Research Article

AI-Assisted Decision-Making and Risk Evaluation in Uncertain Environment Using Stochastic Inverse Reinforcement Learning: American Football as a Case Study

Risa Takayanagi,1 Keita Takahashi,2 and Tomah Sogabe1,2,3

1Engineering Department, The University of Electro-Communications, Tokyo, Japan
2Grid, Inc., Tokyo, Japan
3i-PERC, The University of Electro-Communications, Tokyo, Japan

Correspondence should be addressed to Tomah Sogabe; sogabe@uec.ac.jp

Received 17 August 2022; Accepted 26 September 2022; Published 10 October 2022

Academic Editor: Anna M. Gil-Lafuente

Copyright © 2022 Risa Takayanagi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In this work, we focus on the development of an AI technology to support decision making for people in leadership positions while facing uncertain environments. We demonstrate an efficient approach based on a stochastic inverse reinforcement learning (IRL) algorithm constructed by hybridizing the conventional Max-entropy IRL and mixture density network (MDN) for the prediction of transition probability. We took the case study of American football, a sports game with stochastic environment, since the number of yards gainable on the next offence in real American football is usually uncertain during strategy planning and decision making. The expert data for IRL are built using the American football 2017 season data in National Football League (NFL). The American football simulation environment was built by training MDN using the annual NFL data to generate the state transition probability for IRL. Under the framework of Max-Entropy IRL, optimal strategy was successfully obtained through a learnt reward function by trial-and-error communication with the MDN environment. To precisely evaluate the validity of the learnt policy, we have conducted a risk-return analysis and revealed that the trained IRL agent showed higher return and lower risk than the expert data, indicating that it is possible for the proposed IRL algorithm to learn superior policy than the one derived directly from the expert teaching data. Decision-making in an uncertain environment is a general issue, ranging from business operation to management. Our work presented here will likely serve as a general framework for optimal business operation and risk management and contribute especially to the portfolio’s optimization in the financial and energy trading market.

1. Introduction

Society has many uncertain factors, such as the global environment, human psychology, economic and social trends, and so on. In recent years, it has been called “VUCA world” in business, an acronym for volatility, uncertainty, complexity, and ambiguity (VUCA) [1], which refers to a complex society that is difficult to predict. Though facing these difficulties, we humans have to make many decisions every day, ranging from decisions made in our daily lives, such as daily purchasing and schedule management, to decisions made by leaders that may affect the future fate of the organization. Leaders, especially those who make decisions as the chief executives of organizations, have a great deal of responsibility and struggle with loneliness and suffering as they make decisions [2]. They also need to be able to adapt quickly to unpredictable environments and challenges that they have never experienced before. The purpose of this work is to develop an AI technology to support decision making in uncertain environments, especially for people in leadership positions. Our goal is that AI, which excels in feature extraction, prediction, and optimization, will fully demonstrate its strengths and become a close companion to human decision-making in situations where prediction is difficult.
We treat sports as an example of an uncertain environment. In any sport, the outcome can vary greatly depending on the psychological conditions of the players, the weather, and the venue of the game that day. American football is a set play system in which each play is made from a stationary position, in contrast to soccer, where the game flows continuously for about one and half an hour, except the 15 minutes of break time in the middle of game. Each play in American football is directed by a coach outside the play field, and the progress and the outcome of the game is greatly affected by the coach’s intentions because of the very short play time of each down. Therefore, it is extremely important for the coaches to make a wise and promising strategy based on their own domain knowledge and past coaching experience before the game starts. In American football community, play tactics are usually highly confidential, and data analysis methods are not widely disclosed. However, in recent years, with the advancement of information processing and image processing technologies, advanced data acquisition and analysis have been introduced in the field of sports. Gaining access to detailed and accurate data has become easier in the recent years because of the increasing popularity of open source. Here, we propose a method aiming at the development of an AI-assisted decision-making platform to support the coach during play selection using stochastic inverse reinforcement learning (IRL) and to suggest the plays that the coach should select when facing an uncertain game situation.

Prior to present a general picture about IRL, we, firstly, give a brief about the reinforcement learning (RL) algorithm, a counterpart of inverse reinforcement learning. RL technics were used to evaluate decision-making and player behavior. RL technics have been used in the field of control theory to tackle the nonlinear system or robust control problem from low-dimensional ones, such as cartpole and mountain carto high-dimensional, and naturally sparse environments based on OpenAI Gym [3–9]. In RL, the agent learns by exchanging information about “state,” “action,” and “reward function” between the environment and agent. One of the challenging problems in the conventional RL approach is that the reward function must be designed in advance by the experience or domain knowledge of humans. If the reward function is not designed properly, the agent might take unintended actions. Thus, even if the algorithm is accurate, effective, and reliable, a reward function is necessary for successful learning. In particular, in a realistic sports game, the reward function for value approximation is hardly defined precisely [10]. The IRL algorithm is proposed to tackle this problem [11, 12]. In IRL, the reward function is derived from the expert data, meaning that there is no need for manually design the reward functions. Instead, there is a need to prepare the training data set accumulated from the realistic games.

Most of RL and IRL works mentioned above were usually conducted inside a stable environment. The focus of our work is the policy-learning problem involving an uncertain environment, which remains difficulty and challenging in both RL and IRL fields. Although some research efforts have been devoted along this line, most of results are still in pieces and incomplete [13, 14]. In this work, we choose the most widely used Max-Entropy IRL algorithm as the base tool to tackle decision making under uncertain American football environment [12]. For constructing the virtual American football environment, we used the mixture density network (MDN) to update the current state by following the learnt state transition probability [15]. MDN is an algorithm for learning models with uncertainty, and it learns the probability distribution from the feature of the input data [15]. In the game environment of American football, even if the coach selects the same action at the same location, the next transition state (the number of yards gainable) is probabilistic. We used professional play-by-play data in NFL 2017 season as training data for MDN. The dataset contains 9516 plays of 21 games by 32 teams from Big Data Bowl (https://www.pro-football-reference.com). The information includes the detailed play choices each professional team have made under each situation. The main contributions of this article are summarized as follows:

1. An efficient stochastic IRL algorithm under an uncertain environment was constructed by integrating the conventional max-entropy IRL algorithm with the mixture density network (MDN) while taking the sports game American football as an example.

2. Optimal strategy was successfully obtained through a learnt reward function via the visiting frequency by trial-and-error communication with the MDN environment built upon the expert teaching data.

3. Risk-return analysis was performed on the learnt policy. It is found that the learnt policy possesses a higher return and lower risk than the expert data, implying that the IRL algorithm is able to learn superior policy than the one derived directly from the expert teaching data.

