Information Acquisition Driven by Reinforcement in Non-Deterministic Environments

Naresh Babu Bynagari[1*], **Ruhul Amin**[2]

[1]Director of Sales, Career Soft Solutions Inc, 145 Talmadge rd Edison NJ 08817, Middlesex, **USA**
[2]Senior Data Entry Control Operator (IT), ED-Maintenance Office, Bangladesh Bank (Head Office), Dhaka, **BANGLADESH**

[*]E-mail for correspondence: naresh@careersoftusa.com

Source of Support: None **No Conflict of Interest:** Declared

ABSTRACT

What is the fastest way for an agent living in a non-deterministic Markov environment (NME) to learn about its statistical properties? The answer is to create "optimal" experiment sequences by carrying out action sequences that maximize expected knowledge gain. This idea is put into practice by integrating information theory and reinforcement learning techniques. Experiments demonstrate that the resulting method, reinforcement-driven information acquisition (RDIA), is substantially faster than standard random exploration for exploring particular NMEs. Exploration was studied apart from exploitation and we evaluated the performance of different reinforcement driven information acquisition variations to that of traditional random exploration.

Keywords: Non-deterministic Markov environment (NME), Reinforcement driven information acquisition (RDIA), Modeling agent-environment interaction, Q-learning

INTRODUCTION

Learning must be accounted for in computational theories of agent-environment interaction. To generate and retain intelligent agents, it is vital to learn. Robust real-world robots cannot be created just through precise programming. The real world is far too complex, idiosyncratic, and uncertain to know ahead of time, and computer languages are far too rigid and inflexible to make it possible to program alone. At least some of the burden of skill acquisition must be carried by intelligent agents. Furthermore, because the world does not stand still, agents must learn new abilities and adapt old ones to changes in the environment in order to maintain a high level of performance (Bynagari, 2018).

Though there are many different types of learning and many different things that an agent might learn, all learning ultimately boils down to learning control. Anything learned is only valuable in terms of its impact on the agent's interaction with its environment, or in terms of the agent's ability to manipulate the environment to achieve the desired result (Ganapathy, 2017).

It's necessary to model the environment for efficient reinforcement learning. What is the most cost-effective way to obtain a model of a non-deterministic Markov environment (NME)? The method described in this paper, known as Reinforcement Driven Information Acquisition (RDIA), builds on previous work on "query learning" and "experimental design" (Fedorov, 1972) for an overview, and (Baum, 1991; MacKay, 1992; Hwang et al., 1991; Plutowski et al., 1994; Cohn, 1994) for more recent contributions) as well as "active exploration" (Schmidhuber, 1991a; Schmidhuber, 1991b; Storck, 1994; Paruchuri, 2015). Information gain and reinforcement learning are combined in this strategy. The latter is utilized to create exploratory tactics that make use of the former. Experiments show that reinforcement-driven information acquisition has a number of significant benefits.

Objectives of this Study

This study is aimed to model the environment for efficient reinforcement learning using Reinforcement Driven Information Acquisition (RDIA).

LITERATURE REVIEW

The fundamental concepts of reinforcement learning are discussed in this section. To begin, we'll go through a basic model of agent-environment interaction. The principles of Markov decision processes are then discussed, as well as Q-learning (Watkins, 1989; Vadlamudi, 2018), a prominent reinforcement learning technique. However, a comprehensive examination of Markov decision processes and reinforcement learning is beyond the scope of this article (Bynagari, 2016; Bynagari & Fadziso, 2018; Neogy & Bynagari, 2018). Throughout the article, we'll concentrate on Q-learning and the challenges it faces as a result of non-Markov decision processes. Other algorithms (Barto et al., 1983; Holland, 1986; Sutton, 1990; Williams, 1986) have met with the same fate. The reader may wish to study (Bertsekas, 1987) and (Watkins, 1989), for a more thorough examination of Markov decision processes and Q-learning as well as for a broad overview of reinforcement learning (Whitehead and Ballard, 1991; Ganapathy, 2016).

