Minimizing Expected Loss for Risk-Avoiding Reinforcement Learning

Jung-Jung Yeh[1], Tsung-Ting Kuo[1], William Chen[2], Shou-De Lin[1]
[1]National Taiwan University
[2]Institute for Information Industry
Taipei, Taiwan
r97014@csie.ntu.edu.tw, d97944007@csie.ntu.edu.tw, williamchen@iii.org.tw, sdlin@csie.ntu.edu.tw

***Abstract*—This paper considers the design of a reinforcement learning (RL) agent that can strike a balance between return and risk. First, we discuss several favorable properties of an RL risk model, and then propose a definition of risk based on expected negative rewards. We also design a Q-decomposition-based framework that allows a reinforcement learning agent to control the balance between risk and profit. The results of experiments on both artificial and real-world stock datasets demonstrate that the proposed risk model satisfies the beneficial properties of an RL-based risk learning model, and also significantly outperforms other approaches in terms of avoiding risks.**

Keywords- reinforcement learning; risk avoiding; risk model; profit model

I. INTRODUCTION

The Oxford English Dictionary defines risk as "the possibility of loss, injury, or other adverse or unwelcome circumstance". The definition shows that risk has strong connection to the probability of loss. Some psychology studies show that "losses are weighted about twice as strongly as gains" [18]. In medical or automatic control systems, risk can refer to the possibility of a severe failure that may not be recoverable. In financial market, risk is related to loss of investment. As a result, several risk-averse strategies have been developed, particularly in the domain of finance and control systems.

Because of the need to strike a balance between making a profit and risks, we investigate the design of a reinforcement learning (RL) agent that can help users find such a balance. Traditional RL studies focus on the optimization of expected return given an underlying Markov decision process [17]. However, return-maximization RL agents ignore the distribution of the return and may therefore expose agents to higher risks while pursuing better rewards. In this paper, we try to integrate the concept of risk with an RL agent model by addressing the following two questions:

1. How should risk be defined and modeled in the realm of RL? Why would such definition be better?
2. Given the definition of risk, how can an RL framework facilitate learning about both risk and return?

With regard to the first question, the literature on RL agents provides four definitions of risk: the variance of return (Var(R), which is mostly used in the field of finance) [5][11]; the consideration of returns that are lower than the expected rate of return (R<E[R]) [14]; the loss in the worst case scenario (or minimum return, Min(R)) [7]; and the probability of fatal error P(Min(R)) [6]. However, each of these definitions has some limitations in modeling risks.

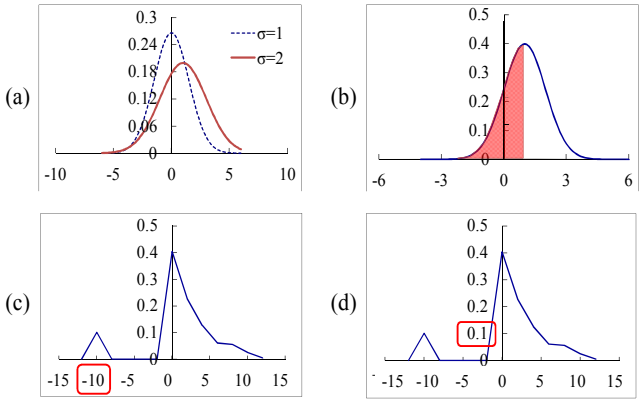

Figure 1. (a) Var(R), (b) R<E[R], (c) Min(R), and (d) P(Min(R)). The x-axis is the value of return and the y-axis is the probability of return.

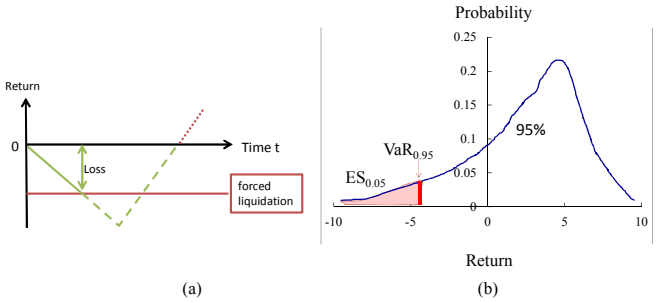

Figure 2. (a) Forced liquidation. (b) $VaR_{0.95}$ is the bounded value, and $ES_{0.05}$ is the average value of the colored region.

First, using variance to model risk penalizes positive but divergent returns. For example, Figure 1(a) shows variance-defined risk policy that favors a lower-variance distribution (σ=1), even though the higher-variance distribution apparently is less likely to suffer a loss and should be less risky. Penalizing returns lower than the expected return(R<E[R]) as shown in Figure 1(b), can also reduce the variance eventually, but it also penalizes positive return. In a nutshell, considering variance or expected rewards alone cannot fully capture the idea of risk. For example, it is not true that investing in a company which consistently earns $1±0.5 million annually is a higher investment risk than a company that earns $0.2 million almost every year, even though the former has a higher variance and receive penalty for R<E[R]. Using the worst possible return to model risk suffers the drawback of being too pessimistic because the worst case might be very rare, as shown in Figure 1(c). An RL-agent that minimizes the worst-case scenario focuses on preventing only a small number of undesired outcomes and ignores the majority of cases, which usually leads to a less

optimal situation on non-fatal situations. This is a similar situation where the probability of fatal events is used to model risk (Figure 1(d)). Furthermore, in some cases (e.g. stock trading) the concept of fatal outcomes is not apparent. The above analysis leads us to conclude that, although the definitions are suitable characteristics to have in a risk model, each of them independently is not sufficient to represent risk.