The remainder of this paper is organized as follows: we will first describe the rules of American Football studied in the current work in Section 2. We also give a detailed description regarding the IRL and MDN algorithms, as well as the reference DP algorithm, in Section 2. Then, we will present the numerical experiment results by testing the IRL model in Sections 3 and 4. It is followed by a discussion on risk evaluation and policy verification in Section 5. Related works are presented in Section 6, and conclusions are presented in Section 7.

2. Model and Algorithms

2.1. Rules of American Football. American Football is one of the most popular sports in USA. Opposed to Soccer or Basketball, in which the game flows continuously, American Football is independent for each play. Between each play, coach instructs 11 players on the field on how and what play would proceed for the next play. The decisions made during this interval play a vital role and have a direct effect on final scores.
The offense team has the opportunity to play 4 downs (4 plays) in each sequence, in which they must advance 10 yards in each of these downs. Generally, the 4th down is used to recover their own position. Therefore, the offense team usually tries to advance 10 yards within 3 downs as much as possible. If the offense team advances 10 yards, they get a new opportunity to continue another sequence. The final target is to advance the ball beyond the goal line at the end of the field. Figure 1(a) shows several examples of how the offense team advances. It shows that the offense team is successful in advancing more than 10 yards, thus gaining a reward of "1" in example 1. In contrast, in examples 2 and 3, the offense team ends up being short of 10 yards, thus getting a reward of "0" in both cases. The difference between examples 2 and 3 is that the offense team in example 2 uses the next play to recover its position, and the opponent team will go on an offense, while in example 3, the defense team steals the ball and goes an offense in the next play. Thus, each play in American Football has different features and strategies for each team, which is determined by the movement of 11 players. In fact, plays can be categorized according to their features. Representative categories mainly include pass play or run play. Pass play is a play where the ball is passed around from player to player. It has a low probability of success and high risk of turnover by the opponent team but has a greater chance to advance the ball with longer distances. In contrast, run play has a low risk of turnover but is difficult to advance long distances. An example about pass play and run play is given in Figure 1(b). Here, guard, tackle, and end belong to run play, while short and deep belong to pass play. The decision strategies involved in American football is primarily focused on how to select and balance these two types of plays.

2.2. Markov Model for American Football. The Markov decision process (MDP) [15] models an environment in which state transitions occur stochastically. MDP is used as a mathematical framework in modeling decision making when dynamic programming (DP) or RL are applied. Figure 2 illustrates the MDP framework for the virtual American football environment. Here, MDP is viewed as transition from state $s_t \in \mathcal{S}$ to state $s_{t+1} \in \mathcal{S}$ upon the action $a \in \mathcal{A}$ taken in the state $s_t \in \mathcal{S}$, which depends only on one previous state $s_t$ and action $a$, and not on any previous state $s_{t-1}$ or action. $\mathcal{S} = \{s_1, s_2, \ldots, s_9\}$ is a set of states called the state space, and $\mathcal{A} = \{a_1, a_2, \ldots, a_9\}$ is a set of actions the agent can take, which is called the action space. This sequence of variables evolves over a Markov process (MP). In addition, MDP can also be described by a 4-tuple $S, A, T, R$, where $T$ represents the transition probability $T(s_{t+1}|s_t, a)$ of moving from state $s_t$ to state $s_{t+1}$ when action $a$ is taken, and $R$ is the reward function received by the agent from the environment. In the conventional RL framework, using the simple rule shown in Figure 1, a simple reward function can be designed that if the offense team is successful in advancing more than 10 yards, a reward of "1" is gained, otherwise, the reward is "0." However, as well-known, a simple and manually designed reward function is usually guaranteed for efficient learning. In the current work, we adopt the IRL algorithm in which the reward function is learnt through expert demonstration data. In this case, the reward $R$ is not a variable returned from the environment as in the usual RL environment. Instead, it would be learnt from the expert data. More details on learning reward will be given in the next section. Usually, the optimal behavior in an MDP is the behavior (policy) that maximizes the cumulative value of discounted rewards in the future from the current time.

In Figure 2, we describe how to map the American Football game into the stochastic MDP-IRL scheme. MDP is defined by a set of states, $\mathcal{S}$, a set of actions, $\mathcal{A}$, and transition functions $T$: $S \times A \rightarrow \mathcal{PD}(S)$, where $\mathcal{PD}$ represents the set of probability distributions over finite set of state $\mathcal{S}$. For an agent, there is the reward function $R$: $S \times A \rightarrow R$. In this work, we define the states based on the field of American football as consisting of down and distance. There are three downs for each set play. The distance is featured by the field distance left to reach the goal, and the field distance is always discretized as follows: $1, 2, 3, 4, \ldots, 10$ yards. Therefore, the total state combinations are the direct product of $[1\text{st} \text{down}, 2\text{nd} \text{down}, \text{and } 3\text{rd} \text{down}] \times [10, 9, 8, \ldots, 1, 0]$, and the number is 33. The detailed information regarding the definition of the state used in the IRL algorithm is also presented in the table at the left side of Figure 2 for better understanding. As for the action space, we consider five kinds of action in this work, namely guard, tackle, end, short, and deep, which are categorized according to the features of plays shown in Figure 1. Run plays are guard, tackle, and end, while pass plays are short and deep. The reward function is learnt through the IRL algorithm itself and will be explained in a more detailed manner in the subsequent section.

2.3. Inverse Reinforcement Learning. As mentioned before, one of the challenges in RL is the design of a reward function. In RL, humans must make a decision on how to act based on either the acquired policy or the state action value function. It is a typical MDP process, and if the reward function in particular is not set appropriately, the agent will fail to take optimal actions. IRL, which requires no human intervention in setting the reward, is proposed to tackle this problem. In the following, we will provide a detailed description regarding the Max-Ent IRL algorithm [12, 16] we adopted in the current work.