Modeling Agent-environment Interaction

The paradigm of agent-environment interaction shown in Figure 1 is frequently utilized in reinforcement learning. Two synchronized finite state automatons interact in a discrete-time cyclical process to represent the agent and the environment in this paradigm.

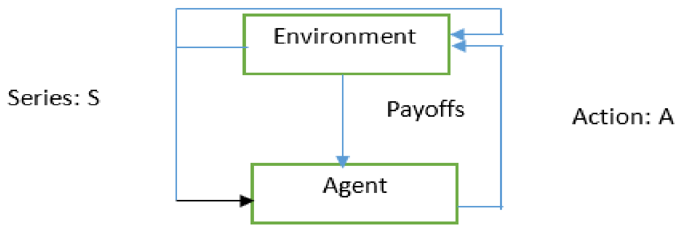

Figure 1: A simple model of agent environment interaction

The following sequence of events occurs at each time point.

- The agent detects the environment's status.
- The agent picks an action to take based on the present condition.
- The environment transitions to a new state and creates a payout based on the present state and the action chosen by the agent.
- The commission is returned to the agent.

The Environment

A Markov decision process is used to model the environment. A Markov decision process is defined by the tuple (S, A, T, R), where S represents the set of potential states, A represents the set of possible actions, T represents the state transition function, and R represents the reward function. At each one time, the environment is in one of S's states and accepts one of A's actions. Typically, S and A are believed to be discrete and finite. The transition function, T, models state transitions by mapping state-action pairings into new states (T: S x A -+ S). In general, the transition function is probabilistic, and it is usually expressed in terms of a collection of transition probabilities, $p_{x,y}\ (a)$?

$$P_{x,y}\ (a) = Prob\ T(x,a) = y)$$

The reward function R, which transforms state-action pairs into scalar-valued rewards ($R: S\ x\ A\ \rightarrow \mathbb{R}$), determines the payoffs provided by the environment. It's also possible that the reward function is probabilistic.

The effects of actions (i.e., the next state and the immediate reward generated) in a Markov decision process (MDP) are solely dependent on the present state. This type of process model is stated to be memoryless and satisfy the Markov property. The Markov feature is crucial to this environment model because it indicates that knowing the current state is always sufficient for optimum control (i.e., maximizing the reward received overtime) (Bertsekas, 1987; Vadlamudi, 2017).

As a result, while it may be conceivable to create action-selection strategies that rely on additional information (a history trace), these strategies will never be able to surpass the best decision strategies that rely just on current state knowledge. The agent is in charge of coming up with control actions. It detects the current state, chooses an action, observes the new state, and rewards the result at each time step (Vadlamudi, 2016). As a form of learning feedback, rewards are used. A control policy, which prescribes an action to take for each state, is one approach to specify an agent's behavior. In formal terms, a policy f is a function from states to actions ($f: S \rightarrow A$), with $f(x)$ denoting the action to be taken in state x.

The goal of reinforcement learning is for the agent to develop a control policy that maximizes some measure of total reward over time (Bynagari, 2015). Any number of reward measures can be employed in theory, but the most common is one based on a discounted total of the benefit received over time (Bynagari, 2017). This amount is known as the return, and it is defined as follows for time t:

$$return\ (t) = \sum_{n=0}^{\infty} \gamma^n r_{t+n}$$

where y, also known as the temporal discount factor, is a constant that ranges from 0 to 1, and r_{t+n} is the reward received at time t + n. The agent's goal is to find a strategy that maximizes the expected return because the process may be stochastic for time t, defined as $V_f(x)$ the value function for policy f, as the expected return for a fixed policy f, given that the process starts in state x and then follows policy f. The agent's goal is to find f^*, a policy that is uniformly excellent for all feasible states. To put it another way, find f^* in such a way that-

$$V_{f*}\ (x) = \max_f V_f\ \ (x)\ \forall_x\ \in S$$

The fact that f^* is properly specified and guaranteed to exist is an important property of MDPs. The Optima& Theorem from dynamic programming [9], in particular, assures that for a discrete temporal, discrete-state Markov decision process, an optimal deterministic policy exists at all times. A policy f is also optimal if and only if it fulfills the following relationship:

$$Q_f\ \big(x, f(x)\big) = \max_{a\ \in A}\Big(Q_f(x,a)\Big)\ \forall_x\ \in S$$

The action-value function $Q_f\ (x, a)$ is defined as the expected return if the agent starts in state x, performs action a once, and then follows policy f [9, 10].

