UNIVERSITY OF TARTU

Faculty of Social Sciences

School of Economics and Business Administration

Mykola Herasymovych

Optimizing Acceptance Threshold in Credit Scoring using Reinforcement Learning

Master's thesis

Supervisors: Oliver Lukason (PhD), Karl Märka (MSc)

Tartu 2018

Name and signature of supervisor……………………………………………….

Allowed for defense on ……………………………………………….

(date)

I have written this master's thesis independently. All viewpoints of other authors, literary sources and data from elsewhere used for writing this paper have been referenced.

…………………………………..

(signature of author)

Acknowledgments

I would first like to thank the Quantitative Economics program coordinator Jaan Masso for making it possible for me to study in the University of Tartu, being always supportive and helpful in case I have any questions or issues with the studies. I would also like to thank staff and professors of the University of Tartu and College of Charleston for all the kind help and advice they have given and all the sacred knowledge they have shared with me during these 3 years of studies. I am greatly thankful to Estonian Ministry of Foreign Affairs Development Cooperation and Humanitarian Aid fund for the financial support of my master studies in the University of Tartu, and to the Harry & Reba Huge Foundation for funding my exchange semester studies in the College of Charleston.

I am sincerely grateful to my supervisors who helped me to produce this thesis. I thank my work supervisor Karl Märka for the chance to work on this topic, invaluable guidance in the idea development and implementation, opportunities to present the project in public and useful remarks on the paper. I also thank my employer Creditstar Group for providing the data and facilities to conduct the research. An equal amount of thankfulness goes to my university supervisor Oliver Lukason for consistent help with structuring the project, his priceless scientific expertise and helpful advice and comments on the paper. I additionally thank my referee Mark Kantšukov for the detailed review of the thesis and valuable suggestions on how to improve it.

Last and foremost, I would like to express my eternal gratitude to my family for their moral support, unconditional love and immutable belief in me. Special thanks go to Halyna Oros who was always there to listen to and even trying to understand my nerdy talks, distract when it was too much and motivate when it was time to get back to work. I also want to acknowledge my friends and groupmates who made my studies an exciting and unforgettable experience full of fun and positive emotions. Finally, I thank the people who are not explicitly mentioned in this acknowledgment letter but who helped me along the way.

Abstract

The thesis investigates issues of credit scoring model acceptance threshold optimization in a consumer credit company, such as model’s performance uncertainty, selection bias, population drift and business objective misspecification. We show that traditional static approaches based on cost-sensitive optimization do not ensure the optimality of the acceptance threshold, which might lead to biased conclusions and significant losses to the firm. We develop a dynamic reinforcement learning system that constantly adapts the threshold in response to the live data feedback, maximizing company’s profits. The developed algorithm is shown to outperform the traditional approach in terms of profits both in various simulated scenarios and on the real data of an international consumer credit company.

Keywords: consumer credit, credits scoring, cutoff point, reinforcement learning, model performance uncertainty, selection bias, population drift, profit scoring

Contents

1. Introduction

If one randomly picks a credit scoring article, there's a 50% chance that its introduction starts with mentioning the recent boom in the credit scoring literature caused by rapid increase in the computational capacity. The latter is also the reason of why more than half of the papers on the topic published during the last decade introduced a new credit rating approach (Louzada et al, 2016). However, due to data availability issues, most authors have access only to publicly available loan applications datasets. They are thus limited to performing a rather abstract kind of research without any ability to try their algorithms on the real data and face the issues of practical implementation. One of those important problems, usually omitted by researchers, is the problem of credit score acceptance threshold optimization that embodies the connection between a theoretical credit scoring algorithm and practical decision-making in a credit business process.

The sparse literature on the topic offers a traditional solution of picking an acceptance threshold that minimizes the misclassification costs (Viaene and Dedene, 2005; Hand, 2009) or maximizes the expected profit (Verbraken et al., 2014; Skarestad, 2017) based on an independent dataset of loan applications. The approach, however, suffers from oversimplifying theoretical assumptions that the misclassification costs are static and accurately known, while the independent dataset is identical to the general population of loan applications. In practice, the credit business environment has a dynamic nature and a high degree of uncertainty, especially in case of consumer loan providers specializing on subprime credit. Thus, we propose to solve the acceptance threshold optimization problem using a dynamic reinforcement learning (RL) system that constantly adapts the cutoff point in response to the live data feedback, maximizing company's profits.

