes and more.

Approaches to tuning the parameters of a PID controller can be divided into two essential classes: standard PID tuning methods and adaptive PID tuning methods. The traditional tuning process approach is complicated, detailed and it is tough to dodge large overshoot and oscillation, consequently producing the optimal parameters is quite challenging. Due to this problem intelligent PID tuning methods are attracting much attention. Aimed at the deficit of self-tuning PID parameters, Machine Learning algorithms are being proposed, specifically Reinforcement Learning methods. RL is most widely used in designing AI for agents that play computer games. It is also being applied extensively in robotics and industrial automation as the world progresses towards Industry 5.0

In this report we explore how PID tuning methods , Reinforcement Learning and Neural Networks evolved.

Literature Survey

This section presents an overview of research activities from the introduction to the field of study until the recent research developments in deep reinforcement learning.

Atkeson et.al (1997) present a comparison on a basic level in the Reinforcement Learning algorithms in their paper. They distinguish between direct and model-based RL. The training data requirements are tested for continuous action tasks taking the example of a pendulum swing up. They describe the implementation of both algorithms and find that the model based algorithm is more data-efficient, finds better policies and handles the change in goals better. It is pointed out that model based algorithms are favourable for simple dynamics, both approaches greedily exploit the current policy as opposed to exploring new ones but model based algorithms learn with two kinds of exploration, one to improve the model and the other to find better policies. It is stated that Q-learning uses mental simulation and a model to plan good policies (Sutton, 1990). They conclude with their findings that model-based algorithms are better than direct reinforcement learning algorithms.

Savinay et.al (2017) present a comparison of algorithms like policy gradient, actor-critic and temporal difference with value function approximation when applied to cart-pole inverted pendulum dynamical system. While the cart-pole modelling details are not known to the RL algorithms, the agents learn through exploration and exploitation to find the best policy. Modelfree learning algorithms are characterized as: single-step backup(temporal difference learning), infinite-step(full-length backup, also popularly known as Monte Carlo learning), and n-step backup with TD(λ) that makes use of the discount factor gamma while taking backups. This is constructed based on the backup span or the time step count of the agent-environment interaction before updating the q value estimate. It is concluded that although the actor-critic policy-gradient (A2C-PG) method converges to the optimum point sooner and provides better steadiness in discrete state-space, value-function approximation with TD learning performs best overall, among the three RL algorithms discussed in the paper.

Brendan et.al (2016) have discussed in their work, a way to combine Policy gradient and Q learning and termed the technique PGQL by connecting the q values and fixed points of the regularized policy gradient. The motive was to handle the off-policy data using the Q-learning part which the vanilla online variants of policy gradient cannot make use of. In value based reinforcement learning we use a function approximator strategy to approximate the quality values and then update the parameters so that the q values tend to the fixed point of the Bellman equation whereas in policy gradient, we parameterize the policy directly and attempt to improve it using gradient ascent on the performance J: $\nabla_\theta J(\pi) = \mathop{\mathbf{E}}_{s,a} Q^\pi(s,a) \nabla_\theta \log \pi(s,a),$ -grad J in terms of parameters of the policy. Following this explanation, Actor-critic and Action-value methods are discussed and it is shown that if we modify the action-value policy a bit, we get the actor-critic

policy gradient method. Following the same thought, the Bellman residual is explicitly attempted to be reduced by adding an auxillary update off-policy to arrive at a hybrid algorithm of PGQL. They test the PGQL on an Atari benchmark and find that it performs better compared to Q learning and PG methods by themselves, in terms of data and stability.

Silver et.al(2014) explain the concept of deterministic policy gradient and show that it is more efficient than the stochastic policy gradient used extensively until then. An off-policy actor-critic algorithm is implemeneted. The algorithm learns the set goal deterministic policy by exploring done by following an exploratory behaviorial policy. This is done to make sure the algorithm does not just exploit the current policy but also explores adequately. Stochastic policy gradient, stochastic actor-critic, and off-policy actor-critic algorithms are clarified as background and then transitioned to presenting the action-value gradients for deterministic case, and the deterministic policy gradient theorem is explained. They demonstrate the dead-ends encountered in stochastic policy gradient. A mathematical explanation is given for deterministic actor-critic algorithms on-policy and off-policy. They conduct an experiment on continuous action variants of standard benchmarks for reinforcement learning and compare SAC and DAC and see that the cost for DAC is significantly less. They conclude that the DAC significantly outdoes its stochastic equal by several orders of magnitude especially in the case of continuous action spaces.