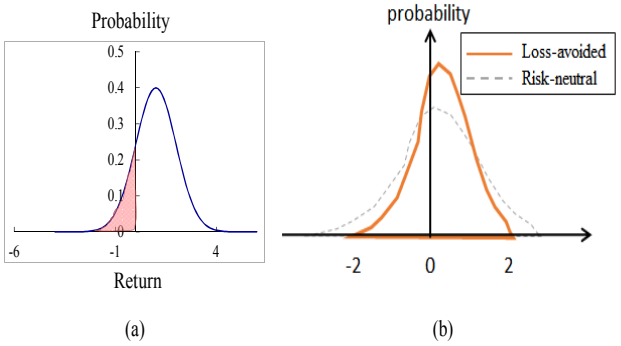

Figure 3. (a) The region of the expected loss R<E[R]. (b) An example of return distribution Var(R).

Besides these characteristics, we believe that three other features should be incorporated into a proper risk model as well. First, a suitable definition of risk should consider the level of loss (or degree of negative rewards). The level of loss is very important because certain types of investment (e.g., marginal buying or short selling) require the agent to keep the level of loss below a certain threshold; otherwise, forced liquidation (Figure 2(a)) could occur so that the agent has to suffer all the loss and quite the market. Second, the definition of risk should have the same measuring unit as profit. Ideally, an RL agent should be able to find a balance between risk and return based on the user's preference. To achieve this goal, the agent has to learn a tradeoff measure such as "how many units of profit would be sacrificed by increasing one unit of risk". Therefore, it is favorable if risk can be measured using the same metrics as return (the comparability). Finally, the definition of risk should be learnable by an RL agent. Value at risk (VaR) [8][9] and expected shortfall (ES) [1], as shown in Figure 2(b), are two well-known measures of risk that are not easily learnable by RL. VaR is defined as the loss a user would suffer under the given probability (usually set at 95%). For example, $VaR_{0.95}$ = -1 means the user has a 0.05 chance of receiving returns lower than -1. ES is defined as the expected loss given a probability (usually set at 5%). $ES_{0.05}$ represents average loss of the worst 5% of the returns. Although both metrics are widely used, it is hard for an RL agent to learn how to improve VaR and ES because normal RL agents do not recognize the distribution of rewards in the learning stage. Even if they do recognize the distribution, modeling them in a value function for learning is not a trivial task.

Based on the above discussion, we propose an alternative definition of risk for an RL agent: Risk is the expected loss (or the average cumulative negative rewards), as shown in Figure 3 (a), defined as:

$$E[R \mid R < 0]$$

Intuitively, this definition considers both the value and the probability of negative returns, and disregards positive returns. It is closer to the consensus concept of risk because people believe that a risky decision has more chance of causing serious loss. Moreover, the definition satisfies all of the mentioned favorable characteristics for risk (Table 1).

TABLE I. COMPARISON OF THE DIFFERENT DEFINITIONS OF RISK. COMP REPRESENTS THE COMPARABILITY OF RISK AND RETURN, AND LEARN REPRESENTS THE LEARNABILITY FOR RL AGENT

Feature / Definition	Var(R)	R < E[R]	Min(R)	P(Min(R))	E[R|R<0]	Comp	Learn
Var(R)	●	●					●
R < E[R]	●	●				●	●
Min(R)			●	●		●	●
P(Min(R))			●	●			●
E[R|R<0]	●	●	●	●	●	●	●

First, reducing E[R|R<0] also reduces the variance. This is because minimizing the expected loss moves the negative distribution curve to the right. Usually, minimizing the expected loss involves sacrificing the chance of higher returns (usually in the right edge of the distribution), and therefore the distribution becomes more centralized with smaller variance (Figure 3(b)). Second, the definition implicitly considers a fatal loss scenario because such a loss would be significant and should be avoided. Third, it models the level of loss directly, which allows our definition of loss to take the *default* situation in finance into consideration to avoid unrecoverable events, such as bankruptcy. Fourth, the expected loss is comparable to the return because they are both rewards; therefore the tradeoff between risk and return can be quantified. Finally, as will be shown later, it is learnable by an RL agent. We propose a Q-decomposition-based framework that can not only learn risk but also find a balance between risk and return.

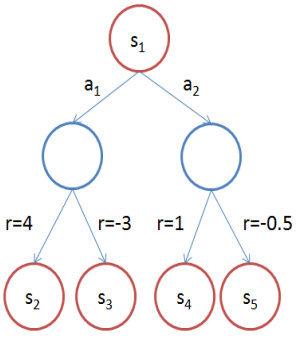

Figure 4. An example of loss-avoiding strategy.