Max-Ent IRL is an algorithm of estimating a probability distribution from an expert’s trajectory history so that the entropy is maximized under certain constraints. Entropy represents the uncertainty of an information. The larger the value, the higher the uncertainty of the information. It is because behavioral history data may contain unobserved events, and entropy maximization favors to uniformly assign probabilities to unobserved events. The entropy of this situation is defined as follows:

$$H(\zeta|\theta) = \sum_{\zeta} \log(P(\zeta|\theta)).$$  \hspace{1cm} (1)
Here, the \( i \)-th path trajectory \( \xi_i = (s_1, a_1, s_2, \ldots, a_{T-1}, s_T) \) is a sequence of states and actions. \( \theta \) is the reward function parameter. \( P(\xi|\theta) \) is the path distribution. In this algorithm, the reward function is approximated using the polynomial parameter \( \theta \) as follows:

\[
R(\xi_i|\theta) = \sum_{i=1}^{T} \theta^{T} \phi(s_i) = \theta^{T} \sum_{i=1}^{T} f(s_i) = \theta^{T} f_{\xi_i},
\]

where \( \phi(s_i) \in \mathbb{R}^{[d]} \) represents the feature of state \( s \). The reward function \( R \) is interpreted as the agent’s utility for visiting that state. Formally, \( f: S \rightarrow \mathbb{R}^{d} \) is a mapping from the state space to a \( d \)-dimensional feature space, or the visiting frequency. In most of the cases, \( f(s_i) \) can be simply treated as the record of the visiting history of expert trajectory and is represented as one-hot vector, i.e., if the state is visited, then \( f(s_i) \equiv 1 \). Otherwise, \( f(s_i) \equiv 0 \). To estimate the parameters of the reward function, the goal is to make the feature vector of trajectory by policy \( \pi^* \), obtained from the estimated reward function \( R \), as close as possible to the expert’s feature vector. The entropy maximization algorithm is designed to fulfill this purpose. Find the optimal parameter \( \theta^* \) for acquiring optimal policy \( \pi^* \) using the following:

\[
\theta^* = \underset{\theta}{\text{argmax}} \ H(\xi|\theta) = \underset{\theta}{\text{argmax}} \sum_{\xi} \log(P(\xi|\theta)).
\]

Here, \( P(\xi|\theta) \) is defined as follows:

\[
P(\xi|\theta) = \frac{\exp(R_\theta(\xi_i))}{Z},
\]

where \( Z(\theta) \) is the partition function and normalization value. The gradient of this entropy function represents the difference between expected expert feature counts and the agent’s expected feature counts and is shown as below. The detailed derivation regarding the formula (17) can be found in the equations (5)–(16).
Mathematical Problems in Engineering

\[ \theta^* = \arg \max_{\theta} \sum_{i} \log(P(\xi_0 | \theta)) \longrightarrow \arg \max_{\theta} \frac{1}{M} \sum_{i=1}^{M} \log P(\xi_i | \theta), \]  
(5)

\[ \theta' = \arg \max_{\theta} \left\{ \frac{1}{M} \sum_{i=1}^{M} \log \frac{Z}{\exp[R(\xi_i)]} \right\}. \]  
(6)

Here, \( Z \) is defined as follows:

\[ Z = \sum_{\xi \in Z} \exp[R(\xi)], \]  
(7)

\[ \theta^* = \arg \max_{\theta} \left\{ \frac{1}{M} \sum_{i=1}^{M} R(\xi_i) - \log Z \right\}. \]  
(8)

Using (9),

\[ \theta^* = \arg \max_{\theta} \left\{ \frac{1}{M} \sum_{i=1}^{M} R(\xi_i) - \log \sum_{i=1}^{M} \exp[R(\xi_i)] \right\}. \]  
(9)

\[ L(\theta) = \frac{1}{M} \sum_{i=1}^{M} R(\xi_i) - \log \sum_{i=1}^{M} \exp[R(\xi_i)], \]  
(10)

\[ R(\xi_i) = \theta^T \cdot \phi(\xi_i), \]  
(11)

\[ \nabla L(\theta) = \frac{d}{d\theta} \left\{ \frac{1}{M} \sum_{i=1}^{M} \theta^T \phi(\xi_i) - \log \sum_{i=1}^{M} \exp[\theta^T \phi(\xi_i)] \right\}, \]  
(12)

\[ \nabla L(\theta) = \frac{1}{M} \sum_{i=1}^{M} \phi(\xi_i) \]  
(13)

\[ -\sum_{i=1}^{M} \frac{1}{\exp[\theta^T \phi(\xi_i)]} \frac{d}{d\theta} \sum_{i=1}^{M} \exp[\theta^T \phi(\xi_i)], \]

\[ \nabla L(\theta) = \frac{1}{M} \sum_{i=1}^{M} \phi(\xi_i) \]  
(14)

\[ -\sum_{i=1}^{M} \frac{1}{\exp[\theta^T \phi(\xi_i)]} \sum_{i=1}^{M} \exp[\theta^T \phi(\xi_i)] \phi(\xi_i), \]

\[ \nabla L(\theta) = \frac{1}{M} \sum_{i=1}^{M} \phi(\xi_i) \]  
(15)

\[ -\sum_{i=1}^{M} \frac{1}{\exp[\theta^T \phi(\xi_i)]} \sum_{i=1}^{M} \exp[\theta^T \phi(\xi_i)] \phi(\xi_i), \]

\[ \nabla L(\theta) = \frac{1}{M} \sum_{i=1}^{M} \phi(\xi_i) - \sum_{i=1}^{M} P(\xi_i | \theta) \phi(\xi_i), \]  
(16)

\[ \nabla L(\theta) = \frac{1}{M} \sum_{i=1}^{M} \phi(\xi_i) - \sum_{i=1}^{M} P(\xi_i | \theta) \phi(\xi_i). \]  
(17)

Here, the first term represents the visiting frequency derived from the expert trajectories, and the second term represents the visiting frequency of the best learning agent. Using a dynamic programming framework, the optimal policy \( \pi^* \) can be defined as follows:

\[ \pi^* (a_i | s_i, \theta) = \arg \max_{a} Q(s_i, a_i), \]  
(18)

where the state action value function \( Q(s, a) \) is recurrently simulated by the Bellman equation as follows:

\[ Q(s_i, a_i) = R(s_i, \theta) + \gamma \left( \sum_{s_{i+1}} T(s_{i+1} | a_i, s_i) V^\pi(s_{i+1}) \right). \]  
(19)

Using the acquired optimal policy \( \pi^* (a_i | s_i, \theta) \), the visiting frequency of the best learning agent \( \mu_0 (s_i, \theta) \) at time step \( t \) is also recurrently updated based on the following formula. In practice, the initial value of \( \mu_0 (s_i, \theta) \) can be set as any arbitrary random number.

\[ \mu_t (s_i, \theta) = \sum_{a \in A_{s_i}} \sum_{x \in S} \mu_{t-1} (s_i) \pi(a | s_i, \theta) T(s_{i+1} | a_i, s_i). \]  
(20)