$f^*\ (x) = a \text{ such that } Q_{f*}\ (x, a) = \max_{a\in A}\left(Q_f(x, b)\right)\ \forall_x\ \in S$ states that a policy is optimal if and only if it prescribes an action that maximizes the local "action-value" in each state. That is to say,

$$f^*\ (x) = a \text{ such that } Q_{f*}\ (x, a) = \max_{a\in A}\left(Q_f(x, b)\right)\ \forall_x\ \in S$$

And

$$V_{f*}\ (x) = \max_{a\in A}\left(Q_f(x, a)\right)\ \forall_x\ \in S$$

The set of action-values for which $f^*\ (x) = a \text{ such that } Q_{f*}\ (x, a) = \max_{a\in A}\left(Q_f(x, b)\right)\ \forall_x\ \in S$ holds for a particular MDP is unique. The ideal action-value function Q* for the MDP is said to be defined by these values. The best policy can be determined directly using dynamic programming techniques (Bellman, 1957, Bertsekas, 1987; Ross, 1983; Ganapathy, 2018) if an MDP is thoroughly known (including the transition probabilities and reward distributions). However, in many circumstances, the environment's structure and dynamics remain unknown. In these conditions, the agent cannot immediately compute the best policy; instead, it must investigate its surroundings and learn an effective control policy through trial and error (see Table 1).

Table 1: A simple version of the one-step Q-learning algorithm

```
Q ← a set of initial values for the action-value function
(e.g uniformly zero)
Repeat forever;
    1. $x$ ← the current state
    2. Select an action $a$ to execute that is usually
       consistent with $f(x)$ but occasionally an
       alternate.
    3. Execute action $a$, and let $y$ be the next state and
       $r$ be the reward received.
    4. Update $Q(x, a)$:
          $Q\ (x, a) \leftarrow (1 - \alpha)Q(x, a) + \alpha[r + \gamma U(y)]$
       Where $U(y) = Q(y, f(y))$
       Here for each $x \in S: f(x) \leftarrow$ a such that
       $Q(x, a) = max_{b\in A} Q(x, b)$
```

Q-Learning / Basic Setup

An NME is home to an agent. The environment is in state S(t) (one of n possible states S1; S2;…Sn) at a discrete-time step t, and the agent acts a(t) (one of m possible actions a_1; a_2;…a_m). This has an impact on the environment: If S(t) = S_i and a(t) = a_j, then S(t + 1) = S_k with probability $p_{ijk}\ S(t + 1) = S_k$. There is reinforcement R(t) at particular moments t.

The goal is to maximize the discounted total of future reinforcement $\sum_{k=0}^{m} \gamma^k R(t + k + 1)\ (where\ 0 < \gamma < 1)$ at time t (where 0 1). For this, we employ Watkins' Q-learning (Watkins, 1989): The agent's evaluation Q(S; a) (which starts at zero) for the state/action pair is (S; a). The algorithm's primary loop is as follows:

1. Take note of the current situation S(t). Pick p ϵ [0; 1] at random. If p ≤ μ ϵ [0; 1], choose a number at random (t). Otherwise, choose a (t) so that Q (S (t); a (t)) is the largest.
2. Run a (t), paying attention to S(t + 1) and R(t).
3. $Q(S(t), a(t)) \leftarrow (1 - \beta)Q(S(t), a(t)) + \beta(R(t) + \gamma\ max_b Q(S(t + 1)\ where\ 0\ < \gamma\ < 1, 0\ < \beta\ < 1$ 1st step

Methods

Model Building with reinforcement driven information acquisition

The goal of our agent is to create a model of the p_{ijk} transition probabilities. RDIA is a type of unsupervised reinforcement learning that is investigated separately from goal-directed reinforcement learning tasks "It builds on prior research on active exploration" (Schmidhuber, 1991b; Schmidhuber, 1991a; Thrun and Moller, 1992).