An example for our loss-avoiding strategy is shown in Figure 4: s_1 is the current state, $\{a_1, a_2\}$ are possible actions, and the transition probability is 0.5 given any action. $Q(s, a)$ is expected reward given state s and takes action a. $Q(s_1, a_1)$=1 and $Q(s_1, a_2)$=0.5. The transition probability is 0.5 given any action. $Q(s_1, a_1)$=1 and $Q(s_1, a_2)$=0.5. Risk-neutral agent takes a_1 for the higher expected return. However, the plausible negative reward for the left route is 6 times more than that of the right route. Therefore, the loss-avoiding agent might want to take a_2 to avoid unfortunately enters to

s_5. This example shows that if we can model risk as the loss, then the agent can develop a policy involving less undesired outcomes.

II. Related Work

RL algorithms for risk management can be divided into several categories. In the category of variance-defined risk approaches, Sato and Kobayashi proposed a direct variance-penalized algorithm that utilizes TD-learning to estimate the variance of return [16]. Subsequently, Li and Chan introduce GARCH model to estimate the variance of return [11]. The model is designed specifically for predicting the variance of financial time series. However, there are two categories of risk in the variance calculation: upside and downside risk. For example, when traders invest money in highly fluctuant assets, the outcomes may be disastrous due to the fact that returns are much lower than expected. On the contrary, it is also possible for investors to make a fortune because returns of the assets exceed much more than the expectation. The former is called downside risk, and the latter is called upside risk [3]. Generally speaking, investors avoid losses whereas they are glad to earn the extra profits, so downside risk plays a more important role in making investment decision. In other cases, such as gambling, people are willing to pay a small quantity of money to make a big fortune, and it results in highly fluctuant returns. The main drawback of volatility algorithms is they views returns higher than expectation as risk while those returns are acceptable, or even encouraged.

Mihatsch and Neuneier proposed a risk-sensitive algorithm to handle cases where the actual return differs from the expected return [14]. In section 4, we will show that our proposed framework outperform this approach in terms of avoiding risk while maintaining high return.

Heger proposed an RL algorithm find an optimal policy in the worst-case scenario (Heger, 1994). It avoids risk by focusing exclusively on the largest possible damages in the control processes. As it focuses on rare events, the worst-case might not happen given limited training data, so it never guarantees a converged optimal policy.

Another approach is to transform rewards to exponential utility functions [10][12][13] and maximize the transformed return which represents the subjective utilities to people. It is often applied to construct efficient financial portfolio based Modern Portfolio Theory [4]. The utility transformation method is also the most popular and discussed risk-sensitive control but the time-dependent characteristics of the utility transformation approach prevent the formulation of consolidated RL algorithm [14].

The probability of fatal error control approach, which considers the problem of finding good policies with limited risk and is formalized as a constrained MDP [6]. However, it cannot guarantee whether there are feasible policies with the limited risk in discounted return problems or not.

III. Expected Loss Minimized Risk-Avoiding Reinforcement Learning

A reinforcement learning task can be described as an agent learns to make decisions in the environment which is modeled as state signals. If state signals summarize all past information needed to make a decision, the learning task is called Markov decision process (MDP) [18]. It is defined as $(S,\mathrm{A},\mathrm{T},\Re,\gamma)$, where S is state set, A is action set, T is transition function, $\Re$ is reward function, $S\times A\times S\to\Re$ and γ is discount factor. If the state space and action space are finite, it is called finite MDP. We focus on finite MDP problems in the following discussion.

At time t, an RL agent perceives state signal s_t, selects admissible actions $a(s)$ from A, and receives reward r_t. The mapping function from states to actions is called policy π, and $\pi(s_t, a_t)$ denotes the probability of selecting a_t given s_t. T, also called transition probability model, is defined as:

$$\mathrm{P}_{ss'}^{a}=\Pr\left\{\mathrm{s}_{t+1}=\mathrm{s}'\,|\,\mathrm{s}_t=\mathrm{s},\mathrm{a}_t=\mathrm{a}\right\}$$

$R_{ss'}^{a}$ is the expected reward received at the state transition from s to s'

$$\mathrm{R}_{ss'}^{a}=\mathrm{E}\left\{\mathrm{r}_{t+1}|\,\mathrm{s}_t=\mathrm{s},\,\mathrm{s}_{t+1}=\mathrm{s}',\mathrm{a}_t=\mathrm{a}\right\}.$$