Steven et.al(1996) introduce in their paper, three new algorithms for temporal difference learning Normalised (NTD), Least squares(LS TD) and Recursive least squares(RLS TD) need more computation than Sutton's TD(λ)(Sutton, 1988) per time step. They use more information, as should be done ideally, from the training knowledge of the agent and show a significant improvement in the learning rate as quantified by the TD variance error in the case of RLS. They show that in these new methods, since there is no involvement of a hyperparameter like alpha or learning rate, there is no chance of the human error of choosing a wrong parameter value to occur. They compare the convergence rates of the previous algorithms with those of the new ones to find out that the learning rules TD, NTD, LS TD and RLS TD have their experimental convergence rates depending on the TD variance error and thus denotes the amount of noise that cannot be removed no matter what learning rule is used. They conclude with a summary of their findings from the comparison of the four TD algorithms and state how the LS and RLS perform better than the rest.

Hasselt et.al(2007) introduce in their work, the much needed algorithm for continuous action spaces which is the most prominent case in the industrial scenarios, the Continuous Actor Critic Learning Automaton abbreviated to, CACLA, a straightforward and easy implementation unlike the previous works. They compare it to the previous algorithms that handled continous action spaces and find that it performs with a commendable improvement specifically when paired with the Gaussian exploration strategy. They demonstrate the characteristics of this model-free algorithm experimentally and show that it finds real valued continuous solutions, performs a

good simplification of features and selects actions fast. A model-free approach for continuous action spaces would be able to determine the best action just after a few observations using Function Approximation. A model-based approach would, on the other hand, require a lot of search only to design a basic model in order to choose the best action sequence. They conclude from the experiments on the tracking agent and cart-pole that this new class of CACLA algorithms(with Gaussian exploration and Function Approximation) within the RL framework has relatively simpler implementation and fewer computational requirements as compared to the previous two algorithms.

Specht(1991) describes in his paper a memory based neural network that estimates continuous variables and converges them to the regression surface. General Regression, Normalization of inputs, selection of hyperparameters, and clustering are discussed as background for Neural Network basics and compared to other regression methods like conventional non linear regression and point out that the main advantages of the GRNN over the others is that as the number of samples increases, the learning and convergence becomes faster and better, especially with sparse data in real-time environments.This General Regression Neural Network discussed is a parallel structure and can be applied to learn industrial plant dynamics in order to control it. They demonstrate an implementation of the GRNN when applied to an adaptive controller and show how the output of the plant has smooth transitions when the set point is changed. The paper is concluded with a summary of the advanteges of the GRNN over the others while pointing out that GRNNs without clustering have a disadvantage of requiring more computation for new data points which can simply be overcome by using the clustering versions.

Hansen et.al(1990) propose in their paper various ways to optimize the performance of a neural network. Cross-validation is a technique presented that helps decide the parameterization for a data set. The neural network set used is referred to an ensemble and conclude that as compared to a single network by itsef, an ensemble is a lot less imperfect. Different search methods are discussed like steepest descent and conventional gradient descent to coverge to the local minima. Cross validation is applied after the algorithm learns the training data set and evaluates the objective function and optimizes the architecture of the neural network. An analysis of the a feedforward neural network is done to see how changing the number of perceptrons and layers effect the architecture. In the experiments to test the ensembe models with a noisy rule, their quantitative predictions had a good performance. They conclude that the search for satisfactory weights happens where there are many “traps” corresponding to various ways of “generalizing” the rule hidden in the training set.

Leuenberger(2017) in his paper describes the application of the CACLA to a two dimensional aerial combat simulation environment, whose Actor and Critic have multi layer perceptrons. The “Ornstein-Uhlenbeck” process has been explained, and stated that it improves the exploration of the action space, and a new idea is introduced: the Monte Carlo version of CACLA, that

memorizes all the states of the present episode. A corrected version of CACLA is put forth as another idea, one that has a corrected learning rule of the Actor module. The correction is based on a third multi layer perceptron (MLP) with the same count of hidden perceptrons and inputs as the other layers, that guesstimates the absolute TD-error. It was observed that compared to the Monte Carlo and the original version of CACLA algorithms, the corrected CACLA achieves better as reflected by the TD error vs Time graphs. The corrected CACLA is constructed to be advantageous until the agents learn to improve their flight safety. The paper is concluded showing the results that the Corrected version performs better than the original version which is better than the Monte Carlo method.