Previous approaches were limited to

- deterministic settings (they did not address the overall problem of learning a model of a nondeterministic NME's statistical features), and
- Were built on ad-hoc elements rather than relying on notions from information theory.

Collecting Machine Learning estimates

The agent has a counter cij for each state/action combination (or experiment) (S_i; a_j), whose value at time t, c_{ij} (t), matches the number of the agent's past experiences with that state/action pair (or experiment) (Si; aj). In addition, there are n counters cijk for each state/action pair (S_i; a_j), $k\ = 1 \ldots n$. The number of past encounters the agent had with (S_i; a_j), where the following state was S_k, is equal to the value of c_{ijk} at time$t$, c_{ijk}(t). It's worth noting that c_{ij} (t) = $\sum_k c_{ijk}\ (t)$. If c_{ij} (t) > 0 at time t, then

$$p^*_{ijk}\ (t) = \frac{c_{ijk}\ (t)}{c_{ij}(t)}$$

To some degree arbitrarily, $p^*_{ijk}\ (t) = 0$, if c_{ij} (t) = 0. As a result, the $p^*_{ijk}\ (t)$ do not satisfy the conditions of a probability distribution before the agent has undertaken any trials of the kind (S_i; a_j). For c_{ij} (t) > 0, the $p^*_{ijk}\ (t)$ construct a maximum likelihood model of the probabilities of the alternative next states (compatible with the agent's previous experiences).

Measuring information gain

If the agent conducts an experiment by performing action a (t) = a_j in state S(t) = S_i, and the new state is S(t + 1) = S_k, then $p^*_{ijk}\ (t)$ will, in general, differ from $p^*_{ijk}\ (t + 1)$. The agent has gained a piece of information by watching the outcome of the experiment, which will improve the accuracy of the estimators. We'll go through three different ways to measure the agent's success in the next section

(Neogy & Paruchuri, 2014; Vadlamudi, 2015). The agent's progress will be used to strengthen it in all three scenarios.

i. According to standard statistics, the approximate confidence area of a multinomial distribution satisfies $P\left(E^2\left(P^*_{ijk,} P_{ijk}\right) < \frac{x^2_{n-1,\alpha}}{cik}\right) = 1-\alpha$, where α is a particular confidence level (Behnen and Neuhaus, 1984, p. 301), and $E^2\left(P^*_{ijk}, p_{ijk}\right) = \sum_{k=1}^{n} \frac{(P^*_{ijk}, p_{ijk})^2}{P_{ijk}}$

The weighted squared error of the P^*_{ijk} estimators is p_{ijk} (Pearson's 2-distance for multinomial distributions is a classic technique of quantifying estimators' departures from real probabilities, (Behnen and Neuhaus, 1984p. 233, 301). The upper bound for E^2 with confidence level $X^2_{n*1,\alpha}$, (the -quantile of the X^2_{n*1}-distribution, which is independent of $c_{ij!}$) divided by c_{ij}. $\frac{1}{c_{ij}}$ is proportional to this upper bound. Reduce the dispersion of the estimators by lowering E^2 upper bound, which is a major goal of optimum experiment design (Cohn, 1994; Fedorov, 1972). As a result, state/action combinations with small counters c_{ij} are preferred when choosing a new experiment. In partially deterministic contexts, however, fewer "deterministic" tests might be preferable to non-deterministic ones.