For any policy π, the state-action value function $Q^{\pi}(s,a)$ is

$$\begin{aligned}Q^{\pi}(\mathrm{s},\mathrm{a})&=\mathrm{E}^{\pi}\left\{\mathrm{R}_t|\,\mathrm{s}_t=\mathrm{s},\mathrm{a}_t=a\right\}\\&=E^{\pi}\left\{\sum_{\mathrm{i}=0}^{\infty}\gamma^{\mathrm{i}}\mathrm{r}_{t+i+1}|\,\mathrm{s}_t=\mathrm{s},\mathrm{a}_t=a\right\}\\&=\mathrm{E}^{\pi}\left\{\mathrm{r}_{t+1}+\gamma\sum_{\mathrm{i}=0}^{\infty}\gamma^{\mathrm{i}}\mathrm{r}_{t+i+2}|\,\mathrm{s}_t=\mathrm{s},\mathrm{a}_t=a\right\}\\&=\sum_{\mathrm{a}}\pi(\mathrm{s},\mathrm{a})\sum_{\mathrm{s}'}\mathrm{P}_{ss'}\left[\mathrm{R}_{ss'}^{a}+\gamma Q^{\pi}(\mathrm{s}',\mathrm{a}')\right]\end{aligned}$$

The above equation is also called Bellman equations and to solve such control problems is to find a policy π that receives highest returns over the long run. It means RL agents should follow the best policy to earn as many returns as it can. If one policy π is better than or equal to another policy π', R^{π} are greater or equal to $R^{\pi'}$. The optimal policies are denoted as π* and the optimal state action value function $Q^{*}(s,a)$ is defined as:

$$\mathrm{Q}^{*}(\mathrm{s},\mathrm{a})=\max_{\pi}\mathrm{Q}^{\pi}(\mathrm{s}),\text{for all } s\in\mathbf{S} \text{ and } a\in\mathbf{A}$$

Q-learning [19] is an off-policy RL algorithm. The value functions are updated independently of actions of agents. The learning function given policy π is defined as:

$$Q(s_t,a_t)\leftarrow Q(s_t,a_t)+\alpha_t\left[r_{t+1}+\gamma\max_{a_{t+1}}Q(s_{t+1},a_{t+1})-Q(s_t,a_t)\right]$$

, where α_t is learning rate.

SARSA-learning [20] is, instead, an on-policy RL algorithm. It learns value function from the policy its agent performs. The learning function is:

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha_t\left[r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)\right]$$

Q-decomposition RL [14] is a RL method to particularly solve complicated learning problems which can be divided into sub tasks. The system is composed of multiple sub agents who learn its own state action value functions independently and one arbitrator who receives all Q-value from sub agents and makes a decision maximizing rewards from global view. Each sub agent learns value functions independently based on its own reward function. The arbitrator, however, doesn't have its own value function but defined as expected value from all Q-value of sub agents:

$$Q_A(s,a) = E[r_{t+1} + \gamma Q(s_{t+1},a_{t+1})] = \Sigma_{j=1}^{n} Q_j(s,a)$$

Based on our definition of risk, we propose an expected loss minimized risk avoiding reinforcement learning (RARL) framework that helps agents learn how to avoid risk as well as balance risk and return. The idea is that the agent should learn from unfavorable experiences that result in negative returns, especially if they involve significant losses, and avoid repeating them. Furthermore, we want the RARL framework to be flexible enough to incorporate users' preferences regarding the tradeoff between risk and profit. That is, the framework should allow users to indicate whether they want a risk-prone or risk-free agent, and even specify the preferred degree of risk (by using the risk aversion parameter κ).

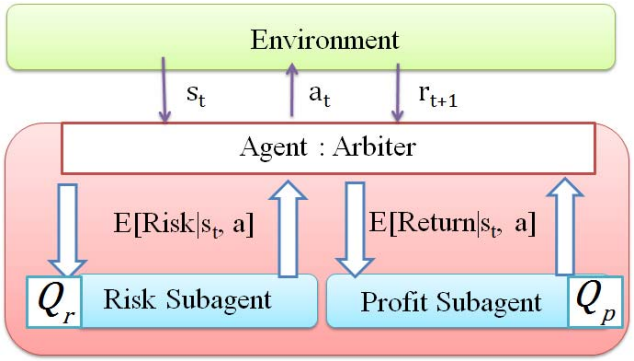

Figure 5. The RARL framework.

To achieve our goal, the RARL agent has to maintain both the expected loss and the expected return of decisions. However, as will be shown later through our experiments, handling the expected loss and the expected return in only one value function is difficult because they are of different scale and can hardly be optimized together. Therefore, we adopt the concept of the Q-decomposition RL framework [15] to design the RARL agent. Our model is comprised of an arbiter and two subagents, namely, a risk subagent and a profit subagent, as shown in Figure 5. The two subagents maintain and learn different value functions independently, while the arbiter coordinates both value functions to determine an optimal action to pursue.