RL algorithms have been intensively developing during recent years, solving dynamic optimization problems in various areas: from achieving superhuman performance level in board and video games (Silver et al., 2016; Mnih et al., 2015) to training self-controlling robots (Kober et al., 2013) to dynamically optimizing prices, marketing policies and loan portfolios (Kim et al, 2016; Sato 2016; Strydom, 2017) successfully outperforming traditional methods. The general nature of RL algorithms makes it relatively easy to adapt them to new problems. To the best of our knowledge, however, they have not yet been applied in the credit scoring. In our work, we explore if the application of RL approach could lead to a similar breakthrough in consumer credit business.

Consequently, the research aim of the thesis is to build a dynamic system that can be effectively used to optimize acceptance threshold in a credit scoring model in order to maximize credit company's profits. We achieve it with the following steps. First, we formulate the acceptance threshold optimization problem as a reinforcement learning task. Second, we train a Q-learning based reinforcement learning agent on a Monte Carlo simulation of a credit business process up to the point where its performance is at the level of the traditional approach. Third, we test the adaptive ability of the trained system in simulated scenarios of selection bias, population drift, and credit scoring model's performance uncertainty. Finally, to evaluate the developed algorithm, we test it on the real data and compare the results to the performance of the traditional approach.

The research contributes to the credit scoring literature by showing practical issues of a scoring model implementation on the example of an actual credit company. In particular, it demonstrates how model performance uncertainty, selection bias, and population drift problems make traditional acceptance threshold optimization approach produce suboptimal policy. It then contributes to both credit scoring and reinforcement learning literature by providing a proof of concept of using a dynamic reinforcement learning system to solve these problems and outperform the traditional approach. Finally, we conduct the research using real data of an international consumer credit company Creditstar Group, which adds practical sense to the work.

The results show that the traditional cutoff optimization approach does not ensure the optimality of the acceptance threshold, which might lead to biased conclusions and significant losses. The proposed dynamic reinforcement learning system manages to outperform the traditional method both in a simulated and real credit business environment leading to significantly higher total profits of the credit company. The main advantages of the developed approach are: 1) its constant adaptation to and learning from actual data generating process, which removes the need for theoretical simplifications and keeps the algorithm up to date; 2) flexible objective function definition that makes it easy to accurately specify the decision-maker's preferences and adjust them on the go if needed; 3) ability to train and test it in a simulated environment that lets the company avoid costly poor initial performance and stress-test various scenarios. Overall, the developed algorithm can be immediately put into practice to accompany lender's decisions and is currently used by the company as a decision support system.

The paper is structured as follows. The next section describes the background theory and the literature on credit scoring and reinforcement learning. Section three explains the problem setup and methodology used to solve it. The fourth section presents the results of the experiment.

Section five provides analysis of the results and limitations, accompanied by implications for science and business practice. Finally, the last section concludes.

2. Literature Review

The following section provides the theoretical background for credit scoring and reinforcement learning and reviews the relevant literature behind the two. It shows the research gap in credit score acceptance threshold optimization and describes how the reinforcement learning approach could fill it. Additionally, it discusses some examples of practical RL application in related finance and business fields.

2.1. Credit Scoring and its Major Challenges

We start this section by explaining the essence of the consumer credit scoring. One of the main aims of the credit business manager is to differentiate between good borrowers, which pay their loans in time, and bad ones that default on their loan within given time. However, due to the information asymmetry the manager does not know the type of a client beforehand and needs to decide whether to give a loan based on a set of variables provided by the client themselves (application data), third party data providers (credit agencies' data) or historical behavior of the customer (data on previously taken loans). Usually, the lender has a sample of loans that were given to clients and matured, thus letting the manager observe characteristics of borrowers and corresponding outcomes of the credit-granting decision. Thus, the problem can be described as a simple classification task.