$$D(t) = \sum_k \left|P^*_{ijk}(t+1) - P^*_{ijk}(t)\right|$$

The information gain (the agent's current progress) for $c_{ij}(t) > 0$ and D(t) = 0 for $c_{ij}(t) = 0$ are used to account for this new factor. Note that for big c_{ij}, $P^*_{ijk}(t+1) - P^*_{ijk}(t)$ is proportional to $\frac{1}{c_{ij}}$, but for deterministic state/action pairs, these differences are zero (unlike $\frac{1}{c_{ij}}$ itself).

ii. We may quantify the agent's progress by measuring the entropy difference between the probability distributions given by the $P^*_{ijk}(t+1)$ values and the $P^*_{ijk}(t)$ values, because the estimators reflect probability distributions over the next states Sk. It is possible to rephrase:

$$D(t) = \left|\sum_k P^*_{ijk}(t+1)\, In P^*_{ijk}(t+1) - \sum_k P^*_{ijk}(t+1) - P^*_{ijk}(t)\right|$$

if c_{ij} is greater than 0. (For P^*_{ijk} = 0, we use the formula $P^*_{ijk}\, In P^*_{ijk}$= 0). The entropy of the related MLM is assumed to be zero if $c_{ij}(t) = 0$ (before the agent has undertaken any experiments of type (S_i, a_j). D(t) will also be 0 in this scenario. D(t) can be proven to be proportional to $\frac{1}{c_{ij}}$ for large c_{ij} once again (and zero in the deterministic case). High entropy probability distributions result in high D(t), which is exactly what we want: high entropy distributions should be investigated more than low entropy distributions (bias towards "nondeterministic" state/action combinations).

iii. The Kullback Leibler distance is a related method for determining probability distribution changes.

$$D(t) = \sum_k d_k(t),$$

Where

$d_k(t) = 0\ if\ P^*_{ijk}(t+1) = 0$ or $P^*_{ijk}(t) = 0$, and $d_k(t) = P^*_{ijk}(t+1) In \frac{(P^*_{ijk}(t+1)}{P^*_{ijk}(t)}$

This metric has qualities similar to the entropy difference described above, but it is more sensitive to increases in the greatest P^*_{ijk} values (emphasis on changes of probability distributions tending towards determinism). The hint is that the agent's progress D(t) is always used as the reinforcement R(t) for the Q-Learning algorithm from the beginning. Because an experiment at time t changes only n estimates (the $nP^*_{ijk}(t+1)$) associated with a_j = a(t) and S_i = S(t)), and because D(t) can always be computed in O(n) operations, the algorithm's total complexity per time step is constrained by O(n) operations (n). The particular definition of D(t) should not make a significant impact because all three D(t) versions promote nondeterminism and are proportional to $\frac{1}{c_{ij}}$ for big c_{ij}.

Results and Discussion

Simulations of reinforcement driven information acquisition

We evaluated the performance of different reinforcement-driven information acquisition variations to that of traditional random exploration (variants of random exploration are the approaches used by the majority of authors). This is a small space. The first test environment has n = 10 states in it. There are 90 different experiments and m = 9 different actions. The likelihoods of a transition are as follows:

$$p_{ijk} = 1\ for\ i = 1, \ldots .9; j = 1, \ldots 9; k = i;\ p_{ijk} = 1\ for\ i = 1, \ldots .9; j = 1, \ldots 9; k = i+1; p_{ijk} = \frac{1}{10} for\ i = 10;\ j = 1, \ldots 10; k = 1, \ldots 10$$

Otherwise, $P_{ijk} = 0$. S_{10} is the only state that enables for a large amount of data to be collected.

RDIA (with settings β = 0.5, γ= 0.9, μ= 0.1) discovers this after a while and establishes a policy that drives the agent to go from every other state to S10 as rapidly as possible.

