A Self-Organizing Neuro-Fuzzy Q-Network: Systematic Design with Offline Hybrid Learning

John Wesley Hostetter
North Carolina State University
Raleigh, NC, USA
jwhostet@ncsu.edu

Mark Abdelshiheed
North Carolina State University
Raleigh, NC, USA
mnabdels@ncsu.edu

Tiffany Barnes
North Carolina State University
Raleigh, NC, USA
tmbarnes@ncsu.edu

Min Chi
North Carolina State University
Raleigh, NC, USA
mchi@ncsu.edu

ABSTRACT

In this paper, we propose a systematic design process for automatically generating ***self-organizing neuro-fuzzy Q-networks*** by leveraging *unsupervised learning* and an *offline, model-free fuzzy reinforcement learning algorithm called* ***Fuzzy Conservative Q-learning (FCQL)***. Our FCQL offers more effective and interpretable policies than deep neural networks, facilitating human-in-the-loop design and explainability. The effectiveness of FCQL is empirically demonstrated in Cart Pole and in an Intelligent Tutoring System that teaches probability principles to real humans.

KEYWORDS

Fuzzy Logic Control; Neuro-Fuzzy; Hybrid Learning; Unsupervised Learning; Offline Reinforcement Learning; ITS; Pedagogical Policy

ACM Reference Format:
John Wesley Hostetter, Mark Abdelshiheed, Tiffany Barnes, and Min Chi. 2023. A Self-Organizing Neuro-Fuzzy Q-Network: Systematic Design with Offline Hybrid Learning. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023*, IFAAMAS, 10 pages.

1 INTRODUCTION

In the last decade, combining deep neural networks and novel reinforcement learning (RL) algorithms has made solving complex problems possible [6, 55]. Generally speaking, RL is an online and interactive process where an agent explores its environment and exploits what it has learned to maximize an expected cumulative reward [56]. Exploration allows the agent to discover new possibilities beyond what it has witnessed thus far [49]. In some settings, exploration is impossible due to restrictions such as legal recourse [52]. Still, data involving past interactions may be available —one approach to solve this is by offline RL: an agent is trained to maximize an expected reward using only data that has already been collected [36]. In challenging, sensitive, and high-risk domains, computational models' *interpretability* is also highly critical. For example, in healthcare, it is generally more important to learn about the discriminative *interpretable* patterns that capture the informative progression of a disease than to induce an accurate predictive computational model. More importantly, interpretability is a crucial factor to earn *credibility* and *adoption* [32]. Due to the opaque nature of deep neural networks, it is often near impossible for humans to *interpret* or *understand* their decision making [15, 46]. Additionally, Deep RL is sample inefficient. In contrast, imitation learning (IL) relies upon expert demonstrations rather than trying to learn from a large number of training trajectories. In IL, the agent learns the optimal policy by imitating the expert's demonstrations; a basic form of IL is behavior cloning (BC) [57].

An alternative to providing a learning agent with expert knowledge is instructions written by an expert (more formally known as a linguistic control strategy) —via IF-THEN rules that describe an approximate and imprecise causality between input and output called ***fuzzy logic rules*** [32, 35, 62]. To incorporate this kind of expert knowledge into neural networks, a ***neuro-fuzzy network*** was proposed, and its use for the RL problem falls under the umbrella of ***fuzzy RL*** [12]. This led to more interpretable policies, but their heavy dependency upon expert knowledge limits potential applications or dramatically increases the development cost and time —hence the obvious appeal of automated learning techniques.

Automatic discovery of the linguistics [8, 58, 59], and how to produce the IF-THEN relationships [28, 64], remains an ongoing research endeavor; the objective of creating both an *interpretable* and an *accurate* system are contradictory requirements in complex domains [15], and to this day, there is still no universally agreed upon *systematic design process* [7, 16, 30, 50, 58].

We propose a systematic design process for a ***self-organizing neuro-fuzzy Q-network*** by using two learning paradigms in tandem: *unsupervised learning* and an *offline, model-free fuzzy RL algorithm called* ***Fuzzy Conservative Q-learning (FCQL)***. FCQL is primarily based on Fuzzy Q-Learning [20], which treats a fuzzy logic rule as a "state" in the environment —similar to how Tabular Q-Learning [66] behaves —and trains Q-values for each of the rule's possible actions. To prevent overestimating Q-values in the offline setting, we use the updated formula proposed in Conservative Q-Learning (CQL) [34]. *The primary novelty and contribution of this work is the proposal of a systematic design process for a neuro-fuzzy Q-network that works with a model-free offline fuzzy RL algorithm.* To the best of our knowledge, existing methods within fuzzy RL are either developed for online interaction due to their dependency on exploration [9], are unable to accommodate for distributional shift [20], or rely upon the existence of a simulation [10, 22]; furthermore,

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

the only existing work capable of automatic design for offline fuzzy RL is fuzzy particle swarm RL, but this requires a simulation of real system dynamics that may be impractical for many real-world complex applications such as e-learning and healthcare [22]. The effectiveness of FCQL is evaluated in two settings: (1) Cart Pole, where FCQL's performance was compared directly to CQL as well as additional offline RL (and IL) methods, and (2) an Intelligent Tutoring System (ITS), where FCQL was empirically compared to an expert-designed policy, referred to as *Expert*, in a real-world classroom study.

2 METHOD

Figure 1 illustrates the systematic design process for our proposed self-organizing neuro-fuzzy Q-network. In the following, Section 2.1 first gives a brief review of fuzzy logic theory, its intuition, and the motivation behind our proposed systematic design process. Sections 2.2, 2.3, and 2.4 describe our primary contribution —the self-organizing nature of the process. More specifically, Section 2.2 addresses the top left of Figure 1: "Discover Fuzzy Sets with CLIP", and Section 2.3 addresses the top right of Figure 1: "Obtain Candidates for Fuzzy Logic Rules with ECM". These two steps may run in parallel to save computational time. Section 2.4 describes the "Generate Fuzzy Logic Rules with Wang-Mendel" step, which would construct the neuro-fuzzy Q-network for inducing RL policies. This neuro-fuzzy Q-network can be used similarly to how a Deep Q-Network [45] is used in Deep Q-Learning. Finally, Section 2.5 describes the bottom of Figure 1 on how to train the neuro-fuzzy Q-network: "Learn Fuzzy Logic Rules' Q-values with FCQL".

Our proposed self-organizing process can handle complex, high-dimensional tasks, as demonstrated in our main empirical study where an Intelligent Tutoring System is used to teach probability principles to undergraduate students. More importantly, it can adapt to the given data in both online and offline environments. Since we are mainly interested in human-centric tasks for which exploring the environment is not feasible, and since relatively little previous research has explored offline settings, we are primarily concerned with results from offline fuzzy reinforcement learning.

Problem Definition: We assume the state space, $\mathcal{S}$, is continuous and n-dimensionsal (i.e., $\mathcal{S} = \mathbb{R}^n$) —as a result, state $\mathbf{s}$ is a column vector of size n but —for convenience and without loss of generality —we will write as a n-tuple such that (s.t.) $\mathbf{s} = (s_1, s_2, \ldots, s_n)$. Throughout the entirety of this paper, let i denote the i^{th} input dimension (i.e., state attribute). For example, s_i is the i^{th} element/value of $\mathbf{s}$. Our action space, $\mathcal{A}$, is a discrete and finite set.

2.1 The Theory of Fuzzy Logic

Fuzzy logic, fuzzy logic rules, linguistic variables, and their linguistic terms are all by-products of the renowned ***fuzzy set theory*** [68]. Fuzzy set theory is the mathematical study of a type of uncertainty called *impreciseness*. Often, fuzzy set theory is mistakenly compared to probability theory, but the two are distinct mathematical branches concerned with handling different types of uncertainty —in fact, the two can complement one another [32, 69].

For a fuzzy set, an element's membership is [typically] between 0 and 1 [32] —unlike "traditional" set theory, where an element either belongs to a set or it does not. Let U represent the *universe of discourse* which contains all the possible elements for some application of concern [62]. To determine the membership of an element $u \in U$ to a fuzzy set G, we write $\mu_G(u)$ (the μ is a special notation for "*membership function*"), and has a range of $[0, 1]$ where 0 denotes absolute non-membership and 1 denotes absolute membership. As an element's membership approaches 0, the less it belongs to the fuzzy set [68]. The Gaussian membership function is used to calculate elements' memberships to fuzzy sets in this study as it easily allows for back-propagation via gradient descent to occur as opposed to the alternatives (e.g., trapezoidal membership functions) [58]. In this work, the terms "fuzzy sets" and "membership functions" can essentially be used interchangeably [32].