Let's denote the vector of characteristics of a loan application as $\boldsymbol{x}$ and the outcome as a binary variable y, which is 1 if the loan is bad (goes overdue) and 0 if the loan is good (the payment is made in time)[1]. Then a variety of classification algorithms (e.g. logistic regression, decision trees) can be applied to predict the outcome variable or estimate the probability of the loan being bad:

$$\Pr\{Bad \mid characteristics\ \boldsymbol{x}\} = p(y = 1|\boldsymbol{x}) = \hat{y} \quad \forall\, \boldsymbol{x} \in \boldsymbol{X} \tag{1}$$

The estimated probability is transformed afterward, recalibrating to a more comprehensible range (Thomas et al., 2017) and possibly adjusting for company's policy objectives and rules

[1] There exist various approaches to define the outcome binary variable ranging from considering a loan bad if it is n number days overdue (usually n equals 60 or 90) to defining a bad loan as the one that has not been paid yet (Thomas et al., 2017). Each definition has its pros and cons and highly depends on the country's legislation and firm's business policy (Barisitz, 2013). In this work we consider a loan bad if it is 60 or more days overdue. The developed algorithm, however, is not sensitive to the outcome variable definition.

resulting in a credit score[2] s^{CS} for a particular application. Finally, an acceptance threshold (or cutoff point) t^{AT} for the credit score is chosen to make the decision, i.e. if $s^{CS} \geq t^{AT}$, then give a loan, otherwise – reject the application. If the score works appropriately, then it is possible to differentiate between the score's probability density functions of actual good and bad clients denoted $f_G(s^{CS})$ and $f_B(s^{CS})$ respectively, as depicted on the Figure 1 (Verbraken et al., 2014).