Random exploration, on the other hand, spends the majority of its time examining the states S1... S9, which is quickly rendered worthless (informative). Table 2 shows the results of random search vs the two RDIA variations that worked best: RDIA based on entropy changes ($D(t) = \left|\sum_k P^*_{ijk}(t+1)\, In P^*_{ijk}(t+1) - \sum_k P^*_{ijk}(t+1) - P^*_{ijk}(t)\right|$

and RDIA based on probability changes

($D(t) = \sum_k \left|P^*_{ijk}(t+1) - P^*_{ijk}(t)\right|$). Reinforcement driven information acquisition takes a bit to figure out where it can learn something new in the beginning. It soon gains traction and outperforms random search. An expanded setting. There are 100 states in the second test environment. There are 10900 different experiments and m = 90 different activities. The likelihoods of a transition are as follows:

$$p_{ijk} = 1\, for\ i = 1, \dots .89; j = 1, \dots 89; k = i;\ p_{ijk} = 1\, for\ i = 1, \dots .89; j = 1, \dots 89; k = i + 1; p_{ijk} = \frac{1}{99} for\ i = 99;\ j = 1, \dots 99; k = 1, \dots 99$$

Otherwise, $P_{ijk} = 0$. The second environment has a total information content of 460.517018 (the sum of the entropies of all state/action pairings' genuine transition probability distributions).

Table 3 shows the number of time steps necessary to reach specified entropy values for random search and RDIA based on entropy changes (with parameters $\beta = 0.5$, $\gamma = 0.9$, $\mu = 0.1$). S100 is the only state that allows for the collection of a large amount of data. RDIA notices this right away and sets up a policy that forces the agent to shift to S100 as soon as feasible from any other state. Random exploration, on the other hand, spends a lot of time on states S_1 through S_{99}.

Table 2: Random search and two reinforcement driven information acquisition variants, the evolutions of the sum of KullbackLeibler distances between estimated and true probability distributions are shown

Experiments	Random Search	RDIA (Entropy)	RDIA (prob. Different)
1	204.92	204.92	204.92
1023	2.96	67.72	65.48
2048	3.39	40.58	21.97
4095	2.73	10.56	5.29
8191	3.71	4.07	3.87
16383	4.10	2.43	2.29
32767	3.42	1.26	1.43
65535	2.02	0.75	0.87
131072	1.57	0.53	0.58
262143	1.06	0.34	0.34

Table 3: Random search and for RDIA based on entropy differences, this table shows the number of time steps required to achieve given entropy values.

Goal Entropy	Random Search	RDIA
170	0.000003	0.0000011
370	0.00000029	0.0000025
459	0.0000000016	0.00000027
460	Unidentified	0.00000068

Because Q-learning requires some time to fix the method for performing tests, the advantage of reinforcement-driven information acquisition is not as evident for small entropy margins as it is in later stages. However, when the entropy margin approaches the optimum, reinforcement-based information collection speeds up by at least an order of magnitude.

- "The Exploitation/Exploration Trade-off." Exploration was studied apart from exploitation in this article. Is there a "best" approach to combine the two? Should reinforcement driven information acquisition be used for which types of goal-directed learning? It is always possible to create settings in which "curiosity" (the desire to learn more about the world) may "kill the cat," or at the very least have a bad impact on exploitation results. Additional studies given in (Thrun and Moller, 1992) demonstrate this: in one habitat described therein, exploration aids in the speeding up of exploitation. Curiosity, on the other hand, slows exploitation in a different habitat. The "exploration/exploitation trade-off" is still a hot topic (Paruchuri, 2015).
- Additional comparisons in the lab. Comparing reinforcement driven information acquisition to better competitors than random exploration, such as Kaelbling's Interval Estimation algorithm (Kaelbling, 1993), will be intriguing.
- Approximators for functions. It will also be interesting to use function approximators like backprop networks to replace the Q-table. Despite the fact that the theoretical foundations of combining Q-learning with function approximators are still poor, previous experimental work by multiple authors suggests that in some circumstances, this may boost performance.