Baker et.al(2016) present the design of neural networks with reinforcement learning in their paper. They rely on Q Learning to design a CNN that requires human computation and expertise. A MetaQNN is introduced that uses reinforcement learning to create better performing convolutional neural networks mechanically. The training of the learning agent is done using epsilon greedy strategy and experience replay to choose the layers of the CNN. The agent iterates through a large finite set of possible architectures to converge to the architecture with the best performance for the specific learning task. It is shown that the MetaQNN outperforms the existing meta modeling approaches for network designing. For each layer type namely convolution, pooling, fully connected, and termination state, the layer parameters are listed along with the allowed ranges of values to restrict the action space. The experimentation is carried out on datasets like MNIST, CIFAR-10 among a few to notice that as epsilon decreases the accuracy or learning capability of the agent improves to select CNN architectures. They conclude that when combined with hyperparameter optimization, to automate the network designing better and using function approximation as well, larger state-action spaces can be dealt with.

Lee et.al(1989) propose in their work a way to design AI base controllers using reinforcement learning and approximate reasoning. Linguistic control rules from human controllers are taken along with a reinforcement learning form related to TD learning. Using the Temporal difference laerning idea, the system has the capacity to learn from past experiences to predict its behaviour in the future. It is applied to the cart-pole balancing problem, whose dynamics are not completely known hence it is treated as a black box. The hybrid model of a rule-based controller that learns from past experiences using temporal difference learning is explained as proposed and then approximate reasining is introduced into the model. The simulation results are studied with respect to adaptability, learning, and training and conclude that the performance and steady state behavior are better than the previous works.

Peters et.al(2006) present the implementation of Policy gradient methods for high-dimensional robots. The estimation methods considered are finite difference method and likelihood ratio method. Vanilla policy gradient approaches are discussed to see the advancements in the policy gradient theorem, natural actor critic approaches and optimal baselines. This method is applied to

simulated plants and robots to observe motor skill planning, learning and control law optimization. The experiments are done on a cart-pole problem modelled as a linear stochastic system, non linear dynamic systems, a seven degree of freedom robot with the SARCOS master arm or T-ball with different combinations of policy gradient approaches and estimation methods. The graphs are presented interpreting the performance per episode and it is concluded that the time variant episodic natural actor critic is most preferred only when the policy can be differentiated with respect to its parameters.

Florensa et.al(2017) propose in their work a general framework that in a pre-training environment, learns skills and uses them to learn faster in the tasks. The learning in the pre-training is guided by a single proxy reward whose design does not require much knowledge about the task. The agent is trained with a high level policy to improve exploration and help tackle sparse rewards. The Stochastic neural network is used to pre-train the agent with the large set of skills, along with a theoretic regularizer to prevent the multi modal policies to collapse in to a single mode when the stochastic neural network is optimized. So, an additional reward bonus is added, which is proportional to the mutual information (MI) between the latent variable and the present state. Policy optimization is done using the Trust Region Policy Optimization(TRPO) algorithm with gaussian exploration strategy. They conclude that their framework successfully combines two parts the unsupervised procedure to learn a large set of skills with proxy rewards and the hierarchical structure that condenses the skills and allows to reuse them in future tasks.

Onat et.al(1998) propose a learning scheme in reinforcement laerning(Q learning) that employs recurrent neural networks to overcome the difficulty of requiring past experiences when dealing with partially observable systems. Sensory input is enhanced to refrain from perceptual aliasing, dynamic behavior representing the environment's state from past experiences is stored in the memory using the recurrent neural networks and a reactive policy is used to use the current sensory information in the best possible way. Two architectures of RNNs are considered: X-model and Elmans. The learning algorithms for RNNs discussed are Error Backpropagation Algorithm(BP), Back Propogation Through Time(BPTT) and Real Time Recurrent Learning(RTLL). A comparison of the network architectures and learning algorithms is made and it is concluded from the convergence profiles that Elman with BP was slowest and X-model with BP was fastest in terms of learning rate but X-model with RTLL would perform better in difficult environments.