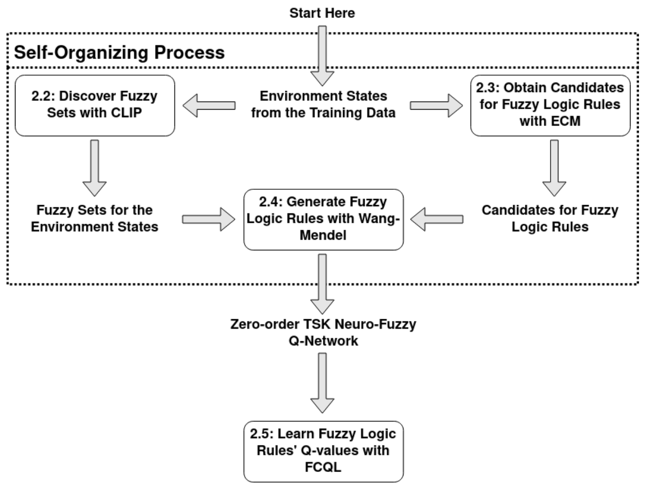

Start Here
Self-Organizing Process
2.2: Discover Fuzzy Sets with CLIP
Environment States from the Training Data
2.3: Obtain Candidates for Fuzzy Logic Rules with ECM
Fuzzy Sets for the Environment States
2.4: Generate Fuzzy Logic Rules with Wang-Mendel
Candidates for Fuzzy Logic Rules
Zero-order TSK Neuro-Fuzzy Q-Network
2.5: Learn Fuzzy Logic Rules' Q-values with FCQL

Figure 1: Diagram of the proposed systematic design process for the self-organizing neuro-fuzzy Q-network.

Returning to the aforementioned by-products of fuzzy set theory, linguistic terms are fuzzy sets that satisfy a set of constraints and have a semantic meaning; the linguistic variable is then simply a variable that can only take on values that are linguistic terms [32]. The implication relationship between the assignment of a linguistic variable(s) to some linguistic term(s) (called the ***antecedents***) and the assignment of another linguistic variable to some linguistic term (called the ***consequents***) is a fuzzy logic rule. The degree of *applicability* or *activation* of a fuzzy logic rule is dependent upon the given information's *relevancy* or *membership* to the fuzzy logic rule's antecedents. The collection of fuzzy logic rules constitutes a ***knowledge base***, and with its decision-making logic (e.g., product-inference engine) [35], a system may be controlled with the inferred output —this is called a ***fuzzy logic controller (FLC)***. Neuro-fuzzy networks are computationally efficient implementations of FLCs that facilitate back-propagation [39].

In the context of self-organizing fuzzy RL (i.e., no a priori knowledge or human expert design), the fuzzy logic rules —and subsequently the linguistic terms they are defined over —are entirely absent at first. Our proposed method is thus essentially two-pronged: (1) discover the areas of interest that fuzzy logic rules can be defined for and (2) determine their corresponding outcomes (i.e., Q-values). The first part is done by a combination of unsupervised learning methods —namely, ***Categorical Learning-Induced Partitioning***

[58] for identifying linguistic terms, ***Evolving Clustering Method*** [28] to obtain candidates for fuzzy logic rules, and then the classic ***Wang-Mendel Method*** [64] for fuzzy logic rule generation. Lastly, offline fuzzy RL learns the fuzzy logic rules' Q-values. Sample code demonstrating our method is publicly available (MIT license) [23].[1]

2.2 Identifying Membership Functions

In order to create fuzzy logic rules, it is necessary to define membership functions that describe fuzzy sets.[2] Our fuzzy logic rules will map linguistic terms that describe the environment's current state, to Q-values of the available actions. In order to discover these linguistic terms, we need a procedure designed specifically for this purpose. In this paper, we chose ***Categorical Learning-Induced Partitioning (CLIP)*** —a quick, single-pass algorithm capable of producing Gaussian membership functions which can improve upon them incrementally online/offline (increased flexibility). Further, CLIP does not require us to determine the number of linguistic terms we need in advance [58], unlike fuzzy C-means [13], for instance. [3] CLIP is inspired by the behavioral category learning process that humans possess [58]. An example of behavioral category learning would be a young infant first seeing a dog and believing that all animals are dogs; however, as the infant grows older, the child begins to *refine* what it means for an animal to be a dog [43, 44, 58]. CLIP works in a very similar manner.

Upon seeing the first state, $\mathbf{s}$, in the training data, D, CLIP will create a fuzzy set —that represents a *concept* [43, 58] —which covers the entire domain for some i. Since CLIP produces fuzzy sets that are defined by Gaussian membership functions [58], then this newly created fuzzy set has parameters $c_i^1 = s_i$ and $\sigma_i^1 = \Phi\left(\sqrt{-\frac{(\min_i - s_i)^2}{\log \epsilon}}, \sqrt{-\frac{(\max_i - s_i)^2}{\log \epsilon}}\right)$, where c_i^1 and σ_i^1 are the center and width of the Gaussian membership function that describe G_i^1, respectively (the superscript 1 is to emphasize that it is the first fuzzy set created in i). A newly created membership function is centered upon the presented value, while $\Phi(\sigma_i^j, \sigma_i^l) := \frac{1}{2}[\sigma_i^j + \sigma_i^l]$ defines a *regulator function* and $j \neq l$. The regulator function's purpose is to preserve each fuzzy set's distinct semantic meaning "…by maintaining a reasonable amount of buffer on either sides of its center, where the boundary of the domain is given as $[\min_i, \max_i]$ and the *minimum membership threshold*, ϵ, is defined s.t. the membership value of any point in the domain should be at least ϵ before regulation" [58]. Thus, if the center of a newly created membership function were to be near one of the boundary's edges, the regulator function would prevent such drastic malformed membership functions from forming —as without the regulator function, the side of the Gaussian membership function closer to the boundary's edge would drop drastically to a membership value of ϵ. By using this regulator function, the newly created membership function would develop the desired Gaussian shape with equal and reasonable amount of spread on both sides of its center.

If a fuzzy set already exists in i but a new state $\check{\mathbf{s}}$ has been encountered ($\check{\mathbf{s}} \neq \mathbf{s}$, as it is some arbitrary state that occurs next within the training data or batch —not necessarily the *successor* to state $\mathbf{s}$ in an episode), then CLIP will calculate a similarity match between the input value $\check{s}_i$ and all existing fuzzy sets in i; this "similarity match" is $\mu_{G_i^j}(c_i^j, \sigma_i^j; \check{s}_i)$ where G_i^j is the j^{th} fuzzy set to be created in i and c_i^j, σ_i^j are its Gaussian membership function's properties, respectively.

The best matched fuzzy set is $\star = \text{argmax}_j \mu_{G_i^j}(c_i^j, \sigma_i^j; \check{s}_i)$ s.t. "$\star$" references the best matched existing fuzzy set in i. If the similarity between $\check{s}_i$ and $G_i^\star$ exceeds a *contrasting threshold* κ, then this fuzzy set, $G_i^\star$, represents $\check{s}_i$ satisfactorily. Otherwise, a new fuzzy set will be created to accommodate for $\check{s}_i$, while adjusting and refining fuzzy sets in i. Formally, this new fuzzy set in i is created by

$$c^{J_i(t)+1} = \check{s}_i$$
$$\sigma^{J_i(t)+1} = \begin{cases} \sigma^L & \text{if } j_i^R = \text{NULL} \\ \sigma^R & \text{if } j_i^L = \text{NULL} \\ \Phi(\sigma^L, \sigma^R) & \text{otherwise} \end{cases} \tag{1}$$

where $J_i(t)$ is the number of fuzzy sets that have been created for i *thus far* at time-step t, and

$$\sigma^L = \Phi\left(\sqrt{-\frac{(c^{j_i^L} - \check{s}_i)^2}{\log \epsilon}}, \sigma^{j_i^L}(t)\right) \tag{2}$$

$$\sigma^R = \Phi\left(\sqrt{-\frac{(c^{j_i^R} - \check{s}_i)^2}{\log \epsilon}}, \sigma^{j_i^R}(t)\right). \tag{3}$$