Here, \( T(s_{i+1} | s_i, a_i) \) is the stochastic transition probability and will be given in more detail in Section 2.4. The integrated final visiting frequency of specified state \( s_i \) in \( P(\xi_i | \theta) \) under a given trajectory length \( T \) for all \( s_i \) is given as follows and will be used in formula (17) to update the learning parameter \( \theta \).

\[ P(s_i | \theta) = \sum_{t=1}^{T} \mu_t (s_i, \theta). \]  
(21)

2.4. Mixture Density Network. In this work, we use the mixture density network (MDN) [15] to learn stochastic state transition models. MDN has the merits of providing a probability distribution over a range of outputs for a given input. The technics to learn multiple Gaussian distribution are given below. Given the input vector, the probability \( p(y|x) \) can be approximated as follows:

\[ p(y|x) = \sum_{k=1}^{K} \pi_k (x) N(y | \mu_k (x), \sigma_k^2 (x)), \]  
(22)

\[ \sum_{k=1}^{K} \pi_k (x) = 1. \]  
(23)

Here, \( k \) is the index of the corresponding mixture component, ranging up to \( K \). The parameter \( \pi_k (x) \) denotes the mixing coefficient of \( k^{th} \) Gaussian distribution. \( N(\cdot) \) means \( k^{th} \) Gaussian distribution, and it has an explicit form, which is as follows:

\[ N(y | \mu_k (x), \sigma_k^2 (x)) = \frac{1}{\sqrt{2\pi\sigma_k^2 (x)}} \exp \left( -\frac{(\mu_k (x) - y)^2}{2\sigma_k^2 (x)} \right). \]  
(24)

Here, \( y \) is the expert training data, \( \mu_k (x) \) represents the mean of the \( k^{th} \) kernel, and the \( k^{th} \) variance is described by \( \sigma_k^2 (x) \). For independent training data, the error function takes the subsequent form, and \( w \) is the weight of MDN.
To minimize the error function, we need to calculate the derivatives of the error $E(w)$ with respect to the learning weight $w$.

Next, we explain how to project the above MDN algorithm to the current virtual American football environment for predicting the state transition probability. Figure 3 shows the details along with one example of using MDN to generate the transition matrix $T(s_{t+1}|a, s_t)$. Probability $p(y|x)$ is redefined as $T(s_{t+1}|a, s_t)$, with the corresponding relation that input $x$ is converted into three neurons assigned to two states and one action of $(s, a)$ and output $[\mu_k, \sigma_k^2]$, with up to the $k^{th}$ Gaussian distribution (here we use $k = 30$) is added and is used to minimize the error under training data $y$, which corresponds to $s_{t+1}$ extracted from the expert database. As for the state space and action space, details have been given in the previous section. Note that the state includes two elements: distance and down. The transition of the down is automatically implemented by following the sequence $[1^{st} \text{ down} \rightarrow 2^{nd} \text{ down} \rightarrow 3^{rd} \text{ down}]$. Thus, it is not included in the output state $s_{t+1}$ (note that the down information is necessary to be specified as one of three input variables for efficiently training MDN). Figure 3 also shows the learnt distribution along with the target distribution, and the large overlapping between these two indicates that trained MDN is able to catch the main feature of expert-training data. For better understanding, we also took one example to illustrate how to utilize the trained MDN model. Assume that the play is currently at a play position of “$1^{st}$ down,” with the distance to goal as index “$2$” (current position is at the yard of index “$8$”). The player takes the action “guard,” which corresponds to “action=$0$.” They form the three-input information. Since the player is currently in $1^{st}$ down, the state transition will automatically be updated to the $2^{nd}$ down while leaving the yard to be gained uncertain. To obtain the state information regarding the yard to be gained, the three-input information is fed into MDN with trained weight $w$. Perform the sampling of the possible advancing yards from the probabilistic distribution function $\mathcal{N}(y|\mu_k(x), \sigma_k^2(x))$. For a reliable histogram distribution for all yards to be gained, 1000 times of sampling is implemented. The table embedded in Figure 3 shows the resultant probability for each possible yard. There is 66% probability of failure to proceed to the next state, which means staying at the same yard “$8$.” There is 9.5% probability to proceed to yard “$9$” and 24% to proceed to yard “$10$.”

3. Stochastic IRL for American Football

A general view of the implementation flow chart of the Max-Ent IRL algorithm in a virtual American football environment is given below. We apply MDN to estimate the probability distribution of the next state, given the input of current state and action. MDN predicts the probability distribution of the distance to be gained in each situation. The probability values are given for each predicted gainable distance, and we integer-discretize them from 0 to 10. After 1000 times of sampling, we obtain a histogram for all possible yards. The probability derived from the histogram is then used in the IRL algorithm. In IRL, the value from MDN is used to replace the transition probability $T(s_{t+1}|a, s_t)$ in formula (20) to simulate the visiting frequency of the best learning agent. A graphical illustration of the flow chart for Stochastic Max-Ent IRL is given as below (Figure 4).

In this work, we designed two simulation conditions in this work for accommodating to the actual game, which requires a quick response of the output policy from the trained IRL agent. We separated MDN sampling-based IRL in two ways. One is the so-called one-shot MDN-IRL, which takes only one time sampling and outputs the predicted policy from the learnt IRL agent under one-shot MDN. Another approach was dubbed as ensemble MDN-IRL, which was designed to balance the sampling bias. In ensemble MDN-IRL, the one-shot MDN-IRL was repeated fifty times, and the final policy at each state $s_t$ was determined by the majority vote, i.e., the dominating action among fifty trials of IRL.

In this work, to verify the validity of the Max-Ent IRL algorithm, we also developed a conventional dynamic programming (DP) optimization algorithm acting as a reference algorithm for comparison. DP is implemented as follows: we, firstly, calculated the action $a_*(s_t)$ under policy $\pi$. $a_*(s_t)$ is an action that maximizes the probability of reaching the target terminal state $s_f$ from each state $s_t$. We defined $V_\pi(s_t)$ as the state value function by considering the probability that the offense team can reach $s_f$ from a state $s_t$ by the transition according to policy $\pi$, and $V_\pi(s_f) = 1$ at the terminate state. $V_\pi(s_t)$ was simulated as given below by taking an action $a_*(s)$ at state $s$ and proceeding to state $s_{t+1}$ after the transition occurred.

$$V_\pi(s_t) = \sum_{s_{t+1}} T(s_{t+1}|a_*(s_t), s_t)V_\pi(s_{t+1}),$$

where $s_t = (x, t)$ and $s_{t+1} = (x', t+1)$, and $x \in \{0, 10\}$ represents the distance from start position $x_0$. Time step $t$ increase monotonically within each transition, and $V_\pi(x, t)$ depends on $a_*(x, t)$ and $V_\pi(x', t')$. For all combinations of $s_t = (x, t)$, by recursively simulating the state value function under each action $a_*(x, t)$ and updating $V_\pi(x, t)$ for each state by following formula (26), the optimal policy can be obtained when the update is converged. The pseudocode regarding the dynamic programming for American football is given in Algorithm 1.