CONCLUSION

Sensor-imposed information restrictions must be dealt with by intelligent control systems. When the agent's sensors provide insufficient information or when the agent must actively regulate its sensors to choose important features, the internal decision issue it faces is invariably non-Markov. It can be difficult to pick up these control skills. We compared the results of several RDIA modifications to the results of ordinary random exploration. The advantage of reinforcement driven information acquisition is not as apparent for tiny entropy margins as it is in later stages since Q-learning takes some time to fix the strategy for executing tests. Reinforcement-based information gathering, on the other hand, accelerates up by at least an order of magnitude as the entropy margin near the optimum. Experiments show that the resulting method, reinforcement driven information acquisition (RDIA), is significantly faster at studying specific NMEs than ordinary random exploration.

REFERENCES

Barto, A. G., R. S. Sutton and C. W. Anderson, 1983. Neuron-like elements that can solve difficult learning control problems, *IEEE Truns. Syst. Man Cybern*. 13 (5): 834-846.

Baum, E. B. 1991. Neural nets that learn in polynomial time from examples and queries. *IEEE Transactions on Neural Networks*, 2(1):5–19.

Behnen, K. and Neuhaus, G. 1984. Grundkurs Stochastik. B. G. Teubner, Stuttgart.

Bellman, R. E. 1983. Dynamic Programming (Princeton University Press, Princeton, NJ, 1957). S. Ross, Introduction to Stochastic Dynamic Programming (Academic Press, New York, 1983).

Bertsekas, D. P. 1987. Dynamic Progrummin~: Deterministic and Stochastic Models (Prentice-Hall, Englewood Cliffs, NJ.

Bynagari, N. B. (2015). Machine Learning and Artificial Intelligence in Online Fake Transaction Alerting. *Engineering International*, 3(2), 115-126. https://doi.org/10.18034/ei.v3i2.566

Bynagari, N. B. (2016). Industrial Application of Internet of Things. *Asia Pacific Journal of Energy and Environment*, 3(2), 75-82. https://doi.org/10.18034/apjee.v3i2.576

Bynagari, N. B. (2017). Prediction of Human Population Responses to Toxic Compounds by a Collaborative Competition. *Asian Journal of Humanity, Art and Literature*, 4(2), 147-156. https://doi.org/10.18034/ajhal.v4i2.577

Bynagari, N. B. (2018). On the ChEMBL Platform, a Large-scale Evaluation of Machine Learning Algorithms for Drug Target Prediction. *Asian Journal of Applied Science and Engineering*, 7, 53–64. Retrieved from https://upright.pub/index.php/ajase/article/view/31

Bynagari, N. B., & Fadziso, T. (2018). Theoretical Approaches of Machine Learning to Schizophrenia. *Engineering International*, 6(2), 155-168. https://doi.org/10.18034/ei.v6i2.568

Cohn, D. A 1994. Neural network exploration using optimal experiment design. In J. Cowan, G. Tesauro, and J. Alspector, editors, Advances in Neural Information Processing Systems (NIPS) 6, pages 679–686. Morgan Kaufmann.

Fedorov. V. V. 1972. Theory of optimal experiments. Academic Press.

Ganapathy, A. (2016). Speech Emotion Recognition Using Deep Learning Techniques. *ABC Journal of Advanced Research*, 5(2), 113-122. https://doi.org/10.18034/abcjar.v5i2.550

Ganapathy, A. (2017). Friendly URLs in the CMS and Power of Global Ranking with Crawlers with Added Security. *Engineering International*, 5(2), 87-96. https://doi.org/10.18034/ei.v5i2.541

Ganapathy, A. (2018). Cascading Cache Layer in Content Management System. *Asian Business Review*, 8(3), 177-182. https://doi.org/10.18034/abr.v8i3.542

Holland, J. H. 1986. Escaping brittleness: the possibilities of general-purpose learning algortihms applied to parallel rule-based systems, in: Muchine Lenrnin~: An Artificial Intelligence Approach II (Morgan Kaufmann, San Mateo, CA, 1986).

Hwang, J., J. Choi, S. Oh, and R. J. Marks. 1991. Query-based learning applied to partially trained multilayer perceptrons. *IEEE Transactions on Neural Networks*, 2(1):131–136, 1991.