After the creation of the new fuzzy set, the existing fuzzy sets in i accommodate this new addition —this ensures that the fuzzy sets within i remain distinct from one another. For computational simplicity, only the left and right neighbors (if they exist) of the newly created fuzzy set are adjusted/refined. This leads to three possibilities, the new fuzzy set has:

(1) no left neighbor (i.e., j_i^L = NULL by Eq. 4), then the right neighbor is fixed: $\sigma^{j_i^R}(t+1) = \sigma^{J_i(t)+1}$ via Eq. 3
(2) no right neighbor (i.e., j_i^R = NULL by Eq. 5), then the left neighbor is fixed: $\sigma^{j_i^L}(t+1) = \sigma^{J_i(t)+1}$ via Eq. 2
(3) left & right neighbors (by Eq. 4 and Eq. 5), then both are: $\sigma^{j_i^L}(t+1) = \sigma^{j_i^R}(t+1) = \sigma^{J_i(t)+1}$ via Eq. 2 & Eq. 3

The following formulas are used to determine neighboring fuzzy sets' eligibility for modification:

$$j_i^L = \begin{cases} \text{NULL} & \text{if } c^{j_i} \geq \check{s}_i \text{ for } 1 \leq j_i \leq J_i(t) \\ \arg\min_{c^{j_i} < \check{s}_i} |c^{j_i} - \check{s}_i| & \text{otherwise} \end{cases} \tag{4}$$

$$j_i^R = \begin{cases} \text{NULL} & \text{if } c^{j_i} \leq \check{s}_i \text{ for } 1 \leq j_i \leq J_i(t) \\ \arg\min_{c^{j_i} > \check{s}_i} |c^{j_i} - \check{s}_i| & \text{otherwise.} \end{cases} \tag{5}$$

Repeat for $1 \leq i \leq n$ using every unique state within D.

2.3 Obtaining Candidates for Fuzzy Logic Rules

The identification of relevant or important fuzzy logic rules is closely related to the identification of exemplars, clusters, or prototypes within input-output data [8, 28, 48, 67]. In offline RL, we have no associated output behavior to the given states *before* learning

[1]https://github.com/johnHostetter/AAMAS-2023-FCQL
[2]Technically, *some* antecedents of the fuzzy logic rules could contain "hard" partitions (e.g., true/false or categorical data).
[3]An alternative incremental procedure called Discrete Incremental Clustering exists, but it produces trapezoidal membership functions instead [59].

Q-values. Despite this, we can still identify "regions of interest" or exemplars that are to be modelled by the ***Evolving Clustering Method (ECM)*** [28]. ECM further partitions the input space to facilitate the creation of fuzzy logic rules. The ECM algorithm is a quick, single-pass procedure that dynamically estimates the number of clusters within the data, as well as their current centers, after being given a threshold value called *Dthr*; *Dthr* affects the number of clusters identified, and subsequently, the number of fuzzy logic rules possibly identified. To select an appropriate value for *Dthr*, it can be adjusted to limit the growth of the knowledge base —essentially, the larger the *Dthr* value is, the smaller the number of identified candidates for fuzzy logic rules will become, but this runs the risk of losing potential approximation power.

For a data point to be within a cluster, it must have a distance that is less than the threshold *Dthr*. ECM measures distance by relying upon the *general Euclidean distance* [28] defined as $||\mathbf{s}-\check{\mathbf{s}}|| = \left(\sum_{i=1}^{n} |s_i - \check{s}_i|^2\right)^{\frac{1}{2}} / n^{\frac{1}{2}}$ where $\mathbf{s}, \check{\mathbf{s}} \in \mathcal{S}$.

Upon sampling state $\mathbf{s}$ from the training data, ECM creates a cluster where the center, $\mathbf{x}^1$, is equal to $\mathbf{s}$, and sets the cluster's radius, r^1, to zero. As ECM begins to see other data, it will adjust this cluster's estimated center as well as its radius. Clusters are no longer updated once their radius reaches the threshold value, *Dthr* —effectively serving as a throttle to a cluster's maximum allowed "area of influence" or "receptive field" within the input space.

ECM can be described with five steps:

(1) Create the first cluster, $(\mathbf{x}^1, r^1)$, by taking the first state, $\mathbf{s}$, from the training data as the center $\mathbf{x}^1 = \mathbf{s}$, setting the radius r^1 to zero and recording the support, $supp^1 = 1$.
(2) If all examples are processed, exit. Else, the distances between the current state, $\mathbf{s}$, and all M existing clusters are calculated: $||\mathbf{s} - \mathbf{x}^m||$, for $m = 1, 2, \dots, M$.
(3) If $||\mathbf{s} - \mathbf{x}^m|| \leq r^m$ for $m = 1, 2, \dots, M$, then state $\mathbf{s}$ belongs to a cluster $(\mathbf{x}^\star, r^\star)$ with the minimum distance $||\mathbf{s} - \mathbf{x}^\star|| = \min(||\mathbf{s} - \mathbf{x}^m||)$ subject to the constraint $||\mathbf{s} - \mathbf{x}^m|| \leq r^m$ for $m = 1, 2, \dots, M$; if the previous conditions are met —no cluster is added or updated, and the algorithm returns to Step 2. Else, go to the next step.
(4) Find the cluster, $(\mathbf{x}^\dagger, r^\dagger)$, that satisfies $\dagger = \arg\min\left(||\mathbf{s} - \mathbf{x}^m|| + r^m\right)$ for $m = 1, 2, \dots, M$.
(5) If $\min\left(||\mathbf{s} - \mathbf{x}^m|| + r^m\right) > 2 \times Dthr$, the state $\mathbf{s}$ does not belong to any existing clusters; a new cluster is created via Step 1, and the algorithm goes to Step 2. Else, the cluster, $(\mathbf{x}^\dagger, r^\dagger)$, is updated by moving $\mathbf{x}^\dagger$, increasing the radius, $r^\dagger$, and incrementing the support, $supp^\dagger$: specifically, $r^\dagger = \min\left(||\mathbf{s}-\mathbf{x}^m||+r^m\right)/2, \mathbf{x}^\dagger = \left((supp^\dagger-1)\times\mathbf{x}^\dagger+\mathbf{s}\right)/supp^\dagger$, and $supp^\dagger$ is incremented by 1. Return to Step 2.

The collection of clusters' centers obtained by ECM are then candidates to generate fuzzy logic rules —these candidates are denoted by X s.t. $X = \{x^1, x^2, \dots, x^M\}$. These candidates will reappear in Section 2.4 where the Wang-Mendel Method will prepare them into a fuzzy logic rule format and select only those that are unique. Notice that the candidates may not *necessarily* equal the original states' values —instead, they are the result of a few states that are brought together by similarity. The purpose of this "preprocessing step" is to assist the Wang-Mendel Method by eliminating redundant state observations that are closely similar. Without this, in high-dimension tasks, it becomes possible for the Wang-Mendel Method to produce a number of fuzzy logic rules that grow linearly with respect to the original training data (i.e., $|D|$) [64]. Therefore, we hope that $|X| << |D|$ before applying the Wang-Mendel Method.

2.4 Generating the Fuzzy Logic Rules

The Wang-Mendel Method is one of the most widely used fuzzy logic rule generation methods [29, 64] and was originally designed for supervised learning by following a five-step procedure. For our task, we mainly leverage the second step.

Suppose X is a set of candidates for fuzzy logic rules where each element is $\mathbf{x} = (x_1, x_2, \dots, x_n)$. A *candidate fuzzy logic rule* is generated by converting $\mathbf{x}$ into its fuzzy representation —a Cartesian product of fuzzy sets, s.t. each fuzzy set within this Cartesian product operation belongs to its respective input-dimension. To determine which fuzzy sets are in this Cartesian product, for $1 \leq i \leq n$ of $\mathbf{x}$, we select the fuzzy set that x_i attains the highest degree of membership to:

$$\star = \arg\max G_i^j(x_i) \text{ for } 1 \leq j \leq J_i \tag{6}$$

where J_i is the number of fuzzy sets that have been partitioned for i (notice that J_i is no longer a function of time, t, as it was earlier —this is because J_i is now a constant value as CLIP is done). A fuzzy logic rule maps a Cartesian product of fuzzy sets to some decision, which would be the Q-values corresponding to possible actions in fuzzy RL. However, as Q-values are *not* known at this point, the zero vector, $\mathbf{0}$, (length of $|\mathcal{A}|$) initializes their values. Thus, given a candidate $\mathbf{x}$, we generate a fuzzy logic rule in the form:

$$Rule_k : (G_1^\star, G_2^\star, \dots, G_n^\star) \mapsto \mathbf{0} \tag{7}$$

where $\star$ satisfies Eq. 6 for $1 \leq i \leq n$ and $Rule_k$ means the k^{th} fuzzy logic rule ($k \geq 1$); rules with identical antecedents are eliminated to prevent redundancy in the knowledge base.