The profit subagent works like a normal RL agent in that its value-function $Q_p(s,a)$ captures the expected overall return of the decisions, while it does not maintain information relevant to the distribution of the returns such as how the diversity of each return or how many times the decision has caused unacceptable losses. To deal with this drawback without imposing too much of a computational burden when learning the overall distribution, we introduce the second agent, the risk subagent whose value-function $Q_r(s,a)$ learns the risk. In our design, we further generalize the idea of *loss* to not necessary *the negative return (R<0)* but the return that are below certain sensitive-loss threshold λ (i.e., $R<\lambda$). Note that the parameter λ (usually a small value or zero provided by users) can be used to represent a constant cost or an unavoidable expenditure, such as the transaction charge for stock transfers or the walking cost of robots. In this way, the amounts of controllable loss do not be affected by static losses (i.e., losses that have nothing to do with the action the agent chooses) during learning. For the risk subagent, the updating function is the same as the profit agent:

$$Q_r(s_t,a_t) \leftarrow Q_r(s_t,a_t) + \alpha_t * \{r_{t+1} + \gamma Q_r(s_{t+1},a_{t+1}) - Q_r(s_t,a_t)\}$$

However, the update condition is different. The risk value function only updates itself when the expected return is lower than the pre-defined value λ. If the expected return is higher than λ, the value function does not change. This strategy allows the risk subagent to keep track of the situation when an unexpected loss occurs (i.e., the return $<\lambda$). Because $Q_r(s,a)$ only updates itself the return is lower than λ, the risk value function gradually approaches the expected controllable negative return as the number of training episodes increases.

Note that the subagents must follow an on-policy learning strategy, such as SARSA-learning, instead of the off-policy learning because, for Q-decomposition, subagents cannot decide what action to take (that is the arbiter's job); therefore, it is not appropriate to use off-policy learning like Q-learning to learn a greedy policy for RARL. The learning algorithm for risk-subagent is as follows:

1: Initialize $Q_r(s,a)$ arbitrarily
2: Repeat (for each episode)
3: Initialize s
4: Choose a from s using policy π derived from Q_r
5: Repeat (for every step in the episode)
6: Take action a', receive r, and switch to next state s'
7: Choose a' from $a(s)$ using π
8: If $r_t + \gamma\, Q_r(s', a') < \lambda$ Then
9: $Q_r(s, a) \leftarrow Q_r(s, a + \alpha * (r_t + \gamma\, Q_r(s', a') - Q_r(s, a))$
10: End if

The arbiter does not learn its $Q_a(s,a)$ on its own. Instead, it asks the subagents to report their Q-functions so that it can decide the best actions from a global perspective. It is defined as:

$$Q_a(s,a) = Q_p(s,a) + \kappa\, Q_r(s,a)$$

In the RARL framework, we incorporate a risk averse parameter κ∈[0,∞), which allows users to control the tradeoff between risk and return. When κ=0, the arbiter takes the action as profit subagent suggests; on the contrary, if κ→∞, it takes the action suggested by the risk subagent. The arbiter considers both subagents' value functions to satisfy different risk-averse requirements and make efficient trade-offs of risk and return. For example, the arbiter knows that by sacrificing a small amount of the return, it can avoid a significant amount of risk, thus it will take the safer action.

The computational complexity of RARL is the same as that of a single agent RL, since $Q_p(s,a)$ and $Q_r(s,a)$ are learned independently and subagents do not have to communicate with each other during the learning process.

IV. Experiments

We conducted two experiments, one on an artificial dataset (a grid world) and the other on a real-world dataset (the Electronic Stock Index). The grid world experiment is designed to show the characteristics of the proposed RARL agent. In the second (stock prediction) experiment, we compare our RARL agent with two agents that model risk in different ways: the Variance-Panelized RL (Sato & Kobayashi, 2000) and the risk-sensitive RL that penalizes R<E[R] [14]. We implemented the agent that avoids the worse-case scenario, but did not include it in our discussion because we found that such agent will eventually refuse to make transactions because doing so can surely avoid the worst-case scenario. Meanwhile, agents that try to avoid fatal errors are not compared because the concept of fatal outcomes is not obvious in stock trading problems.

A. Results on Grid World

We constructed a simple grid world to study the behavior of the RARL agents. Figure 6 shows a grid world without walking costs (assuming $\lambda = 0$), where S is the start state and G is the goal state. Agents receive rewards equal to the numbers in the grids. The action set A={→,↑}, and the policy is ε-greedy with ε=0.075. The learning rate α is initialized at 0.001 and decreases to 0.0001 within one million episodes. The highlight path is the greedy path after learning converges. We can see that when κ increases, the agent shifts from a high-reward-high-risk path in (a) to a low-reward-low-risk path in (b).

(a)

0	0	-2	-2	0	0	0	G
2	2	2	0	0	0	0	0
0	0	-5	-5	0	-5	-5	-5
0	0	-5	-5	0	-5	-5	-5
0	0	-5	-5	7	-5	-5	-5
0	0	-5	-5	7	-5	-5	-5
S	0	0	0	7	0	0	0

(b)

0	0	-2	-2	0	0	0	G
2	2	2	0	0	0	0	0
0	0	-5	-5	0	-5	-5	-5
0	0	-5	-5	0	-5	-5	-5
0	0	-5	-5	7	-5	-5	-5
0	0	-5	-5	7	-5	-5	-5
S	0	0	0	7	0	0	0

Figure 6. Agents walk from S to G; (a) the greedy path for κ=0, (b) the greedy path for κ=0.4.

The average return and the average loss in the final 10,000 episodes of different risk-averse agents are shown in Figure 7. When κ increases beyond 0.1, the proposed RARL agent switches its policy from a risky path to a safer one. It earns higher returns when κ∈[0,0.1); however, it also suffers more losses because it steps into a trap-rounded region more often. When κ∈[0.1,∞) the RARL agent's losses are lighter, but at the expense of smaller gains. Other grid experiments that we conducted produced similar outcomes.