Kaelbling. L. P. 1993. Learning in Embedded Systems. MIT Press.

MacKay, D. J. C. 1992. Information-based objective functions for active data selection. Neural Computation, 4(2):550–604, 1992.

Neogy, T. K., & Bynagari, N. B. (2018). Gradient Descent is a Technique for Learning to Learn. *Asian Journal of Humanity, Art and Literature*, 5(2), 145-156. https://doi.org/10.18034/ajhal.v5i2.578

Neogy, T. K., & Paruchuri, H. (2014). Machine Learning as a New Search Engine Interface: An Overview. *Engineering International*, 2(2), 103-112. https://doi.org/10.18034/ei.v2i2.539

Paruchuri, H. (2015). Application of Artificial Neural Network to ANPR: An Overview. *ABC Journal of Advanced Research*, 4(2), 143-152. https://doi.org/10.18034/abcjar.v4i2.549

Plutowski, M., G. Cottrell, and H. White. 1994. Learning Mackey-Glass from 25 examples, plus or minus 2. In J. Cowan, G. Tesauro, and J. Alspector, editors, Advances in Neural Information Processing Systems (NIPS) 6, pages 1135–1142. Morgan Kaufmann.

Schmidhuber J. and Storck, J. 1993. Reinforcement driven information acquisition in nondeterministic environments. Report.

Schmidhuber. J. 1991a. Curious model-building control systems. In Proceedings of the International Joint Conference on Neural Networks, Singapore, volume 2, pages 1458–1463. IEEE press.

Schmidhuber. J. 1991b. A possibility for implementing curiosity and boredom in model-building neural controllers. In J. A. Meyer and S. W. Wilson, editors, Proc. of the International Conference on Simulation of Adaptive Behavior: From Animals to Animats, pages 222 – 227. MIT Press/Bradford Books, 1991.

Storck. J. 1994. Reinforcement-Lernen und Modell bildung in nicht-deterministischen Umgebungen. Fortgeschrittenenpraktikum, Fakult¨at f¨ur Informatik, Lehrstuhl Prof. Brauer, Technische Universit¨at M¨unchen.

Sutton, R.S. 1988. Learning to predict by the method of temporal differences, Mach. Learn. 3 (1): 9-44.

Thrun S. and M¨oller. K. 1992 Active exploration in dynamic environments. In D. S. Lippman, J. E. Moody, and D. S. Touretzky, editors, Advances in Neural Information Processing Systems (NIPS) 4, pages 531–538. Morgan Kaufmann.

Vadlamudi, S. (2015). Enabling Trustworthiness in Artificial Intelligence - A Detailed Discussion. *Engineering International*, 3(2), 105-114. https://doi.org/10.18034/ei.v3i2.519

Vadlamudi, S. (2016). What Impact does Internet of Things have on Project Management in Project based Firms?. *Asian Business Review*, 6(3), 179-186. https://doi.org/10.18034/abr.v6i3.520

Vadlamudi, S. (2017). Stock Market Prediction using Machine Learning: A Systematic Literature Review. *American Journal of Trade and Policy*, 4(3), 123-128. https://doi.org/10.18034/ajtp.v4i3.521

Vadlamudi, S. (2018). Agri-Food System and Artificial Intelligence: Reconsidering Imperishability. *Asian Journal of Applied Science and Engineering*, 7(1), 33-42. Retrieved from https://journals.abc.us.org/index.php/ajase/article/view/1192

Watkins. C. J. C. H. 1989. Learning from Delayed Rewards. PhD thesis, King's College, Oxford, University of Cambridge, England.

Whitehead S. D. and Ballard, D. H.. 1991.A study of cooperative mechanisms for faster reinforcement learning, Technical Report 365, Computer Science Department, University of Rochester, Rochester. NY.

Williams, R. J. 1986. Reinforcement learning in connectionist networks, Technical Report ICS 8605, Institute for Cognitive Science, University of California at San Diego.

--0--