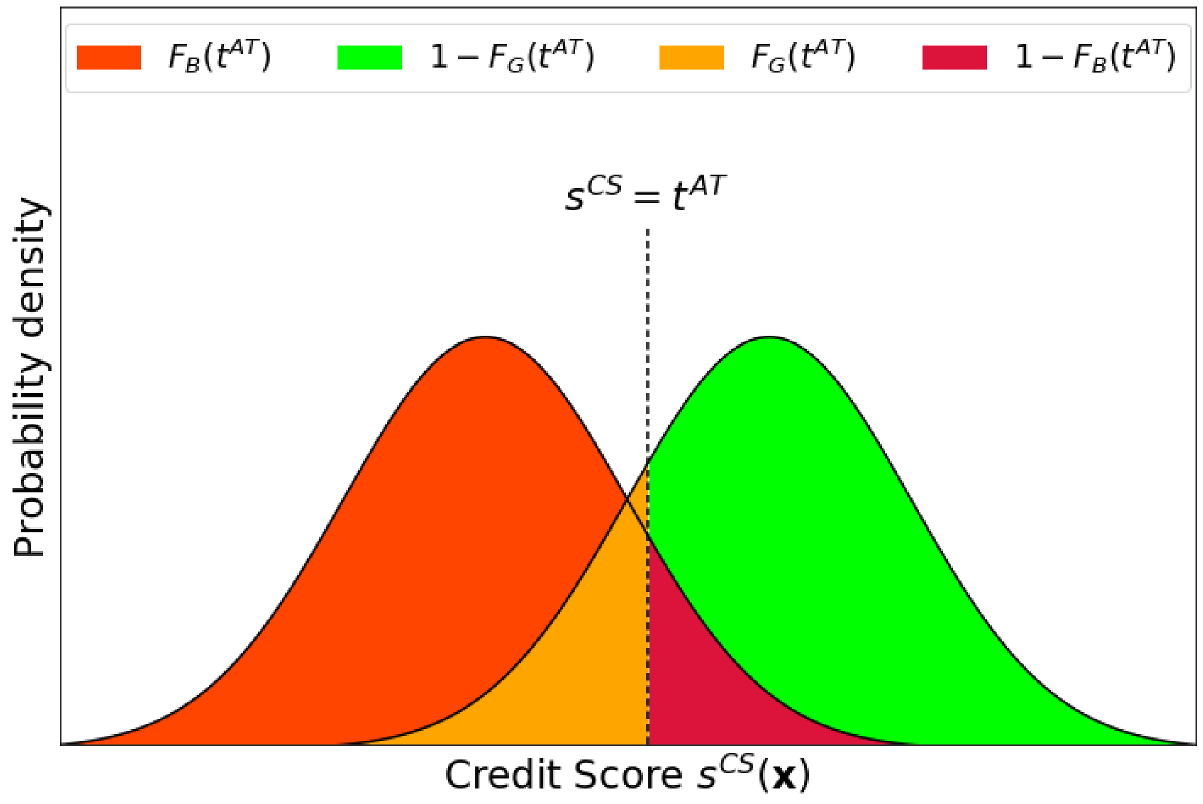

Figure 1. Example of credit score distributions and classification process.

Notes: Adapted from Crook et al. (2007), Hand (2009) and Verbraken et al. (2014). $s^{CS}(\boldsymbol{x})$ – application's credit score estimated based on the application data $\boldsymbol{x}$; $f_G(s^{CS})$ and $f_B(s^{CS})$ – credit score's probability density functions of actual good and bad applications respectively; t^{AT} – acceptance threshold for the credit score; $F_B(t^{AT})$ – correctly classified bad applications; $1 - F_G(t^{AT})$ – correctly classified good applications; $1 - F_B(t^{AT})$ – bad applications misclassified as good ones; $F_G(t^{AT})$ – good applications misclassified as bad ones.

The predictive performance of the model is then assessed comparing predicted outcomes to actual ones for a test dataset independent from the one the model was trained on. The variety of performance metrics is described in detail in meta-studies like Crook et al. (2007) and Louzada et al. (2016), but all of them in one or another way are related to the confusion matrix. The latter sums up the four possible groups of post-classification outcomes (see Table 1):

[2] A credit score is a numerical expression based on a level analysis of a loan application's characteristics, to represent the creditworthiness of an applicant.

1. Bad applications for which $s^{CS} < t^{AT}$ (area under $f_B(s^{CS})$ to the left of t^{AT}) are correctly classified and correspond to the value of cumulative density function of $f_B(s^{CS})$ up to threshold $t^{AT}, F_B(t^{AT})$. They are also known as true positives (n_{BB} in Table 1);
2. Good applications for which $s^{CS} \geq t^{AT}$ (area under $f_G(s^{CS})$ to the right of t^{AT}) are correctly classified and correspond to the value of cumulative density function of $f_G(s^{CS})$ over threshold $t^{AT}, 1 - F_G(t^{AT})$. They are also known as true negatives (n_{GG} in Table 1);
3. Bad applications for which $s^{CS} \geq t^{AT}$ (area under $f_B(s^{CS})$ to the right of t^{AT}) are misclassified as Good ones and correspond to the value of cumulative density function of $f_B(s^{CS})$ over threshold $t^{AT}, 1 - F_B(t^{AT})$. They are also known as false negatives or type II errors (n_{GB} in Table 1);
4. Good applications for which $s^{CS} < t^{AT}$ (area under $f_G(s^{CS})$ to the left of t^{AT}) are misclassified as Bad ones and correspond to the value of cumulative density function of $f_G(s^{CS})$ up to threshold $t^{AT}, F_G(t^{AT})$. They are also known as false positives or type I errors (n_{BG} in Table 1);

It is easy to see that frequencies of true negatives and false positives (the actual good clients) depend on the prior probability of an application being good π_G^{PP} and the cumulative distribution density function of the score for good applications $F_G(t^{AT})$, while frequencies of true positives and false negatives (the actual bad clients) depend on the prior probability of an application being bad π_B^{PP} and the cumulative density function of the score for bad applications $F_B(t^{AT})$. Consequently, all the values in the confusion matrix depend on the cutoff point value t^{AT}, meaning that the latter defines the overall Type I and Type II error rates of classification[3].