2.5 Fuzzy Conservative Q-Learning (FCQL)

The k^{th} fuzzy logic rule, $Rule_k$, must learn the Q-values of $|\mathcal{A}|$ discrete actions available at the current state $\mathbf{s}$. For $Rule_k$, a_k^ℓ and Q_k^ℓ, denote the ℓ^{th} possible action and its corresponding Q-value for the k^{th} fuzzy logic rule, respectively. The fuzzy logic rules defined previously with Eq. 7 are expanded into a trainable format:

$$Rule_k : \texttt{IF } s_1 \texttt{ is } G_1^\star \texttt{ and ... and } s_n \texttt{ is } G_n^\star \tag{8}$$
$$\texttt{THEN Action } a_k^1 \texttt{ is } Q_k^1 \texttt{ and ... and Action } a_k^{|\mathcal{A}|} \texttt{ is } Q_k^{|\mathcal{A}|}$$

Let $a^\dagger$ be the selected action in $Rule_k$ for some transition in D. The actual Q-value of $a^\dagger$ is

$$Q(\mathbf{s}, a^\dagger) = \frac{\sum_{k=1}^{K} Rule_k(\mathbf{s}) \times Q_k^\dagger}{\sum_{k=1}^{K} Rule_k(\mathbf{s})}. \tag{9}$$

where $Rule_k(\mathbf{s}) = \prod_i^n G_i^\star(s_i)$ and $\star$ is the chosen linguistic term for $Rule_k$'s i^{th} linguistic variable (i.e., the terms that made $Rule_k$ in Eq. 7). Eq. 9 defines a zero-order Takagi-Sugeno-Kang (TSK) neuro-fuzzy network with product-inference engine and a knowledge base of size K [15, 62]. This TSK neuro-fuzzy network can theoretically approximate any real continuous function on a closed and bounded domain to any given degree of accuracy [33, 41, 61, 63]. To learn Q-values in the offline fuzzy RL setting, we combine Fuzzy Q-Learning with the CQL framework by augmenting the standard Bellman error objective function; a few variants exist [34], but in this paper, the training objective

$$\overbrace{\min_{Q} \alpha\, \mathbb{E}_{\mathbf{s}\sim D}\left[\log \sum_{a} \exp\left(Q(\mathbf{s}, a)\right) - \mathbb{E}_{a^{\dagger}\sim\hat{\pi}_{\beta}(a^{\dagger}|\mathbf{s})}\left[Q(\mathbf{s}, a^{\dagger})\right]\right]}^{\text{Augmentation by CQL}}$$
$$\underbrace{+\,\frac{1}{2}\,\mathbb{E}_{\mathbf{s},a^{\dagger},\mathbf{s}'\sim D}\left[\left(Q(\mathbf{s}, a^{\dagger}) - \left(\mathcal{R}(\mathbf{s}, a^{\dagger}) + \gamma[\max_{a\in\mathcal{A}} Q(\mathbf{s}', a)]\right)\right)^{2}\right]}_{\text{Standard Bellman Error}}$$

learns the Q-values for each fuzzy logic rule in the form of Eq. 8 where $\alpha > 0$ is a tradeoff factor s.t. as the size of available data grows, the magnitude of α can become lesser [34]. Further, $\hat{\pi}_{\beta}$ is the behavior policy that collected the training data D, $\gamma \in [0, 1]$ is the discount factor, $\mathbf{s}'$ is the next state *after* state $\mathbf{s}$ and $\mathcal{R}$ is the reward function. The functions $Q(\mathbf{s}, a)$, $Q(\mathbf{s}, a^{\dagger})$ and $Q(\mathbf{s}', a)$ are calculated by Eq. 9 with their respective arguments. Offline training continues until it reaches either convergence, a specified error threshold, or the maximum number of iterations.

3 CART POLE & RESULTS

For CLIP and ECM, the training data's order of presentation may affect the output, but subsequent runs do not appear to produce substantially different results.

Description: A cart is placed on a one-dimensional track with a pole affixed by a hinge. Each state in the environment is described by four continuous features: cart position, cart velocity, pole angle, and pole angular velocity. The goal is to balance the pole by moving the cart left or right; a reward of +1 is given for every time-step the pole remains balanced. An episode will end if either: (1) the pole angle exceeds ±12 deg; (2) the position of the cart is greater than ±2.4; or (3) the episode is longer than 500 time-steps (truncation).

Six approaches: For baselines, deep neural networks were trained with 4 different strategies: 1. ***Conservative Q-Learning (CQL)*** [34] with a Double Deep Q-Network [60]; 2. ***Batch Constrained Q-Learning (BCQ)*** [19]; 3. ***Neural Fitted Q Iteration (NFQ)*** [51]; 4. ***Behavior Cloning (BC)*** [57]. These baselines were selected due to their applicability in discrete control and capability/potential for offline learning. Additionally, a fifth baseline was trained, ***Fuzzy Q-Learning (FQL)*** [20], to demonstrate the necessity of a model-free fuzzy RL procedure for the offline setting (which we propose FCQL to answer this need). All five baselines are compared to our proposed ***Fuzzy Conservative Q-Learning (FCQL)***. The deep neural networks for CQL, BCQ, NFQ, and BC had two hidden layers, each with 256 neurons using ReLU activation function and bias; its output layer has a linear activation function and one output neuron per possible action. Both FQL and FCQL automatically design themselves when given the available training data as outlined in the "Method" section, and output is calculated by Eq. 9. However, FCQL will use the suggested augmentation from the CQL framework in Eq. 2.5, whereas FQL will only use the standard bellman error. For consistency, all methods used *Adam* [31].

Policy induction: Parameters shared by conditions (e.g., discount factor γ) were identical. The following values were used: $\alpha = 0.5$, $\gamma = 0.99$, learning rate $\eta = 1e - 4$ with a batch size of 64. For the FCQL, parameters for CLIP were $\epsilon = 0.2$, $\kappa = 0.6$, and ECM's distance threshold, $Dthr$, was 0.4. The cart pole training data and policy inductions for CQL, BCQ, NFQ as well as BC were from an offline Deep RL library called *d3rlpy* (MIT license) [53]. Two different methods were used to collect the cart pole training data: (1) *Replay* and (2) *Random*. The Replay data was collected by a DQN using experience replay and solving the cart pole environment *online*. In contrast, the Random data was collected by selecting actions randomly to balance the cart pole *online*.

Consequently, inducing a policy using only Random data increases the difficulty of inducing an effective policy. Each condition was run 50 times across different seeds; for every seed, each algorithm was shown the same data in the same order. When training on the Replay data —during each run —the amount of data available for *offline* training increased from 10 episodes to 250 episodes by increments of 10 to demonstrate how the conditions behave as more training data becomes available. When training on the Random data —during each run —the amount of data available for *offline* training increased from 100 episodes to 1000 episodes by increments of 100 instead as it became more challenging to balance the cart pole with a Random-only behavior policy. The model was evaluated *online* for 100 episodes using OpenAI Gym [14], and its average performance was recorded.

Discovery of Fuzzy Logic Rules: The fuzzy logic rule growth for either the FCQL or FQL conditions (as they both used the same self-organizing procedure and resulting neuro-fuzzy Q-network in their respective policy inductions) witnessed approximately linear growth as the amount of offline training data increased, as shown in Figures 2 and 3 when trained on the Replay and Random data, respectively. However, its growth does appear to plateau as even more data becomes available —this is likely due to the ECM algorithm serving as a threshold or bottleneck in preventing superfluous fuzzy logic rules from being generated.

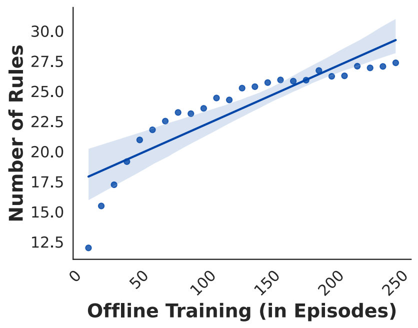


Figure 2: Replay data; the fuzzy logic rule growth for either the FCQL or FQL conditions.