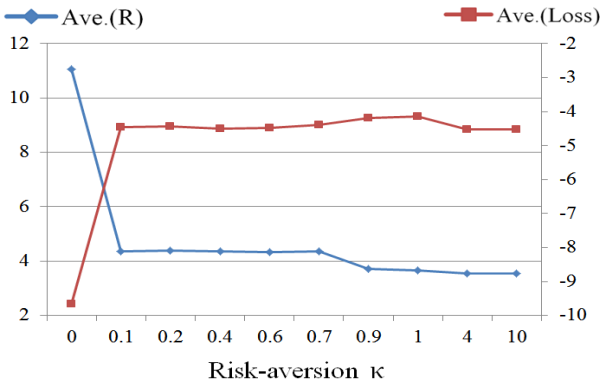

Figure 7. Average Losses and Average Returns for different κ.

B. Results on Real World Stock Data

In the experiment on the real-world Electronic Stock Index dataset, we developed a trading agent using the proposed RARL framework. We compare the framework with the following four alternative approaches (recall that 2 and 3 were introduced in Section 2):

1. A risk neutral system (κ=0)
2. A variance penalized agent
3. A risk-sensitive agent: in our experiment, we use both SARSA and Q-learning to implement this agent because it was proposed as a Q-learning agent.
4. Single-value function learning: We want to know if using a Q-decomposition framework is beneficial; thus, we construct a single RL agent to learn a combination of profit and expected loss values together:

$$Q_s(s_t, a_t) = E[R|\ s_t=s, a_t=a] - \kappa * E[R|R<0, s_t=s, a_t=a\ ]$$

The updating function is formulated as follow:

$$TD = r_t + \gamma * Q_s(s_{t+1}, a_{t+1}) - Q_s(s_t, a_t),$$

$$Q_s(s_t, a_t) = Q_s(s_t, a_t) + \alpha * (1+\kappa) * (TD), \text{ if } TD < 0$$
$$Q_s(s_t, a_t) + \alpha * (1-\kappa) * (TD), \text{ otherwise}$$

To measure profit, we use the overall capital gain (the compound rate of return during the whole investment period):

$$CG = \{\prod_{i=1}^{T}(1+R_i)\} - 1$$

We exploit four evaluation metrics to evaluate risk: expected loss, variance, $VaR_{0.95}$ and $ES_{0.05}$. As mentioned previously, the latter two metrics are widely used in the finance field but hard to be incorporated to RL value

functions. Here we want to show that our agent can perform well based on these two measures.

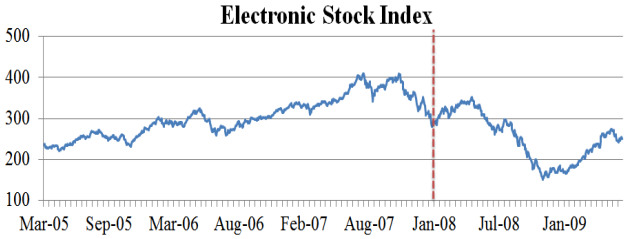

Figure 8. Electronic Stock Index.

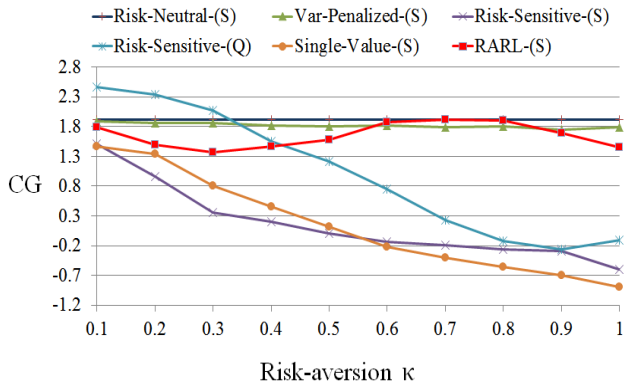

Figure 9. Capital gain of different risk avoiding agents.

We test our framework by designing an RARL trading agent for Taiwanese Electronic Stock Index market. In the experiment, we used data for the period March 2005 to November 2009 and the training-testing periods are separated from Jan 2008, as shown in Figure 8. As different risk-avoiding algorithms use specific risk-aversion parameters, we show their performance (CG and E[R|R<0]) for different levels of risk-aversion first (Figure 9 and Figure 10) and then compare their ability to avoid risk given a similar level of capital gain (Table 2). We allow the agents to perform 6 times of action every day.

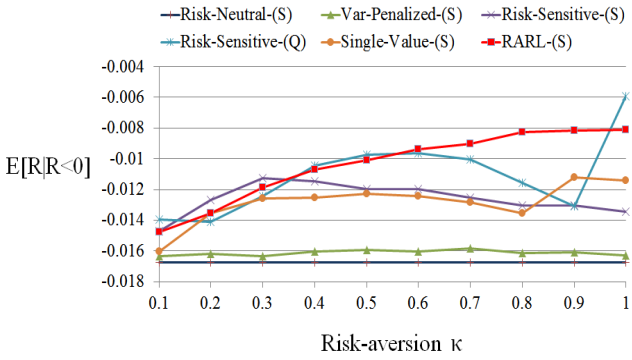

Figure 10. Expected loss for different risk avoiding agents.