Thus, the final stage of building a credit scoring model which is going to be investigated in this thesis is the optimization of the acceptance threshold t^{AT}. Traditionally, an optimal cutoff point is either chosen to minimize classification error rate[4] on the independent test dataset or derived from a performance metric itself[5]. However, a lot of factors outside the basic classification methodology have to be accounted for when setting the threshold, such as actual

[3] Type I and Type II error rates of classification are calculated as $\frac{n_{GB}}{(n_{GB} + n_{BB})} = 1 - F_B(t^{AT})$ and $\frac{n_{BG}}{(n_{BG} + n_{BB})} = F_G(t^{AT})$ respectively.

[4] The classification error rate is calculated as $\frac{n_{GB} + n_{BG}}{(n_{GB} + n_{BB} + n_{BG} + n_{BB})}$.

[5] See Fluss et al. (2005) for examples.

misclassification costs, utility function of the decision-maker, sample selection bias, business cycle change and population drift.

Table 1. **Confusion matrix with misclassification costs.**

Predicted class	Actual class		
	Good	Bad	Predicted total
Good	$n_{GG} = \pi_G^{PP}(1 - F_G(t^{AT}))$ $[c(G\|G) = 0]$	$n_{GB} = \pi_B^{PP}(1 - F_B(t^{AT}))$ $[c(G\|B) = c_B]$	$n_{GG} + n_{GB}$
Bad	$n_{BG} = \pi_G^{PP} F_G(t^{AT})$ $[c(B\|G) = c_G]$	$n_{BB} = \pi_B^{PP} F_B(t^{AT})$ $[c(B\|B) = 0]$	$n_{BG} + n_{BB}$
Actual total	$n_{GG} + n_{BG}$	$n_{GB} + n_{BB}$	$n_{GG} + n_{BG} + n_{GB} + n_{BB}$

Notes: Adapted from Crook et al. (2007), Hand (2009) and Verbraken et al. (2014). The calculations extend the framework depicted on Fig. 1. For i and j being either G – good or B – bad: t^{AT} – acceptance threshold for the credit score; $F_i(t^{AT})$ – value of the credit score cumulative distribution function for applications of class i at point t^{AT}; π_i^{PP} – prior probability of an application being of class i; n_{ij} – number of applications of actual class i classified as class j; $c(i|j)$ – cost of classifying an application of actual class j as one of class i; in credit scoring the costs of correctly classifying an application are usually assumed to be zero leading to $c(G|G) = c(B|B) = 0$.

Recent advances in computational power and machine learning algorithms have led to a boom in the credit scoring literature. According to ScienceDirect search results, annual number of articles with the "credit scoring" keyword more than doubled during the last decade reaching 207 in 2017 (see Figure 2). Even though it is impossible to cover all the related papers in the literature review, meta-studies like Abdou and Pointon (2011), Crook et al. (2007), Lessmann et al. (2015) and Louzada et al. (2016) help to get an overview of the topic and the main trends of research in the credit scoring literature.

According to the systematic review in Louzada et al. (2016) which covers a representative sample of credit scoring papers since 1992, around 50% of articles introduce some kind of new method to rate credit applications and about 20% compare traditional techniques (see Figure 3). However, as was pointed out by various authors (Hand, 2006; Crook et al., 2007; Abdou and Pointon, 2011) the actual improvement in predictive performance that more sophisticated classification algorithms provide is only marginal and only observed for a particular dataset of credit applications (the so-called illusion of progress or flat maximum effect[6]). Thus, in recent

[6] See Hand (2006) for more information.

years it becomes more and more popular to discuss practical issues of a credit scoring model implementation (see Table A-1 of Appendix A for the summary of studies on major credit scoring practical issues).