4. Results

4.1. MDN Prediction of Proceeding Distance for American Football

At first, we show the trained prediction results from MDN. As mentioned before, the environment of American football for IRL is simulated using the trained MDN to estimate the probability distribution of the next state. The input is the two states and one action, and the states have two elements, namely “down” and “distance.”
Initializing reward function parameter \( \theta \)

Initializing state function \( f(s_t) \)

Initializing visiting frequency \( \mu(s_t) \)

Trained MDN network

State transition: \( T(s_{t+1}|s_t,a) \)

Calculating reward function: \( r(s, \theta) = \theta^T \cdot f(s) \)

Estimating value function:
\[
Q(s_t, \alpha) = R(s_t, \theta) + \gamma \left( \sum_{s' \in S} T(s_t, s'|a) V^\pi(s') \right)
\]

Estimating optimal policy \( \pi^* \):
\[
\pi^* = \arg \max_r \left( Q(s, \theta) + \gamma \left( \sum_{s' \in S} T(s, s'|a) V^\pi(s') \right) \right)
\]

Estimating visiting frequency \( \mu(s_t, \theta) \):
\[
\mu(s_t, \theta) = \sum_{a \in A} \mu_{s_t}(a) \pi(s_t, a|T(s_t)| s_t, a_t)
\]

Calculating total visiting frequency:
\[
P(s_t|\theta) = \sum_{s' \in S} \mu(s_t)
\]

Calculating the difference of visiting frequency:
\[
\Delta \mu = D(s_t) - P(s_t|\theta)
\]

Update reward function parameter \( \theta = \theta + \alpha \cdot \Delta \mu \)

**Figure 3:** Sketch of MDN principle. The state and action pair are the input for MDN. The output of MDN is the probability distribution of the proceeding distance under the input of current state and executed action. The output of MDN is analytical mixed Gaussian probability density distribution, and its corresponding sampling probability table is shown here for better understanding. An example is also presented here to illustrate the flow of how MDN is working.

**Figure 4:** Graphical illustration of the flow chart for the implementation of Max-Ent IRL algorithm for virtual American football.
The expert data is professional play-by-play data in the NFL 2017 season. The dataset contains 9218 plays of 21 games by 32 teams (reference). The original information of the data includes the date, time, opposing team, the team holding the ball, selected play, distance advanced, defensive formation, time remaining, and so on for each play set.

After learning in MDN, we obtained 1000 samples from the learned distance distribution. The sample output is a prediction of the distance advanced based on an input that is a combination of down, distance, and action. Figure 5 shows the results of MDN predicting the distribution of the distance advanced because of five different actions for each combination of states. Here, because of the space limit, we only show one example where the player is in the “1st down.” The horizontal axis in the plot of Figure 5 shows the predicted length of the advanced distance in the unit of yard. The distribution colored by pink color is the real data based on the play history of expert, and the one colored in cyan is the predicted distribution. The areas where the two graphs overlap are colored by purple color. It can be clearly seen that MDN is able to perform reasonable prediction for most of the action by comparing to the real data. However, special attention should be paid to some prediction results, which show deviation from the real data, mainly because of the irregular distribution of the real data, such as the “deep” action shown in Figure 5. Here, a multiple-peaked distribution profile exists in the real data. The mean of the dominating prediction distribution overlaps with the real data, while there exists a large deviation for the standard deviation between the predicted one and real one. This deviation also affects the second predicted distribution, which is poorly consistent with the real data. Because of the irregularities of the real data, the stochastic Max-Ent IRL policy was also severely influenced, on which we will give more details in the section of risk evaluation.

4.2. Stochastic Max-Ent IRL Simulation Results for American Football. In this section, we will show the simulated results by Max-Ent IRL simulation results for American football. In stochastic Max-Ent IRL, we trained the agent to learn the optimal under stochastic environment based on MDN. Under each training epoch, we monitored the update and convergence behavior of parameter $\theta$ and the difference in the visiting frequency of states between experts and agents. In this environment, the goal is to advance 10 yards at the end of 3rd downplay. It is because, in the American football, the basic requirement for scoring is to advance 10 yards within 3 plays. As mentioned before, in this work, the simulation is performed by calculating the distribution of the advanced distance by the action in each state using MDN and then randomly sampling 1000 data of distance advanced from it, discretizing the sample values from 0 to 10 and tabulating the percentages. Here, it is assumed that there is no backward movement when any action is taken. If the offense team exceeds the remaining distance when they take an action, the cumulative distance is set to 10. We used this transition probability to recurrently update the reward learning parameter $\theta$. Figure 6(a) shows the convergence behavior of parameter $\theta$ during the training procedure. In the initial stage of training, the amount of update is small, however, it converges after 50 epochs.

We compared the visiting frequency extracted directly from the expert and the best agent learnt from by Max-Ent IRL in each state. In Figure 6(b), from the left to right, there are three tables, which correspond, respectively, to the visiting frequency of the expert by acting with the highest possibility at each state, the visiting frequency of the trained Max-Ent IRL agent by taking action with the one-shot IRL-MDN, and the visiting frequency of the trained ensemble agent by taking an average rated action with the fifty-times-repeat of one-shot IRL-MDN. The dominating frequency highlighted with red color showed high similarity between the learning agent and the expert. Figure 6(c) showed the contour plot for the three visiting frequencies for better visualization. It is found that the frequency differences among the three conditions are negligible for the states located in the initial and at the end of the playground. However, clear discrepancy can be recognized at the middle of the playground, where the agent trained under the fifty-times-repeat of one-shot IRL-MDN showed a slightly closer feature to the expert visiting frequency, when compared to one-shot IRL-MDN. The results shown in Figure 6 suggest that the IRL agent understands the features of each action taken by the expert and is capable of selecting the action that maximizes the value function while following approximately the visiting frequency of the expert even in an environment where the transition probability changes and is difficult to predict.