The actions, which allow the agents to decide which position they want to maintain, are denoted as A∈ {short→1, neutral→0, long→1}. The long position means agents hold shares, the neutral position means no shares are held, and the short position means agents borrow shares to sell. The short position is profitable in a bear market because traders borrow shares to sell at a higher price and buy them back later at a lower price. The transaction cost, which is charged whenever a position changes, is set at ξ= 0.35%. If an agent switches its position from short to long, the transaction cost would be 0.7% and vice versa. We also define the interest rate (at 0.06% per day), and agents are charged interest if it stays in short position.

TABLE II. PERFORMANCE METRICS FOR RL AGENTS. NOTE THAT THE BEST CG ACHIEVED BY RISK-SENSITIVE AND SINGLE-VALUE IS ONLY 1.50 AND 1.46.

Measurement Type	Merged	Risk			
Agent \ Measure	CG	E[R|R<0]	σ[R]	VaR	ES
Risk-Neutral	1.91447	-0.01672	0.02093	-0.02465	-0.01841
Var-Penalized, κ=0.1	1.89143	-0.01633	0.02045	-0.02487	-0.01854
Risk-Sensitive, κ= 0.1	1.50497	-0.01469	0.01554	-0.02263	-0.01724
Risk-Sensitive, κ=0.3	1.90891	-0.01166	0.01466	-0.02604	-0.01679
Single-Value, κ=0.1	1.46346	-0.01605	0.01706	-0.0234	-0.01787
RARL, κ=0.8	1.90599	-0.00826	0.01179	-0.01812	-0.01489

In the state design, we use two types of technical indicators: the candlestick chart and the moving average. The candlestick chart lists four kinds of prices during a day t: open, close, maximal, and minimal prices. These prices generate 4! different candlestick charts for one day. We use the candlestick chart from day t-1 and t-2. The moving average MA in the previous k days with price P at day t is defined as:

$$MA_{t,k} = (\sum_{i=0}^{k-1} P_{t-i}) \div k$$

We consider $MA_{t,12}$ and $MA_{t,24}$, and discretize their values into 2^4 segments. A state is comprised of candlestick charts t-1and t-2, $MA_{t,12}$, $MA_{t,24}$, and the previous position. Therefore $|S| = 4!*4!*2^4*2^4*3 = 442{,}368$ states. An episode starts as well as end at the neutral position so the length of each episode is not fixed, and we found the average length of an episode of our RARL agent is about one day. The initial learning rate α is set at 0.001 and gradually reduced to 0.00001 after 5,000 episodes. The sensitive-loss threshold λ is set at a very small number, 0.08%, to tolerate transaction cost slightly. All experiments are repeated 20 times and the results are averaged to eliminate noise and randomness. Reward r_{t+1} is defined as:

$$r_{t+1} = ln(a_t*(p_{t+1} - p_t)/p_t) - \xi*|a_t - a_{t-1}| + 1, \text{ if } a_t = 0 \text{ or } 1$$
$$ln(a_t*(p_{t+1} - p_t)/p_t) - \xi*|a_t - a_{t-1}| - \omega + 1, \text{ if } a_t = -1$$

, where ξ is transaction cost which is set to 0.35% and ω is interest rate set to 0.01%. The first part of r_{t+1} is viewed as the capital gain from the investment and the second part is transaction costs and interest cost. We take the natural log on the overall return to reflect the fact that the overall rate of return of an investment is the compound sum of r_{t+1}.

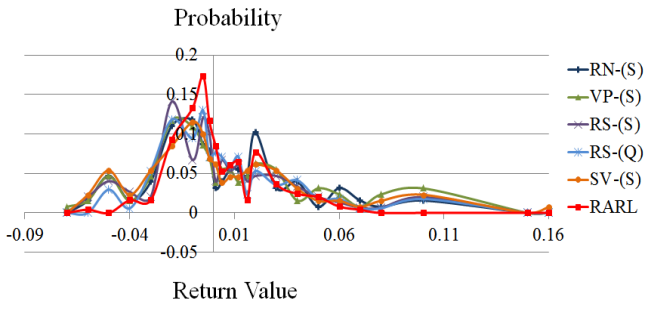

Figure 11. Distribution of return.