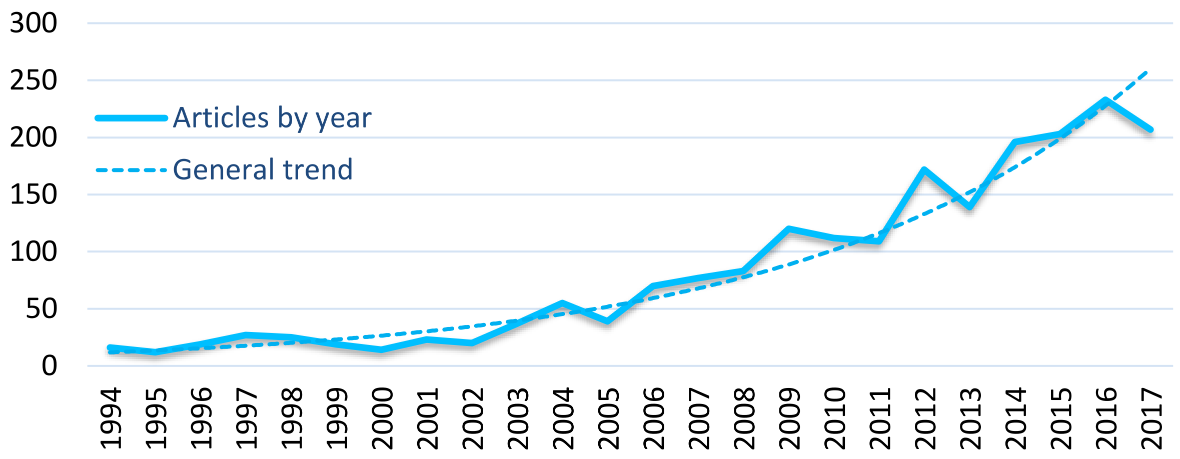

Figure 2. Number of published articles with "credit scoring" keyword.

Note: Based on the search results from the ScienceDirect.

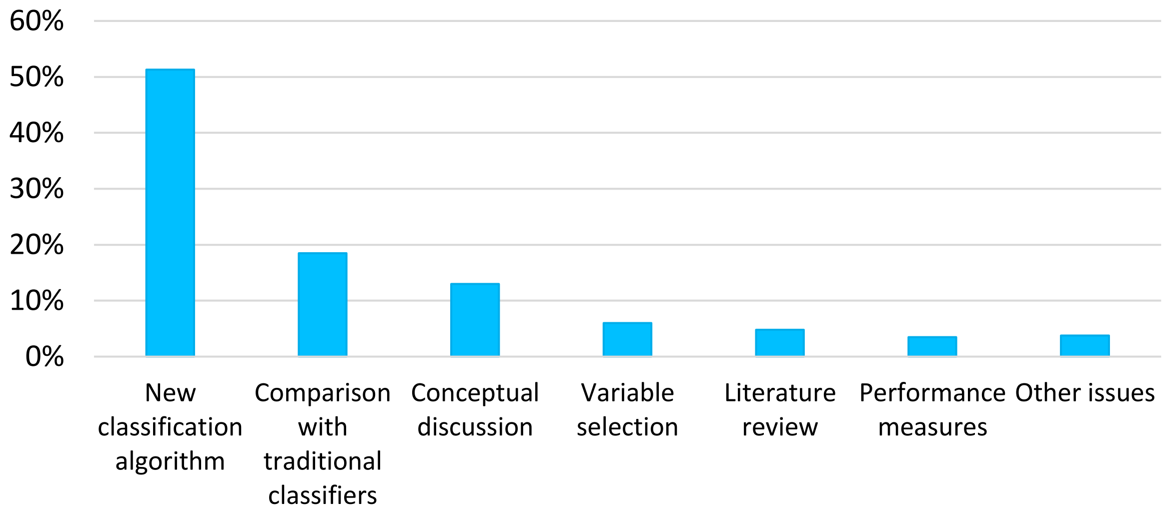

Figure 3. Percentage of papers published on the topic in 1992-2015.

Note: Adapted from Louzada et al. (2016).

While most credit scoring papers focus on the classification algorithm development, feature selection techniques or performance measures (according to Louzada et al., 2016), the problem of the acceptance threshold optimization for a credit scoring model has not received that much attention yet. The latter, however, is an essential step in credit scoring since it connects the classification algorithm itself, its predictive performance and economic outcome in the form of credit volumes and resulting profits. A possible reason why the topic is usually omitted from the discussion is the lack of access to real business data which is essential when choosing the optimal cutoff point (Lessmann et al., 2015). On the other hand, most of the practical issues that appear when one tries to implement a credit scoring model in practice are closely related to the acceptance threshold optimization problem.