Figure 7 shows the policy of the IRL agent learnt under various conditions. Figure 7(a) shows the policy table of the expert under the action with the highest probability. A typical policy feature of the expert is that the action “short” is dominating among all the games. One concern is that overlearning would occur under such unbalanced behavior. Fortunately, the overlearning or underlearning...
issue can be effectively mitigated because the agent is not directly mimicking the behavior pattern of the expert. Instead, the Max-Ent IRL algorithm is designed to learn the reward function, i.e., the hidden motivation behind the expert to make a decision, from the visiting frequency of the expert. The policy table learnt by the Max-Ent IRL agent under one-shot MDN is shown in Figure 7(b). It can be clearly seen from here that the policy showed more varieties than the expert table, indicating that the agent is not mimicking the behavior of the expert. One of the main features is that a large number of “deep” actions is chosen in the policy table of the Max-Ent IRL agent. The tendency becomes more apparent for the trained agent with the fifty-times-repeat of one-shot IRL-MDN (Figure 7(c)). Since the actions “deep” and “short” belong to the same category of pass play, we claim that Max-Ent IRL agent can grasp the feature of expert behavior and effectively learn the policy even under the action table shown in Figure 7(a). The results presented so far were obtained under the Max-Ent IRL learning scheme, where the expert took the greedy action based on the highest possibility. The state frequency and policy tables obtained under the expert acting with the second highest probability are presented in Figures 8 and 9. In the next section, we will show a thorough comparison in terms of risk evaluation and policy verification among all the greedy and non-greedy policies.

At last, we show the policy simulated by the reference DP algorithm, and the results are shown in Figure 7(d). In the DP algorithm, we calculated the value function for all states and estimated the optimal action. An apparent feature shown in Figure 7(d) is that the dominating action is the action “short,” which is the same as shown in the policy table of expert in Figure 7(a). Meanwhile, another feature in DP policy table is that when the agent is at the long distance to goal, the policy is to choose pass play, i.e., actions such as “short” or “deep.” On the other hand, when the agent is near the goal, the DP policy suggests that it is better to choose run play (e.g., guard, tackle, and end). The results are consistent with the perspective of domain knowledge. However, one drawback of DP approach is the issue of flexibility because the policy is greedy in the sense that only one best action is presented from the mathematical optimization. In the real game, the strategies for the coach to decide which play to choose show a lot of uncertainties and are heavily dependent on ad-hoc condition and situation during each play such as scores, player condition, and previous tactics. From this point of view, the Max-Ent IRL algorithm is more favorable since multiple policy candidates are available even under the same situation.
5. Risk Evaluation and Policy Verification

Since all the policies obtained in this work are performed under stochastic conditions, it is important to design a rigorous measure to evaluate the quality of the learnt policy. In this work, we designed two approaches to perform the evaluation. One is based on the risk and return evaluation. It is a conventional approach used in financial engineering field. Another one is the test demonstration of the trained model on the data that are not included in the training data, which is a standard measure used in the field of machine learning [17].

At first, we give a brief mathematical introduction on how the risk and return are calculated in this work. In this work, the risk is defined as a variance of the distance that the offense team may have advanced after the three plays. A return is defined as the expected value of the distance advanced after three plays. The number of plays is $t \in \{1, 2, 3\}$, distance from start position at time $t$ is $x_{i,j,k} \in \{0, 1, \ldots, 10\}$, and action selected at time $t$ is $a_{t} \in \{0, 1, \ldots, 4\}$. $P_{x_{t}}$, which is the probability of being at a distance $x$ at time $t$ after three plays, is as follows:

$$P_{x_{t}} = \sum_{x_{i,j,k}=0}^{10} \sum_{x_{j,k}=0}^{x_{t}} \left\{ T(0,a_{1},x_{i})T(0,a_{2},x_{i})T(0,a_{3},x_{k}) \right\},$$  \hspace{1cm} (27)$$

where, $i$ represents the advanced distance after the first play, and $j$ represents the advanced distance after the second play. $T(0,a_{1},x_{i})T(0,a_{2},x_{j})T(0,a_{3},x_{k})$ represents the probability of the advanced distance where the trajectory is $0 \rightarrow i \rightarrow j \rightarrow x_{k}$. We calculate all combinations of trajectories and get a probability of reaching $x_{j}$ after three plays from the starting position. In this case, we only consider $i \leq j$ to test in a simple situation. The expected value of advanced distance after three plays is as follows:

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|c|c|c|}
\hline
\textbf{Epoch} & \textbf{0} & \textbf{0.000} & \textbf{0.005} & \textbf{0.010} & \textbf{0.015} & \textbf{0.020} & \textbf{0.025} & \textbf{0.030} & \textbf{0.035} & \textbf{0.040} \\
\hline
\textbf{Error} & \textbf{0.015} & \textbf{0.020} & \textbf{0.025} & \textbf{0.030} & \textbf{0.035} & \textbf{0.040} & \textbf{0.045} & \textbf{0.050} & \textbf{0.055} & \textbf{0.060} \\
\hline
\end{tabular}
\end{table}

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure6}
\caption{(a) Learning curve of reward parameter during the training procedure and the learnt visiting frequency of the best learning agent together, with the visiting frequency extracted directly from the expert data. (b) The visiting frequency of the expert by acting with the highest possibility at each state (left), the visiting frequency of the trained Max-Ent IRL agent by taking action with the one-shot IRL-MDN (middle), and the visiting frequency of the trained ensemble agent by taking an average rated action with the fifty-times-repeat of one-shot IRL-MDN (right). (c) Contour plot for the three visiting frequencies shown in (b).}
\end{figure}
State visiting frequency of the expert by taking the action with the 2nd highest possibility
State visiting frequency of the trained IRL agent by taking the action with one-shot of MDN
State visiting frequency of the trained IRL agent by taking the action with average of 50-shots of MDN

Figure 7: (a) Policy table of the expert under the action with the highest probability; (b) policy table learnt by the trained Max-Ent IRL agent under one-shot MDN; (c) policy table learnt by the trained ensemble agent by taking an average rated action with the fifty times repeat of one-shot IRL-MDN; (d) policy table of reference DP algorithm.