Psaltis et.al(1988) present a modified BP(error backpropogation) system that can be applied to controllers. The would be the input to the feedback controller module and hence the training of the network will gradually shift from feedback to feedforward mode of the controller as the error signal becomes minor. Initially, during the training, features of the plant module that are not included in the training dataset and learnt by the controller algorithm to remove any uncertainity,

i.e., the controller is modified to compensate for the lack of initial characteristics.The error backpropagation algorithm is modified to expand its efficacy to cases where the error used to train the network is not that error which is measured at the output of the network. The experiment is conducted on a simple plant that translates coordinates from polar to cartesian form. Finally, generalized training in combination with specialized training is proposed to avoid their potential disadvantages.

Gerald(1995) presented an application of Temporal Difference learning in the game learning of TD-Gammon to play backgammon. TD-Gammon architecture includes a standard neural network, a multi layer perceptron with backpropogation. The learning in the neural network involves the change of weights of the neurons to converge the non-linear function to the target. At every time step, to change the weights, Sutton's TD(lambda) algorithm is applied where alpha(leraning rate) is a hyperparameter and lambda is a heuristic parameter. The results prove that the later versions of TD-gammon surpasses the learning done by neurogammon. TD learning when applied to deterministic games needs an external noise source to produce the variation and exploration as obtained from the random dice rolls in backgammon. It is concluded that TD-Gammon's self-teaching was able to perform well when competed against by world champions of backgammon, although there have been a few improvements suggested by the authors.

Silver et.al(2013) presented in their paper, the first deep reinforcement laerning model, a convolutional neural network, trained with a Q learning variant and stochastic gradient updates. It is an extension to their neural fitted Q-learning (NFQ) algorithm published prior to this. The CNN has input as RGB images and outputs a value function that estimates future rewards. Experiece replay is used to store the agent's knowledge at every time step and condensed into a 'replay memory', to have smooth laerning and reduce the oscillations. The algorithm is tested on seven Atari 2600 games using the same neural architecture, RL algorithm and hyperparameters and without any prior game-specific knowledge. When compared to other laerning methods like deep SARSA, it is observed that DQN with epsilon greedy strategy outperforms the others. They conclude that without any modification made to the algorithm's strategy, it performs well on six of the seven games tested and has a close to human level performance in the case of BeamRider.

Zhao et.al(2016) propose a deep reinforcement learning method called deep SARSA that uses a deep convolutional neural network(DCNN) to evaluate the state-action pair values and implement SARSA, an on-policy method to update the values, along with experience replay to solve compllex control problems. The CNN has three convolution layers and two fully connected layers. This method is applied to an Arcade Learning environment for video games. Two popular video games from Atari 2600: breakout and seaquest, are simulated with the algorithm proposed. SARSA stands for state-action-reward-state-action, meaning from state_1 the agent performs action_1 and gets a reward based on the reward function for state_1 and action_1 and then

reaches the next state state_2 and then performs action_2 and the cycle goes on. SARSA is an MDP policy(i.e., an on-policy technique of reinforcement learning) with a slightly different update equation than that of the off-policy Q-learning technique. The simulation results demonstrate faster convergence and better performance of the deep SARSA algorithm as compared to deep Q learning. SARSA learning has some advantages when it is applied the problem is about decision making.

Howell et.al(2007) apply the Continuous Action Reinforcement Learning Automata (CARLA) algorithm to make a pid controller a self-tuning one. Initially, the PID parameters are set to the standard Ziegler-Nichols values. Then the parameters are tuned on-line using the CARLA algorithm. It is an extension to the older discrete learning automata algorithm(Najim et.al, 1996). A the learning proceeds, the action values end up with a Gaussian distribution centered around the mean as most successful action. The CARLA trained pid controller is applied to a Ford Zetec engine and the on-line tuning using CARLA showed acceptable results when compared to offline tuning following the standard Ziegler Nichols rules, as the set point was varied. It is concluded that this adaptive PID controller has a commendable performance even though it does not require any process modelling data about the plant considered, and also adapts to the non-linearity in the system.