Results on Replay data: Figure 4 shows that FCQL consistently outperforms CQL. Although BC offers comparable performance to FCQL as training episodes increase, FCQL maintains an early advantage by requiring fewer training episodes and containing

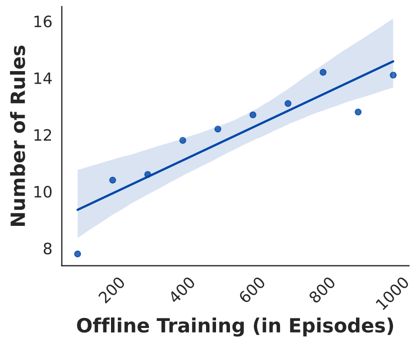


Figure 3: Random data; the fuzzy logic rule growth for either the FCQL or FQL conditions.

a rule-based knowledge base, whereas BC is implemented via a deep neural network. While BCQ is shown to be a robust offline RL algorithm in [18], it was unable to control the cart pole in the online evaluations in our experiments. One potential reason is that the original BCQ implementation for discrete action spaces uses a conditional Variational Autoencoder (VAE) to induce a policy, as it was initially proposed to solve the Atari games. In this work, the encoder and decoder had a latent space of 32 features. Given that the Cart Pole problem is a mere four features, the encoding to and from a space larger than the original state space may have resulted in BCQ being unable to learn the Q-values effectively.

Results on Random data: Figure 5 reveals that —with our self-organizing process —both the FCQL and FQL consistently outperform CQL, BCQ, NFQ, and BC. BC will try to mimic the behavior of the training data —which was collected by acting randomly —and thus, fails to learn a sufficient policy. CQL now struggles with balancing the cart pole, but since both FCQL and CQL use the same training objective, the only difference remains in their function approximation of the Q-values. FCQL and FQL benefit from self-organizing the neuro-fuzzy Q-network's architecture to adequately accommodate or reflect the patterns within the input space, which may have enhanced their approximation capabilities.

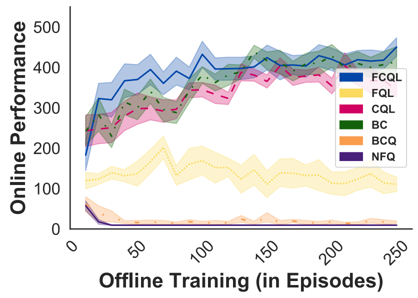


Figure 4: Replay data; starting from 10 episodes being available for offline training, up to 250 episodes of offline training, in increments of 10.

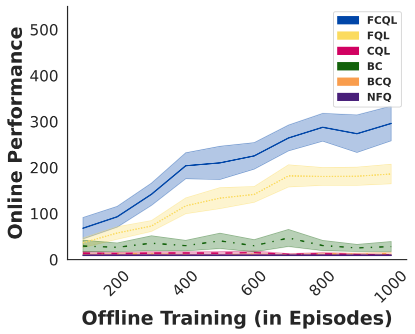


Figure 5: Random data; starting from 100 episodes being available for offline training, up to 1000 episodes of offline training, in increments of 100.

Human-in-the-loop: To demonstrate the feasibility of including humans in the design of the neuro-fuzzy Q-network, a suboptimal neuro-fuzzy Q-network partially trained by FCQL (with a mean of 381.7 and standard deviation of 101.1) is selected. Here we will show that the human expert can inspect and correct this suboptimal neuro-fuzzy Q-network. Figure 6 shows the identified linguistic terms for this example, and the fuzzy logic rule that had the strongest influence when the cart failed to balance the pole was $Rule_{21}$ shown in Figure 7 (i.e., "*if the cart is all the way to the left but is moving left slowly, and the pole is falling to the left quickly, then moving the cart to the left or right has Q-values of 15.90 and 15.72, respectively*"). In this original rule, the difference between Q-values for pushing the cart left (15.90) or right (15.72) is rather small even though pushing left is much better. So a human can modify the rule's Q-value for left from 15.90 to 17.0. The updated policy was then evaluated online for 100 episodes and consistently scored a perfect 500.0. The human was able to quickly adjust the suboptimal model to achieve optimal performance in ~5 minutes.

4 ITS EXPERIMENTS & RESULTS

Description: Figure 8 shows our ITS's graphical user interface. It is a web-based application that teaches ten probability principles (e.g., the Addition Theorem and Bayes' Theorem). The pedagogical decisions are whether the student should *solve* the next problem (***Problem-Solving (PS)***), study the tutor's solution as a *worked-out example* (***Worked-Example (WE)***) or *work collaboratively* with the tutor to solve the next problem (***Collaborative Problem-Solving (CPS)***). During the CPS, an additional level of interaction between the student and tutor occurs —the tutor can decide to either *tell* the student the next step or *elicit* the student to solve the next step themselves. The ITS provides adaptive instructions, immediate feedback, and on-demand hints to enhance learning. The student can request hints by clicking the [Hint] button shown in the "Response Window" of Figure 8.

Participants: This study was given to students as a homework assignment in an undergraduate Computer Science class in the Fall of 2021 at North Carolina State University. Students were told to complete the study in one week and will be graded based on demonstrated effort rather than learning performance. 129 students were *randomly assigned* into two conditions: FCQL ($N = 65$) and

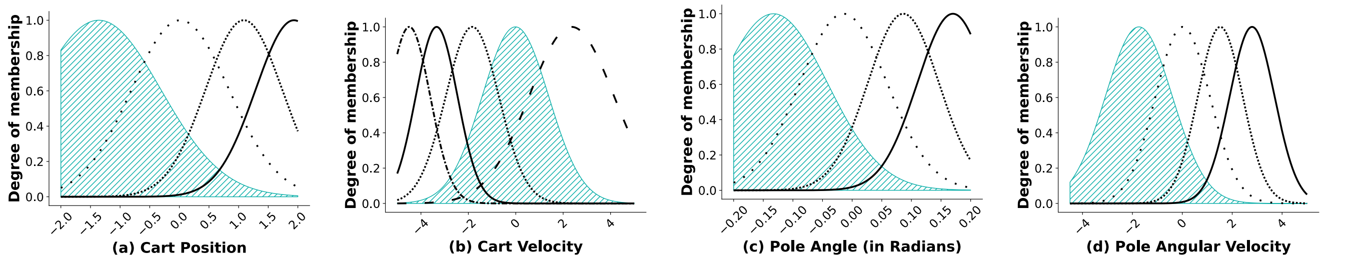


Figure 6: Linguistic terms describing the state environment as discovered by CLIP. Hatched and colored linguistic terms correspond to the selected $Rule_{21}$ in the "human-in-the-loop" example.

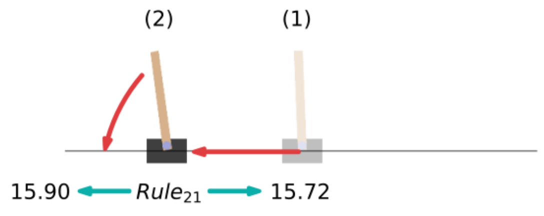
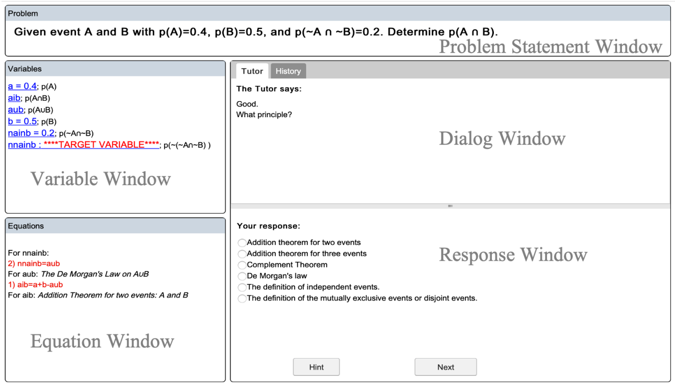


Figure 7: $Rule_{21}$ describes state (2). Here, (1) is the original state. In (2), the cart is left of the original position (Fig.6 (a)), the cart velocity is moving to the left slowly (Fig. 6 (b)), the pole angle is left from the vertical (negative) (Fig.6 (c)), and the pole angular velocity is also negative (Fig.6 (d)).

Expert ($N = 64$). Due to preparation for final exams and the study's length, 92 students completed the study, but two students were excluded from our analysis due to perfect performance in the pretest. The final group sizes were FCQL ($N = 45$) and Expert ($N = 47$). A Chi-square test revealed the students' completion rate between conditions was not significantly different: $\chi^2(2) = 0.8768$, $p = .349$.