Figure 8: State frequency comparison for the IRL agents and the expert taking the action with the second highest probability.
The variance of advanced distance after three plays and the corresponding risk standard deviation are given below. The standard deviation is a measure of risk in this work.

\[ \text{V}[x] = \sum_{x_k=0}^{10} \left( (x_k - E[x])P_{x_k} \right)^2; \sigma = \sqrt{V}. \] 

Figure 10 shows a plot that evaluates risks and returns based on optimal policy calculated under various conditions. To better understand the risk-return distribution characteristics, we have manually sketched the well-known risk-return frontier guideline in Figure 10. There are three typical ranges along the frontier line, namely the inefficient frontier range located at the lower part, the efficient frontier range located at the top, and the global minimum variance point located at the cross-point between the inefficient and efficient frontier. By following the frontier guideline, it can be clearly seen that the IRL agent trained by the expert action with the highest probability and averaged values is distributed near the global minimum variance point. It is interesting to note that the policy from the trained IRL presented slightly lower risk and higher return when compared to the policy directly from the expert action with the highest probability. The policy from reference DP algorithm is located at almost the same place as the trained IRL agent. The global minimum variance point represents the return with the lowest risk. It indicates that the policy obtained under the expert data in this work is categorized as a "conservative“ policy. Meanwhile, since the curvature of the frontier line is relatively flat, the return in the efficient frontier line region showed no dramatic difference to the global minimum variance point. Therefore, although the "conservative" feature is in the learnt policies, they are still
able to reach a reasonable return. However, we found that for the policy from the IRL agent trained by the expert action with the second probability showed risk and return feature in the inefficient frontier line region, where the high risk is accompanied with low return. The same tendency is also found for the expert action with the second highest probability.

At last, we gave a brief discussion about policy verification using the risk-return plot. We chose five test NFL teams, whose play history data are not included in the training for the IRL agent. It is difficult to directly apply to the game of the test teams. We, instead, mapped the policy of the five test teams to the risk-return plot, and the results are shown in Figure 10. It can be seen from Figure 10 that the risk-return feature of two winning teams showed a closer risk-return feature to the optimal policy of the IRL agent. The two teams that lost the game showed more features along the inefficient frontier line. However, we found that one team that won the game to also be located in the inefficient frontier line. We claim that the verification showed that the optimal policy obtained in this work is able to represent a general feature of the play strategies among all the teams. However, there still exists the issue of overlearning, which, we consider, could be improved by adjusting the imbalance of the training data by either increasing the number of data or by performing data preprocessing, such as feature extraction or data filtering before IRL execution.

6. Related Works

Previous research on applying data analysis techniques to American football focused on play estimation [18], the predictive analysis of play success using tracking data [19], and the evaluation of players as the quarter back [20]. RL has proven its usefulness in environments where the concept of “competition” is present. The techniques have been applied to analyze and evaluate strategies and assist in the process of decision making in competitive environments. For example, virtual game RL programs, such as AlphaGo [21], AlphaGo Zero [22], and Atari [23], have defeated human players. Moreover, RL has also been applied to practical sports, such as Ice Hockey [24], Soccer [25, 26], Basketball [27], and smart grid energy optimization [17, 28]. Various IRL algorithms have been proposed so far [29] for various applications, such as inferring and optimizing the strategy of offensive and defensive play in soccer game [30], optimizing the brush-stroke quality in computer-generated painting [31], learning the car-following behavior of drivers [32], programming by demonstrations to develop advanced learning systems [33], and boosting table-to-text generation [34, 35]. However, none of these works have investigated policy-learning under stochastic environment and conducted a joint evaluation of the learnt policy with risk-return analysis.

7. Conclusions

In this work, we demonstrate an efficient approach based on a stochastic inverse reinforcement leaning (IRL) algorithm constructed by integrating the conventional max-entropy IRL algorithm with the mixture density network (MDN). We take the case study of American football, a sports game processing strong stochastic feature, since the number of yards gainable on the next offence in real American football is usually uncertain during the stage of strategy planning and decision making. The American football simulation environment was built by training MDN using the annual NFL data to generate the state transition probability for IRL. Under the framework of Max-Entropy IRL, an optimal strategy was successfully obtained through a learnt reward function via the visiting frequency by trial-and-error communication with the MDN environment. At last, we performed risk-return analysis on the learnt policy. It is found that all the optimal policies from either the expert or the learnt agent with the best selected action or the reference DP algorithm showed the “conservative” feature to minimize the risk. Meanwhile, the policy of the trained IRL agent showed a slightly higher return and lower risk than the expert data used to train the IRL agent. It indicates that it is possible for the proposed IRL algorithm to form a superior policy than the one derived statistically from the teaching expert data. We also performed policy verification using five NFL teams whose play data were not used in the training data. It is found that the policy of the winning team and lost team is roughly consistent with the feature of the optimal policy obtained in this work. However, we found that there still exists the issue of overlearning, which leads to the failure of mapping the winning policy into the inefficient frontier line. The performance of the propose approach could be improved by adjusting the imbalance of the training data by either increasing the number of data or by performing data preprocessing, such as feature extraction or data filtering before IRL execution.

Data Availability

The simulation code for reproducing the simulation results can be accessed from https://github.com/KShiba24/IRL-MDN-RISK-NFL.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by New Energy and Industrial Technology Development Organization (NEDO) (Grant no. JPNP20015). The authors gratefully acknowledge the funding from New Energy and Industrial Technology Development Organization (NEDO) and the Ministry of Economy, Trade and Industry (METI), Japan.
References

[1] N. Horney, B. Pasmore, and T. O’Shea, “Leadership agility: an imperative for a VUCA world,” People & Strategy, vol. 33, no. 4, p. 32, 2010.

[2] N. Bennett and G. J. Lemoine, “What a difference a word makes: understanding threats to performance in a VUCA world,” Business Horizons, vol. 57, no. 3, pp. 311–317, 2014.

[3] R. S. Sutton and A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, Mass, 1998.

[4] A. W. Moore, “Efficient memory-based learning for robot control,” Diskurs, vol. 34, p. 87, 1990.

[5] Y. Yang, W. Gao, H. Modares, and C.-Z. Xu, “Robust actor–critic learning for continuous-time nonlinear systems with unmodeled dynamics,” IEEE Transactions on Fuzzy Systems, vol. 30, no. 6, pp. 2101–2112, 2022.

[6] Y. Yang, H. Modares, K. G. Vamvoudakis, W. He, C.-Z. Xu, and D. C. Wunsch, “Hamiltonian-driven adaptive dynamic programming with approximation errors,” IEEE Transactions on Cybernetics, pp. 1–12, 2021.

[7] A. Turgo, “A novel approach to the problem of reinforcement learning for robot control,” Journal of Intelligent & Robotic Systems, vol. 12, no. 21, pp. 691–730, 2011.

[8] A. W. Moore, “Efficient memory-based learning for robot control,” Diskurs, vol. 34, p. 87, 1990.

[9] Y. Yang, H. Modares, K. G. Vamvoudakis, W. He, C.-Z. Xu, and D. C. Wunsch, “Hamiltonian-driven adaptive dynamic programming with approximation errors,” IEEE Transactions on Cybernetics, pp. 1–12, 2021.