Figure 9 shows that the RARL agent's CG remains at high level with the increase of κ. Figure 10 shows that only the expected loss curve of our RARL agent is concave (the others are more unpredictable), which means the loss decreases more quickly with smaller κ. It is a nice property of RARL since small κ means agents do not sacrifice too many profits to reduce the risk. By combining both results, we can infer that, our RARL framework generally outperforms other methods in terms of balancing profit and risk. Var-Penalized failed to minimize the expected loss and Risk-Sensitive experiences a significant drop in CG as κ increases. The performance of Single-Value is inferior to that of the proposed RARL framework in every aspect, demonstrating that learning the expected return and expected loss statistics in a single value function is hard. Next, the CG for the agents is fixed to roughly 1.9(the CG earned by the risk-neutral agent) as shown in Table 2. Given the similar CG, our RARL agent outperforms other methods significantly in terms of all four indicators of Risk. Surprisingly, its variance still outperforms the variance-penalized agent, which is designed to reduce the variance of return. Figure 11 shows the distribution of return for different algorithms; it is clear that the RARL agent generally avoids high loss (by sacrificing some extremely high positive returns). Other agents suffer severe losses, especially when they are severer than -0.05. The distribution chart shows that RARL can really avoid significant loss comparing with other agents.

V. Conclusion

The contribution of this paper is threefold. First, we analyze the concept of risk for RL agents, and propose that an effective RL-based risk model should alleviate variance, avoid fatal losses, reveal the level of loss, be capable of comparing to the return metrics, and be learnable through designing proper value functions in an RL agent. Second, we propose using the expected loss to model risk and demonstrate through our experiments that this simple yet powerful metric satisfies the above criteria for a good risk model. Third, we propose the RARL framework, which exploits the q-decomposition method to learn risk and profit separately, and then integrates the value functions to balance the tradeoff between risk and return. Our RARL framework has two parameters that allow users to specify their preference for risk (i.e., the risk aversion parameter) and tolerance for loss (i.e., the loss-sensitive values). The promising results of experiments on artificial and real-world datasets demonstrate the efficacy of the proposed model.

Acknowledgement

This work was sponsored by AOARD grant number No. FA2386-13-1-4045 and Institute for Information Industry.

References

[1] Carlo Acerbi and Dirk Tasche. Expected Shortfall: a natural coherent alternative to Value at Risk. Economic Notes, 31(2), 379-388, 2002

[2] Aswath Damodaran. Investment Philosophies: Successful Investment Philosophies and the Greatest Investors Who Made Them Work. Wiley, 2003

[3] Edwin J. Elton and Martin J. Gruber. Modern portfolio theory and investment analysis. New York: John Wiley & Sons, 1995

[4] Jerzy A. Filar, Lodewijk C. M. Kallenberg and Huey-Miin Lee. Variance-Penalized Markov Decision Processes. Mathematics of Operations Research, Vol. 14, No. 1, pp. 147-161, INFORMS, 1989

[5] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. Journal of Artificial Intelligence Research, 24(1):81–108, 2005

[6] Matthias Heger. Consideration of Risk in Reinforcement Learning. In Proceeding of International Conference on Machine Learning, 105-111, 1994

[7] Philippe Jorion. Value at Risk: The New Benchmark for Managing Financial Risk. 3rd ed. McGraw-Hill, 2006

[8] Hisashi Kashima, Risk-Sensitive Learning via Minimization of Empirical Conditional Value-at-Risk. Transactions on Information and Systems, pages 2043-2052, Oxford University Press, 2007

[9] Sven Koenig and Reid Simmons. Risk-sensitive planning with probabilistic decision graphs. In Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, pages 363–373, 1994

[10] Jian Li and Laiwan Chan. Reward Adjustment Reinforcement Learning for Risk-averse Asset Allocation, In Proceeding of International Joint Conference on Neural Networks, 534 – 541, 2006

[11] Yaxin Liu, Richard Goodwin, and Sven Koenig. Risk Averse Auction Agents. In Proceedings of the international joint conference on Autonomous Agents and Multi Agent Systems, 2003

[12] Eric S. Maskin and John G. Riley. Optimal Auctions with Risk Averse Buyers. Econometrica, 52(1), The Econometric Society, 1984

[13] Oliver Mihatsch and Ralph Neuneier. Risk-sensitive reinforcement learning. Machine Learning, 49(2-3):267–290, 2002

[14] Stuart Russell and Andrew L. Zimdars. Q-decomposition for reinforcement learning agents. In Proceeding of International Conference on Machine Learning, 656–663, 2003

[15] Makoto Sato and Shigenobu Kobayashi. Variance-penalized reinforcement learning for risk-averse asset allocation. In Proceedings of Intelligent Data Engineering and Automated Learning, 244–249, 2000

[16] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning: An Introduction. The MIT Press, Mar 1998

[17] Richard H. Thaler, Amos Tversky, Daniel Kahneman, and Alan Schwartz. The Effect of Myopia and Loss Aversion on Risk Taking: An Experimental Test. The Quarterly Journal of Economics, Vol. 112, No. 2, In Memory of Amos Tversky, 647-661, 1997

[18] R. E. Bellman, Dynamic Programming. Princeton University Press, Princeton, NJ,1957

[19] Watkins, C. J. 1989. Learning from delayed rewards. Ph.D. thesis, Kings College, Cambridge, UK.

[20] Rummery, G. A., & Niranjan, M. (1994). On-line Q-learning using connectionist systems (Technical Report CUED/FINFENG/TR 166). Cambridge University Engineering Department.