Experiment procedure & grading: Consists of:

(1) **Textbook:** Students read about probability principles and review examples.
(2) **Pretest:** Students' a priori knowledge is bench-marked with 14 single- and multiple-principle problems.



Problem

Given event A and B with p(A)=0.4, p(B)=0.5, and p(~A ∩ ~B)=0.2. Determine p(A ∩ B).

Problem Statement Window

Variables

a = 0.4; p(A)
aib; p(A∩B)
aub; p(A∪B)
b = 0.5; p(B)
nainb = 0.2; p(~A∩~B)
nnainb : ****TARGET VARIABLE****; p(~(~A∩~B))

Variable Window

Equations

For nnainb:
2) nnainb=aub
For aub: *The De Morgan's Law on A∪B*
1) aib=a+b-aub
For aib: *Addition Theorem for two events: A and B*

Equation Window

The Tutor says:

Good.
What principle?

Dialog Window

Your response:

- Addition theorem for two events
- Addition theorem for three events
- Complement Theorem
- De Morgan's law
- The definition of independent events.
- The definition of the mutually exclusive events or disjoint events.

Response Window

Figure 8: The interface of our probability tutor.

(3) **ITS training:** Students receive training on 12 problems (shown in the same order for each student) with the assistance of an automated tutor (FCQL or Expert).
(4) **Posttest:** Students' learning is evaluated with 20 problems —14 isomorphic to the pretest, with the remaining 6 being non-isomorphic multiple-principle problems.[4]

All of the tests were graded in a double-blind manner by two experienced graders, and were normalized to $[0, 1]$.

Policy induction: The FCQL policy induction was done offline using pre-collected training data containing 1,834 students' interaction logs over nine semesters of classroom studies (Fall 2016 to Spring 2021). During these studies, the tutor, the general procedure, the training materials, and the training problems were all the same. The training corpus provides the state representation, action, and reward information for policy induction.

State: We extracted 142 features that might impact student learning from the student-system interaction logs that can be categorized into five groups: *Autonomy (10 features):* the amount of work done by the student, such as the number of elicits since the last tell; *Temporal Situation (29 features):* time-related information about work process, such as average time per step; *Problem-Solving (35 features):* information about the current problem-solving context, such as the difficulty of the current problem; *Performance (57 features):* information about the student's performance during problem-solving, such as the percentage of correct entries; *Hints (11 features):* student's hint usage, such as the total number of hints requested thus far.

Action: PS, WE, or CPS described above.

Reward: No immediate reward during tutoring, but the delayed reward is the students' Normalized Learning Gain (NLG), which measures their learning gain irrespective of their incoming competence [1–5, 27]. NLG is defined as $\frac{posttest-pretest}{\sqrt{1-pretest}}$, where 1 is the maximum score for both pre- and post-test. A hierarchy of FCQL policies were created such that one FCQL policy determines what action to take on the *problem-level*, and separate FCQL policies determine whether to *elicit* or *tell* the next step during a CPS. Parameters for the FCQL policy induction were: $\alpha = 0.1$, $\gamma = 0.99$, learning rate $\eta = 1e-2$, batch size of 128, $\epsilon = 0.3$, $\kappa = 0.7$, and $Dthr$ was 0.08.

Results: There was no significant difference between students' incoming competence, $t(90) = -0.490$, $p = 0.625$. The training time between conditions was also not significant, $t(90) = -1.282$,

[4]The posttest is designed to be significantly harder than the pretest.

$p = 0.203$. The FCQL violated normality, according to the Shapiro-Wilk test ($p = 0.003$), as well as the Expert ($p = 0.002$). Levene's Test for Equality of Variances found the two conditions had equal variance, ($p = 0.593$). The difference in posttest scores between FCQL (.803 ± .163) and Expert (.683 ± .165) was significant, $t(90) = 3.502$, $p = 0.001$, Cohen's $d = 0.730$. For completeness —due to the violation of normality —the Mann-Whitney U test was also consulted. Posttest scores between FCQL (mean rank = 56.60) and Expert (mean rank = 36.83) were statistically different, $U = 603$, $z = -3.553$ (standardized test statistic), $p \leq 0.0005$. By difference of means and comparison of mean ranks, FCQL statistically improved students' overall performance in the posttest problems compared to the Expert designed policy. Furthermore, students' incoming competence was also factored in by adjusting for their pretest scores with a one-way ANCOVA test. Between FCQL and Expert, there was a significant difference in posttest scores, $F(1, 89) = 22.520$, $p \leq 0.0005$, partial $\eta^2 = 0.202$. To summarize, our results showed that FCQL-induced policies are significantly more effective than the Expert designed ones in that FCQL performed significantly higher in their posttest than Expert despite both conditions solving the same problems in the same order and spending the same amount of time on the ITS.

5 RELATED WORK

Historically, fuzzy reinforcement learning (RL) was introduced to incorporate existing linguistic control knowledge to reduce the time required to solve RL [12]. ARIC was first proposed [11] but was later generalized to GARIC, which could learn despite weak reinforcement signals and allowed any differentiable membership function —unlike its predecessor [9]. These two methods were later evaluated in an Orbital Operations Simulator for Shuttle Attitude Control. However, they are dependent on exploration as they are on-policy actor-critic frameworks —limiting use to online RL [10].

An extension of Watkin's Q-Learning [66] with FLCs called Fuzzy Q-Learning was later proposed [20, 21] —like its predecessor —it is an off-policy method. Fuzzy Q-Learning was intended for online RL and works in domains with a continuous state space with continuous or discrete actions [26]. One important stipulation is that the aforementioned methods relied upon experts to provide the membership function definitions and fuzzy logic rules manually [9, 11, 20, 21, 26]. Initially, this was viewed as a benefit, but providing expert knowledge can be time-consuming, or there may be none available; thus, a self-organizing FLC by Q-learning was proposed —where the model automatically builds the fuzzy logic rules and learns their corresponding Q-values via *online* interaction by trial and error [30]. The dynamic evolving fuzzy neural network [28] —where the Evolving Clustering Method originates from —has also been used before for fuzzy RL, but again was for *online* [54] and cannot be interpreted since it is an *approximate fuzzy rule base* [15]. Other actor-critic frameworks such as RNN-FLCS [40], FACRLN [65], and RFALCON [38] could learn the structure and parameters of the FLC automatically but were designed for online fuzzy RL.

The first work to propose self-organizing FLCs to model-based offline RL is fuzzy particle swarm RL (FPSRL) —this was explicitly designed for when online learning is strictly forbidden, but requires a simulation to learn its necessary parameters (and subsequently its policy) [22]. Furthermore, FPSRL assumes that it is relatively easy to (1) model system dynamics given transition samples & (2) yield an interpretable control policy. These assumptions, and its simulation dependency, restrict FPSRL from settings where system dynamics cannot be easily modeled, or it is unclear if an interpretable control policy can easily be obtained —furthering the need for an offline model-free fuzzy RL algorithm that can be dynamic and evolve to environmental changes. To the best of our knowledge, there is no published work describing such a method.

The incorporation of RL with other logics has been explored. Relational RL (RRL) combines RL with inductive logic programming (or relational learning) to produce interpretable and generalizable policies; these can be applied to planning tasks such as the simple blocks world [17]. Relational Deep RL leveraged deep neural networks with RRL to achieve state-of-the-art performance in StarCraft II mini-games [70]; however, the authors' claim of interpretable policies is restricted to using *self-attention* —not (fuzzy) logic rules (i.e., natural language instructions) like in our proposed method. Alternatives such as neural logic RL have also been proposed, where induced policies are represented using first-order logic [25].

6 LIMITATIONS AND BROADER IMPACT

A self-organizing neuro-fuzzy Q-network has been proposed, and its effectiveness has been shown in Cart Pole as well as teaching students probability principles within an Intelligent Tutoring System (ITS). The resulting agent embodies a knowledge base consisting of interpretable fuzzy logic rules. In settings where exploration of options is prohibited, domain experts can validate the agent beforehand. Furthermore, the agent's decision making is transparent, as humans can examine which rules were applied, how strongly they influenced the output and then alter it accordingly.