[10] Y. Luo, O. Schulte, and P. Poupart, “Inverse reinforcement learning for team sports: valuing actions and players,” in Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, pp. 3356–3363, Japan, Yokohama, 2020.

[11] G. Brockman, “OpenAI Gym,” 2016, https://arxiv.org/abs/1606.01540.

[12] Y. Luo, O. Schulte, and P. Poupart, “Inverse reinforcement learning for team sports: valuing actions and players,” in Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, pp. 3356–3363, Japan, Yokohama, 2020.

[13] P. E. Utgoff, “Inverse reinforcement learning,” in Encyclopedia of Machine Learning, C. Sammut and G. I. Webb, Eds., Springer US, Boston, MA, pp. 554–558, 2011.

[14] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey, “Maximum entropy inverse reinforcement learning,” AAAI, pp. 1433–1438, 2008, https://dblp.uni-trier.de/db/conf/aaai/aaai2008.html\#ZiebartMBD08.

[15] G. An, S. Moon, J.-H. Kim, and H. O. Song, “Uncertainty-based offline reinforcement learning with diversified Q-ensemble,” Advances in Neural Information Processing Systems, vol. 34, pp. 7436–7447, 2021, [Online]. Available: https://proceedings.neurips.cc/paper/2021/file/3d3d286aad153a4a58156d0e02d8570c-Paper.pdf.

[16] J. Choi and K.-E. Kim, “Inverse reinforcement learning in partially observable environments,” Journal of Machine Learning Research, vol. 12, no. 21, pp. 691–730, 2011.

[17] C. M. Bishop, Mixture Density Networks, Aston University, England, 1994.

[18] M. Wulfmeier, P. Ondruska, and I. Posner, “Maximum entropy deep inverse reinforcement learning,” [Online]. Available: https://arxiv.org/abs/1507.04888, 2016.

[19] T. Sogabe, D. B. Malia, C.-C. Chen, and K. Sakamoto, “Attention and masking embedded ensemble reinforcement learning for smart energy optimization and risk evaluation under uncertainties,” Journal of Renewable and Sustainable Energy, vol. 14, no. 4, p. 045501, 2022.

[20] R. Yurko, F. Matano, L. F. Richardson et al., “Going deep: models for continuous-time within-play valuation of game outcomes in American football with tracking data,” Journal of Quantitative Analysis in Sports, vol. 16, no. 2, pp. 163–182, 2020.

[21] N. Lee and K. M. Kitani, “Predicting wide receiver trajectories in American football,” in Proceedings of the 2016 IEEE Winter Conference on Applications of Computer Vision (WACV), pp. 1–9, Lake Placid, NY, USA, 07–10 March 2016.

[22] B. C. Burke, “DeepQB: deep learning with player tracking to quantify quarterback decision-making & performance,” Decision, vol. 56, p. 97, 2019.

[23] D. Silver, A. Huang, C. J. Maddison et al., “Mastering the game of Go with deep neural networks and tree search,” Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[24] D. Silver, J. Schrittwieser, K. Simonyan et al., “Mastering the game of Go without human knowledge,” Nature, vol. 550, no. 7676, pp. 354–359, 2017.

[25] V. Mnih, K. Kavukcuoglu, D. Silver et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[26] G. Liu and O. Schulte, “Deep reinforcement learning in Ice Hockey for context-aware player evaluation,” in Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, pp. 3442–3448.

[27] G. Liu, Y. Luo, O. Schulte, and T. Kharrat, “Deep soccer analytics: learning an action-value function for evaluating soccer players,” Data Mining and Knowledge Discovery, vol. 34, no. 5, pp. 1531–1559, 2020.

[28] A. Scott, K. Fujii, and M. Onishi, “How does AI play football? An analysis of RL and real-world football strategies,” [Online]. Available: https://arxiv.org/abs/2111.12340, 2022.

[29] N. Sandholtz and L. Bornn, “Markov decision processes with dynamic transition probabilities: an analysis of shooting strategies in basketball,” Annals of Applied Statistics, vol. 14, no. 3, p. 678, 2020.

[30] T. Sogabe, “Optimization of decentralized renewable energy system by weather forecasting and deep machine learning techniques,” in Proceedings of the 2016 IEEE Innovative Smart Grid Technologies - Asia (ISGT-Asia), pp. 1014–1018, Melbourne, Australia, Nov. 2016.

[31] S. Adams, T. Cody, and P. A. Beling, “A survey of inverse reinforcement learning,” Artificial Intelligence Review, vol. 55, no. 6, pp. 4307–4346, 2022.

[32] P. Rahimian and L. Toka, “Inferring the strategy of offensive and defensive play in soccer with inverse reinforcement learning,” in Machine Learning and Data Mining for Sports Analytics, U. Brefeld, J. Davis, J. Van Haaren, and A. Zimmermann, Eds., vol. 1571, pp. 26–38, Springer International Publishing, Cham, 2022.

[33] M. Zhang, “Real-time generation method of oil painting style brushstrokes based on inverse reinforcement learning,” Mobile Information Systems, vol. 202, pp. 1–10, 2022.

[34] Y. Zhou, R. Fu, and C. Wang, “Learning the car-following behavior of drivers using maximum entropy deep inverse reinforcement learning,” Journal of Advanced Transportation, vol. 20, pp. 1–13, 2020.

[35] M. Mourad, A. Ezzeddine, B. Nadjar Araabi, and M. Nili Ahmadabadi, “Learning from demonstrations and human evaluative feedbacks: handling sparsity and imperfection using inverse reinforcement learning approach,” Journal of Robotics, vol. 22, pp. 1–18, 2020.

[36] Z. Shi, X. Chen, X. Qiu, and X. Huang, “Toward diverse text generation with inverse reinforcement learning,” 2018, https://arxiv.org/abs/1804.11258.

[37] S. Chaturvedi, S. Ghosh, Z. Qi, and S. Srivastava, “How helpful is inverse reinforcement learning for table-to-text
Mathematical Problems in Engineering

generation?” in Proceedings of the The Joint Conference of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language, p. 23, Aug. 2021.

[36] D. P. Rana and R. G. Mehta, Eds., Data Preprocessing, Active Learning, and Cost Perceptive Approaches for Resolving Data Imbalance, Engineering Science Reference, Hershey, PA, 2021.

[37] D. N. Anh, “Feature analysis for imbalanced learning,” Journal of Advanced Computational Intelligence and Intelligent Informatics, vol. 24, no. 5, pp. 648–655, 2020.