One of the most discussed conceptual issues in the credit scoring literature is the error minimizing nature of standard credit scoring models rather than optimization of real business objectives. Originally, the training objective of classification models and cutoff point determination is to minimize the misclassification error treating costs of type I and type II errors as equal ($c_B = c_G$ in Table 1). However, it is a well-known fact that in credit industry an accepted bad borrower costs much more than a rejected good client (Lessmann, 2015). There is a set of papers (Viaene and Dedene, 2005; Hand, 2009; Verbraken et al., 2014) that investigate this issue and propose a solution in the form of cost-sensitive optimization that aims to maximize expected profit (so called profit scoring).

As a baseline approach to cost-sensitive cutoff point optimization we consider a method applied by most reviewed papers in profit scoring. The general framework starts by defining the average misclassification cost per loan application as follows (the following derivations are adapted from Viaene and Dedene, 2005; Hand, 2009; Oreski et al., 2012; Lessmann et al., 2015):

$$MC(t^{AT};\, c_B, c_G) = c_B \pi_B^{PP}(1 - F_B(t^{AT})) + c_G \pi_G^{PP} F_G(t^{AT}), \tag{2}$$

where t^{AT} – acceptance threshold, c_B – average cost per misclassified bad application reflecting the Type I error cost, c_G – average cost per misclassified good application reflecting the Type II error cost, π_G^{PP} and π_B^{PP} – prior probabilities of being a good and a bad application respectively and $F_G(t^{AT})$ and $F_B(t^{AT})$ – cumulative density of the scores up to cutoff score t^{AT} for good and bad applications respectively. Next, we minimize the expected misclassification cost per borrower with respect to cutoff value:

$$\min_t MC(t^{AT};\, c_B, c_G) = MC(T^{AT};\, c_B, c_G), \tag{3}$$

with T^{AT} being the optimal acceptance threshold defined as follows:

$$T^{AT} = \underset{t}{\operatorname{argmin}}\, MC(t^{AT};\, c_B, c_G) \tag{4}$$

The optimal cutoff point satisfies the first order condition:

$$\frac{f_B(T^{AT})}{f_G(T^{AT})} = \frac{\pi_G^{PP}}{\pi_B^{PP}} \frac{c_G}{c_B} \tag{5}$$

Thus, the optimal threshold is derived from prior probabilities of the classes, their score density functions and estimated average misclassification costs or their inverse, average expected profit (see an illustration of the traditional cost-sensitive cutoff optimization in Figure 4).