Limitations: In this work, we did not evaluate the proposed methods on other datasets since this work is exploratory to verify the effectiveness of the FCQL framework; in the future, we will apply it to more datasets to further investigate its robustness and generalizability. A primary challenge for FCQL is balancing interpretability, and effectiveness [15]. A notable limitation of FCQL is its current inability to handle very high-dimensional tasks concerning knowledge base readability. Although it was demonstrated in the ITS, the resulting fuzzy logic rules contain 142 antecedents —hampering any effort by a human to read them. However, this may allow for interpretation by easily facilitating pattern mining (e.g., discovering rule activation patterns) compared to deep neural networks. This limitation on scalability is not reserved only to FCQL, but is a common theme amongst methods in fuzzy logic control; to address the scalability limitation, rough set theory [47] may be applied [16].

We believe neuro-fuzzy Q-networks can be a great alternative for offline RL where there is limited data [42], possible human expert knowledge to incorporate [9, 11, 24, 35], and the simultaneous demand for interpretation and accuracy [15]. Future research could improve upon the proposed FCQL procedure by extending the neuro-fuzzy Q-network to handle *computer vision tasks* such as the Atari games by incorporating object-sensitivity [37]. FCQL's current format is most applicable to problems with continuous inputs, from which a diagnosis must be obtained, in an environment where there is no available ground truth —only rewards or punishments.

ACKNOWLEDGMENTS

This research was supported by the NSF Grants: Integrated Data-driven Technologies for Individualized Instruction in STEM Learning Environments (1726550), CAREER: Improving Adaptive Decision Making in Interactive Learning Environments (1651909), and Generalizing Data-Driven Technologies to Improve Individualized STEM Instruction by Intelligent Tutors (2013502).

REFERENCES

[1] Mark Abdelshiheed, John Wesley Hostetter, Preya Shabrina, Tiffany Barnes, and Min Chi. 2022. The Power of Nudging: Exploring Three Interventions for Metacognitive Skills Instruction across Intelligent Tutoring Systems. In *Proceedings of the 44th annual conference of the cognitive science society*. 541–548.

[2] Mark Abdelshiheed, John Wesley Hostetter, Xi Yang, Tiffany Barnes, and Min Chi. 2022. Mixing Backward- with Forward-Chaining for Metacognitive Skill Acquisition and Transfer. In *Artificial Intelligence in Education*. Springer International Publishing, Cham, 546–552.

[3] Mark Abdelshiheed, Mehak Maniktala, Tiffany Barnes, and Min Chi. 2022. Assessing Competency Using Metacognition and Motivation: The Role of Time-Awareness in Preparation for Future Learning. In *Design Recommendations for Intelligent Tutoring Systems*. Vol. 9. 121–131.

[4] Mark Abdelshiheed, Mehak Maniktala, Song Ju, Ayush Jain, Tiffany Barnes, and Min Chi. 2021. Preparing Unprepared Students For Future Learning. In *Proceedings of the 43rd annual conference of the cognitive science society*. 2547–2553.

[5] Mark Abdelshiheed, Guojing Zhou, Mehak Maniktala, Tiffany Barnes, and Min Chi. 2020. Metacognition and Motivation: The Role of Time-Awareness in Preparation for Future Learning. In *Proceedings of the 42nd annual conference of the cognitive science society*. 945–951.

[6] Marcin Andrychowicz, Bowen Baker, et al. 2018. Learning dexterous in-hand manipulation. *arXiv:1808.00177* (2018).

[7] Kai Ang and Chai Quek. 2005. RSPOP: Rough Set-Based Pseudo Outer-Product Fuzzy Rule Identification Algorithm. *Neural computation* 17 (2005), 205–43.

[8] Plamen Angelov and Xiaowei Gu. 2017. Empirical Fuzzy Sets. *International Journal of Intelligent Systems* 33 (09 2017). https://doi.org/10.1002/int.21935

[9] H.R. Berenji and P. Khedkar. 1992. Learning and tuning fuzzy logic controllers through reinforcements. *IEEE Transactions on Neural Networks* 3, 5 (1992), 724–740.

[10] H.R. Berenji, R.N. Lea, Y. Jani, P. Khedkar, A. Malkani, and J. Hoblit. 1993. Space shuttle attitude control by reinforcement learning and fuzzy logic. In *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*. 1396–1401 vol.2. https://doi.org/10.1109/FUZZY.1993.327605

[11] Hamid R. Berenji. 1992. A reinforcement learning—based architecture for fuzzy logic control. *International Journal of Approximate Reasoning* 6, 2 (1992), 267–292. https://doi.org/10.1016/0888-613X(92)90020-Z

[12] Hamid R. Berenji and Sterling Software. 1991. Refinement of Approximate Reasoning-based Controllers by Reinforcement Learning. In *Machine Learning Proceedings 1991*, Lawrence A. Birnbaum and Gregg C. Collins (Eds.). Morgan Kaufmann, San Francisco (CA), 475–479.

[13] James C. Bezdek, Robert Ehrlich, and William Full. 1984. FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences* 10, 2 (1984), 191–203. https://doi.org/10.1016/0098-3004(84)90020-7

[14] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. https://doi.org/10.48550/ARXIV.1606.01540

[15] J. Casillas, O. Cordón, F.H. Triguero, and L. Magdalena. 2013. *Interpretability Issues in Fuzzy Modeling*. Springer Berlin Heidelberg. https://books.google.com/books?id=7r_qCAAAQBAJ

[16] Ron Tor Das et al. 2016. ieRSPOP: A novel incremental rough set-based pseudo outer-product with ensemble learning. *Applied Soft Computing* 46 (2016), 170–186.

[17] Sašo Džeroski, Luc De Raedt, and Kurt Driessens. 2001. Relational Reinforcement Learning. *Machine Learning* 43, 1 (01 Apr 2001), 7–52. https://doi.org/10.1023/A:1007694015589

[18] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. 2019. Benchmarking Batch Deep Reinforcement Learning Algorithms. https://doi.org/10.48550/ARXIV.1910.01708

[19] Scott Fujimoto, David Meger, and Doina Precup. 2018. Off-Policy Deep Reinforcement Learning without Exploration. https://doi.org/10.48550/ARXIV.1812.02900

[20] P.Y. Glorennec. 1994. Fuzzy Q-learning and dynamical fuzzy Q-learning. In *Proceedings of 1994 IEEE 3rd International Fuzzy Systems Conference*. 474–479 vol.1. https://doi.org/10.1109/FUZZY.1994.343739

[21] P.Y. Glorennec and L. Jouffe. 1997. Fuzzy Q-learning. In *Proceedings of 6th International Fuzzy Systems Conference*, Vol. 2. 659–662 vol.2. https://doi.org/10.1109/FUZZY.1997.622790

[22] Daniel Hein, Alexander Hentschel, Thomas Runkler, and Steffen Udluft. 2017. Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies. *Engineering Applications of Artificial Intelligence* 65 (2017), 87–98. https://doi.org/10.1016/j.engappai.2017.07.005

[23] John Wesley Hostetter. 2023. *johnHostetter/AAMAS-2023-FCQL: First release*. https://doi.org/10.5281/zenodo.7668308

[24] J.-S.R. Jang. 1993. ANFIS: adaptive-network-based fuzzy inference system. *IEEE Transactions on Systems, Man, and Cybernetics* 23, 3 (1993), 665–685.

[25] Zhengyao Jiang and Shan Luo. 2019. Neural Logic Reinforcement Learning. *CoRR* abs/1904.10729 (2019). arXiv:1904.10729 http://arxiv.org/abs/1904.10729

[26] L. Jouffe. 1998. Fuzzy inference system learning by reinforcement methods. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28, 3 (1998), 338–355. https://doi.org/10.1109/5326.704563

[27] Song Ju, Guojing Zhou, Mark Abdelshiheed, Tiffany Barnes, and Min Chi. 2021. Evaluating Critical Reinforcement Learning Framework in the Field. In *Artificial Intelligence in Education*, Ido Roll, Danielle McNamara, Sergey Sosnovsky, Rose Luckin, and Vania Dimitrova (Eds.). Springer International Publishing, Cham, 215–227.