Shang et.al(2013) present in their work, a way to optimize the PID gain parameters using Function optimization by Reinforcement Learning(FORL). For high dimensional functions, each dimension is split into cells and employs dimensional search making sure to balance exploration and exploitation. It is impemented ona second order closed loop system with the reference signal as the step signal. The objective function of the PID controllerr converges to the optimal value quickly, around the end of the tenth iteration of the FORL. Although the Particle Swarm Optimization algorithm is most steady and the Genetic algorithm has consistent fluctuations through all iterations, it is shown that when compared with algorithms like Genetic Algorithms(GA), Particle Swarm Optimization(PSO), Evolutionary Programming(EP), etc., the FORL has a superior performance in terms of rise time, settling time and overshoot percentage all considered together. It is concluded that the FORL algorithm is the best function optimization technique for high dimensional systems.

Tran et.al(2020) propose a PSO-EP algorithm in their work, an extension to the Particle Swarm Optimization introduced by Kennedy et.al(1995), and FLC(Fuzzy Logic Control) to optimize PID controller gain parameters for complicated non linear systems. In this paper, the PSO-EP is applied to an unmanned aerial vehicle, a tricopter model based on Newton-Euler formulation with six degrees of freedom, and the algorithm’s performance is tested based on its response time, accuracy, precision, reliability and steady state response or stability. The hybrid model is presented with an updated velocity and position strategy and a mutated EP strategy. A fuzzy control with a triangular membership function is combined with the conventional PID control for

better performance. The flight hovering mode is monitored and the convergence of the intgra absolute error is plotted with the optimal as that for conventional PID tuning method of Ziegler Nichols. Most simulations gave a high performance in terms of overshoot, precision and reliability and it is concluded that in each pilot channel: roll, pitch and yaw, the proposed model has the finest performance compared to other controller designs.

Guan et.al(2020) present the process of designing a PID controller for a non-linear system with update rules based on Reinforcement Learning. The actor critic policy is governed by the RBF(Radial Basis Function) Network, a three layered neural network, which has the input as system error, to design the new update rules. Temporal Difference error is taken into consideration and based on its performance index the gradient descent method is assumed. Model-free design is proposed since it suits most complex real world systems. The proposed algorithm is simulated on a non linear system with user specific learning rates as design specifications. The reference point and output graphs are compared and a temporal difference error graph is presented and shown that due to the high non linearity of the system, there is very high overshoot percentage. It is concluded that with low storage and computational costs, the proposed scheme is efficient for complex, non linear systems.

Mukhopadhyay et.al(2019) in their work, explore advantage actor critic method to design a self-tuning PID controller and compare its performance with that of a deep Q network Actor Critic method, a deep reinforcement learning algorithm. The neural networks were implemented using Pytorch and Keras libraries. The hyperparameters were chosen using heuristics to guarantee acceptable leraning. The rise time, settling time and overshoot percentage is compared for the different algorithms and it was found that DQNAPID performed better than A2CAPID, DQN and A2C algorithms by a landslide even without prior tuning and process modelling knowledge but for sinusoidal signals it suffers phase lag and lesser noise diminution. It is concluded along with these results, that Proximal Policy Optimization may be employed to realize these algorithms physically.

III. Observations and Conclusions

AI techniques applied to a variety of fields have improved the extent of problem solving, accuracy and time management in each one of them. Machine Learning and Deep Learning methods are being applied to many industrial fields. While some methods of ML involve considerate human interference, Reinforcement Learning is one that uses the least of it and Deep Neural networks aim to work on larger data sets and further minimizing human supervision. Reinforcement learning and Neural networks are making huge progresses as we have seen in the survey. For this survey, we have considered various Reinforcement learning algorithms and Neural network architectures and different combinations of exploration strategies and learning policies, along with comparisons of the same.

We can go beyond the existing methods and implement DNNs for better performance of the control element. We can employ the best ideas from the research papers considered to improve the scope in terms of generality of application as well as complexity and accuracy to cater to most exceptional cases, and do it well, and that can be done by scheduling PID gains following Zhao et.al(1993) for gain scheduing of a PID controller using Fuzzy logic, implementing CACLA, employing Actor-Critic architecture with Gaussian exploration startegy Temporal Difference learning and most importantly, implementing Convolutional Neural Networks that learn from the training data using the aforementioned reinforcement learning techniques. Essentially the proposal is an extension to the existing methods by switching up Reinforcement Learning and Fuzzy Logic to Deep Reinforcement Learning and using the best fitting algorithms and strategies to suit practical systems controlled in the industries today.

Bibliography

Atkeson, C. G., & Santamaria, J. C. (1997, April). A comparison of direct and model-based reinforcement learning". In *Proceedings of international conference on robotics and automation* (Vol. 4, pp. 3557-3564). IEEE.

Baker, B., Gupta, O., Naik, N., & Raskar, R. (2016). Designing neural network architectures using reinforcement learning. *arXiv preprint arXiv:1611.02167*.

Bradtke, S. J., & Barto, A. G. (1996). Linear least-squares algorithms for temporal difference learning. *Machine learning*, *22*(1-3), 33-57.

Eberhart, R., & Kennedy, J. (1995, November). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks* (Vol. 4, pp. 1942-1948). Citeseer.

Florensa, C., Duan, Y., & Abbeel, P. (2017). Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*.

Guan, Z., & Yamamoto, T. (2020, July). Design of a Reinforcement Learning PID controller. In *2020 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-6). IEEE.

Hansen, L. K., & Salamon, P. (1990). Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, *12*(10), 993-1001.

Howell, M. N., Gordon, T. J., & Best, M. C. (2000). The application of continuous action reinforcement learning automata to adaptive PID tuning.

Lee, C. C., & Berenji, H. R. (1989, September). An intelligent controller based on approximate reasoning and reinforcement learning. In *Proceedings. IEEE International Symposium on Intelligent Control 1989* (pp. 200-205). IEEE.

Leuenberger, G. (2017). *Actor-Critic reinforcement learning with neural networks in continuous games* (Doctoral dissertation, Faculty of Science and Engineering).

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mukhopadhyay, R., Bandyopadhyay, S., Sutradhar, A., & Chattopadhyay, P. (2019, July). Performance Analysis of Deep Q Networks and Advantage Actor Critic Algorithms in Designing Reinforcement Learning-based Self-tuning PID Controllers. In *2019 IEEE Bombay Section Signature Conference (IBSSC)* (pp. 1-6). IEEE.

Onat, A., Kita, H., & Nishikawa, Y. (1998, May). Recurrent neural networks for reinforcement learning: Architecture, learning algorithms and internal representation. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)* (Vol. 3, pp. 2010-2015). IEEE.

O'Donoghue, B., Munos, R., Kavukcuoglu, K., & Mnih, V. (2016). Combining policy gradient and Q-learning. *arXiv preprint arXiv:1611.01626*.

Nagendra, S., Podila, N., Ugarakhod, R., & George, K. (2017, September). Comparison of reinforcement learning algorithms applied to the cart-pole problem. In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)* (pp. 26-32). IEEE.

Najim, K., & Poznyak, A. S. (1996). Multimodal searching technique based on learning automata with continuous input and changing number of actions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *26*(4), 666-673.

Peters, J., & Schaal, S. (2006, October). Policy gradient methods for robotics. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2219-2225). IEEE.

Psaltis, D., Sideris, A., & Yamamura, A. A. (1988). A multilayered neural network controller. *IEEE control systems magazine*, *8*(2), 17-21.

Shang, X. Y., Ji, T. Y., Li, M. S., Wu, P. Z., & Wu, Q. H. (2013, September). Parameter optimization of PID controllers by reinforcement learning. In *2013 5th Computer Science and Electronic Engineering Conference (CEEC)* (pp. 77-81). IEEE.

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014, June). Deterministic policy gradient algorithms.

Specht, D. F. (1991). A general regression neural network. *IEEE transactions on neural networks*, *2*(6), 568-576.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, *3*(1), 9-44.

Sutton, R. S. (1991). Integrated modeling and control based on reinforcement learning and dynamic programming. In *Advances in neural information processing systems* (pp. 471-478).

Tesauro, G. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*(3), 58-68.

Tran, H. K., Chiou, J. S., & Dang, V. H. (2020). New Fusion Algorithm-Reinforced Pilot Control for an Agricultural Tricopter UAV. *Mathematics*, *8*(9), 1499.

Van Hasselt, H., & Wiering, M. A. (2007, April). Reinforcement learning in continuous action spaces. In *2007 IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning* (pp. 272-279). IEEE.

Zhao, D., Wang, H., Shao, K., & Zhu, Y. (2016, December). Deep reinforcement learning with experience replay based on SARSA. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)* (pp. 1-6). IEEE.

Zhao, Z. Y., Tomizuka, M., & Isaka, S. (1993). Fuzzy gain scheduling of PID controllers. *IEEE transactions on systems, man, and cybernetics*, *23*(5), 1392-1398.