[28] N.K. Kasabov and Qun Song. 2002. DENFIS: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *IEEE Transactions on Fuzzy Systems* 10, 2 (2002), 144–154. https://doi.org/10.1109/91.995117

[29] J. Kim and N. Kasabov. 1999. HyFIS: adaptive neuro-fuzzy inference systems and their application to nonlinear dynamical systems. *Neural Networks* 12, 9 (1999), 1301–1319. https://doi.org/10.1016/S0893-6080(99)00067-2

[30] Min-Soeng Kim, Sun-Gi Hong, and Ju-Jang Lee. 1999. Self-organizing fuzzy inference system by Q-learning. In *FUZZ-IEEE'99. 1999 IEEE International Fuzzy Systems. Conference Proceedings (Cat. No.99CH36315)*, Vol. 1. 372–377 vol.1. https://doi.org/10.1109/FUZZY.1999.793268

[31] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. https://doi.org/10.48550/ARXIV.1412.6980

[32] G. Klir and Bo Yuan. 1995. Fuzzy sets and fuzzy logic - theory and applications. Prentice-Hall Inc., Upper Saddle River, New Jersey.

[33] B. Kosko. 1994. Fuzzy systems as universal approximators. *IEEE Trans. Comput.* 43, 11 (1994), 1329–1333. https://doi.org/10.1109/12.324566

[34] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. 2020. Conservative Q-Learning for Offline Reinforcement Learning. https://doi.org/10.48550/ARXIV.2006.04779

[35] C.C. Lee. 1990. Fuzzy logic in control systems: fuzzy logic controller. I & II. *IEEE Transactions on Systems, Man, and Cybernetics* 20, 2 (1990), 404–435.

[36] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. https://doi.org/10.48550/ARXIV.2005.01643

[37] Yuezhang Li, Katia P. Sycara, and Rahul Radhakrishnan Iyer. 2017. Object-sensitive Deep Reinforcement Learning. *ArXiv* abs/1809.06064 (2017).

[38] Cheng-Jian Lin and Chin-Teng Lin. 1996. Reinforcement learning for an ART-based fuzzy adaptive learning control network. *IEEE Transactions on Neural Networks* 7, 3 (1996), 709–731. https://doi.org/10.1109/72.501728

[39] C.-T. Lin and C.S.G. Lee. 1991. Neural-network-based fuzzy logic control and decision system. *IEEE Trans. Comput.* 40, 12 (1991), 1320–1336. https://doi.org/10.1109/12.106218

[40] Chin-Teng Lin and C.S.G. Lee. 1994. Reinforcement structure/parameter learning for neural-network-based fuzzy logic control systems. *IEEE Transactions on Fuzzy Systems* 2, 1 (1994), 46–63. https://doi.org/10.1109/91.273126

[41] P. Lindskog. 1997. *Fuzzy Identification from a Grey Box Modeling Point of View*. Springer Berlin Heidelberg, Berlin, Heidelberg, 3–50. https://doi.org/10.1007/978-3-642-60767-7_1

[42] H.H. Lou and Y.L. Huang. 2000. Fuzzy-logic-based process modeling using limited experimental data. *Engineering Applications of Artificial Intelligence* 13, 2 (2000), 121–135. https://doi.org/10.1016/S0952-1976(99)00057-3

[43] Jean M. Mandler. 2008. On the Birth and Growth of Concepts. *Philosophical Psychology* 21 (2008), 207 – 230.

[44] Jean M Mandler, Patricia J Bauer, and Laraine McDonough. 1991. Separating the sheep from the goats: Differentiating global categories. *Cognitive Psychology* 23, 2 (1991), 263–298. https://doi.org/10.1016/0010-0285(91)90011-C

[45] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (01 Feb 2015), 529–533. https://doi.org/10.1038/nature14236

[46] Besa Muslimi, Miriam A. M. Capretz, and Jagath Samarabandu. 2008. An Efficient Technique for Extracting Fuzzy Rules from Neural Networks. *International Journal of Electrical and Computer Engineering* 2, 4 (2008), 1231 – 1237.

[47] Zdzislaw Pawlak. 1998. Rough Set Theory and its Applications to Data Analysis. *Cybernetics and Systems* 29, 7 (1998), 661–688. https://doi.org/10.1080/019697298125470 arXiv:https://doi.org/10.1080/019697298125470

[48] Agus Priyono, Muhammad Ridwan, Ahmad Alias, Riza Rahmat, Azmi Hassan, and Mohd Mohd Ali. 2005. Generation of Fuzzy Rules with Subtractive Clustering. *Jurnal Teknologi* 43 (02 2005), 143. https://doi.org/10.11113/jt.v43.782

[49] Rafael Figueiredo Prudencio, Marcos R. O. A. Maximo, and Esther Luna Colombini. 2022. A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems. https://doi.org/10.48550/ARXIV.2203.01387

[50] C. Quek and R.W. Zhou. 1999. POPFNN-AAR(S): a pseudo outer-product based fuzzy neural network. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 29, 6 (1999), 859–870.

[51] Martin Riedmiller. 2005. Neural Fitted Q Iteration – First Experiences with a Data Efficient Neural Reinforcement Learning Method. In *Machine Learning: ECML 2005*, João Gama, Rui Camacho, Pavel B. Brazdil, Alípio Mário Jorge, and Luís Torgo (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 317–328.

[52] Rowena Rodrigues. 2020. Legal and human rights issues of AI: Gaps, challenges and vulnerabilities. *Journal of Responsible Technology* 4 (2020), 100005. https://doi.org/10.1016/j.jrt.2020.100005

[53] Takuma Seno and Michita Imai. 2021. d3rlpy: An Offline Deep Reinforcement Learning Library. https://doi.org/10.48550/ARXIV.2111.03788

[54] Hitesh Shah and M. Gopal. 2014. A Reinforcement Learning Algorithm with Evolving Fuzzy Neural Networks. *IFAC Proceedings Volumes* 47, 1 (2014), 1161–1165. https://doi.org/10.3182/20140313-3-IN-3024.00058 3rd International Conference on Advances in Control and Optimization of Dynamical Systems (2014).

[55] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, et al. 2016. Mastering the game of go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[56] Richard S. Sutton and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction*. A Bradford Book, Cambridge, MA, USA.

[57] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral Cloning from Observation. https://doi.org/10.48550/ARXIV.1805.01954

[58] Sau Wai Tung, Chai Quek, and Cuntai Guan. 2011. SaFIN: A Self-Adaptive Fuzzy Inference Network. *IEEE Transactions on Neural Networks* 22, 12 (2011), 1928–1940.

[59] W. L. Tung and C. Quek. 2002. DIC: A Novel Discrete Incremental Clustering Technique for the Derivation of Fuzzy Membership Functions. In *PRICAI 2002: Trends in Artificial Intelligence*, Mitsuru Ishizuka and Abdul Sattar (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 178–187.

[60] Hado van Hasselt, Arthur Guez, and David Silver. 2015. Deep Reinforcement Learning with Double Q-learning. https://doi.org/10.48550/ARXIV.1509.06461

[61] L.-X. Wang. 1992. Fuzzy systems are universal approximators. In *[1992 Proceedings] IEEE International Conference on Fuzzy Systems*. 1163–1170. https://doi.org/10.1109/FUZZY.1992.258721

[62] Li-Xin Wang. 1997. *A Course in Fuzzy Systems and Control*.

[63] L.-X. Wang and J.M. Mendel. 1992. Fuzzy basis functions, universal approximation, and orthogonal least-squares learning. *IEEE Transactions on Neural Networks* 3, 5 (1992), 807–814. https://doi.org/10.1109/72.159070

[64] L.-X. Wang and J.M. Mendel. 1992. Generating fuzzy rules by learning from examples. *IEEE Transactions on Systems, Man, and Cybernetics* 22, 6 (1992), 1414–1427. https://doi.org/10.1109/21.199466

[65] Xue-Song Wang, Yu-Hu Cheng, and Jian-Qiang Yi. 2007. A fuzzy Actor–Critic reinforcement learning network. *Information Sciences* 177, 18 (2007), 3764–3781. https://doi.org/10.1016/j.ins.2007.03.012

[66] Christopher J. C. H. Watkins and Peter Dayan. 1992. Q-learning. *Machine Learning* 8, 3 (01 May 1992), 279–292. https://doi.org/10.1007/BF00992698

[67] Ronald R. Yager and Dimitar P. Filev. 1994. Generation of Fuzzy Rules by Mountain Clustering. *J. Intell. Fuzzy Syst.* 2, 3 (may 1994), 209–219.

[68] L.A. Zadeh. 1965. Fuzzy sets. *Information and Control* 8, 3 (1965), 338–353.

[69] Lotfi Zadeh and Rafik Aliev. 2018. *Fuzzy Logic Theory and Applications: Part I and II*.

[70] Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter Battaglia. 2018. Relational Deep Reinforcement Learning. https://doi.org/10.48550/ARXIV.1806.01830