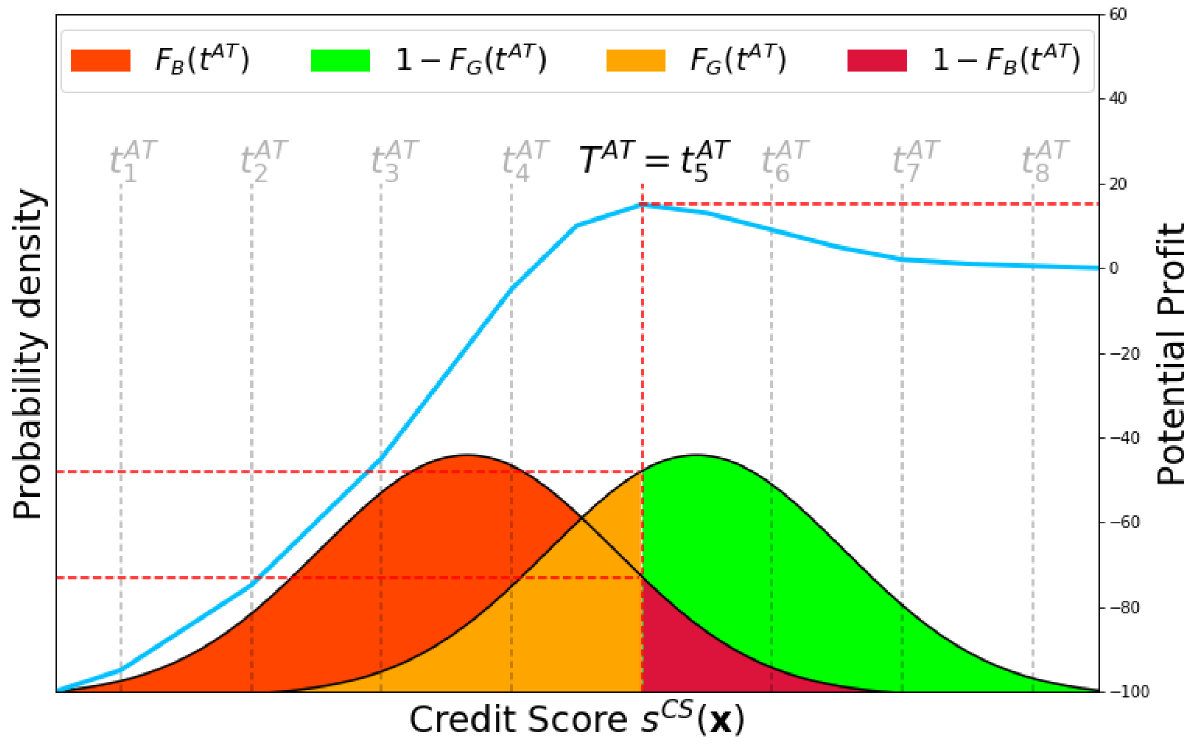

Figure 4. Traditional cost-sensitive acceptance threshold optimization.

Notes: $s^{CS}(\boldsymbol{x})$ – application's credit score estimated based on the application data $\boldsymbol{x}$**;** $f_G(s^{CS})$ and $f_B(s^{CS})$ – credit score's probability density functions of actual good and bad applications respectively; t^{AT} – acceptance threshold for the credit score; $F_B(t^{AT})$ – correctly classified bad applications; $1-F_G(t^{AT})$ – correctly classified good applications; $1-F_B(t^{AT})$ – bad applications misclassified as good ones; $F_G(t^{AT})$ – good applications misclassified as bad ones; blue line is the estimated potential profit (in thousands of euros for illustration purposes); grey dotted lines show alternative acceptance thresholds $t_i{}^{AT}$ and corresponding levels of potential profit; vertical red dotted line is the estimated optimal acceptance threshold T^{AT}, while horizontal red dotted lines show the corresponding potential profit and shares of correctly classified and misclassified good and bad applications.

As far as we know, most of the more advanced approaches to choosing the acceptance threshold (Verbraken et al., 2014; Skarestad, 2017) are built on top of this general framework, which however has some significant flaws. Although it might solve the problem of profit maximization for a particular application dataset, the assumption that the misclassification costs are accurately known and depend on the class only rather than on an individual example is unrealistic in the real world of consumer credit (Hernandez-Orallo et al., 2011; Sousa et al., 2013). Besides, in practice, the lender's utility function usually depends not only on profit but on other factors as well, such as credit volumes (Oliver and Thomas, 2009; Dey, 2010). Last but not least, this kind of optimization assumes that the dataset the model was trained on and the cutoff was optimized for is going to be identical to the real (live) data generation process of loan applications, which is not the case in credit scoring due to selection bias and population drift.

This brings the discussion to the issue of the sample selection bias (Eisenbeis, 1978; Hand, 2006; PAKDD, 2009). A credit scoring model is usually built using a pre-filtered set of data on accepted loan applications and their outcomes. This kind of sample is a biased representation of a general population since it excludes loan applications that were rejected by the lender and for which the outcome cannot be known. Thus, any model trained on as well as any cutoff point optimized for the sample of accepted applications will be biased[7] (see illustration on the Figure A-1 of Appendix A). Thomas et al. (2017) present a set of solutions most widely used in credit scoring to cope with sample selection bias that includes extrapolation and augmentation methods to account for the rejected cases (so-called reject inference). Banasik et al. (2003) apply Heckman selection approach widely used in econometrics and compare models trained on accepted sample only, accepted sample accounting for selection bias and the whole sample of accepted and 'would've been' rejected clients. In both papers, however, authors conclude that reject inference only slightly improves the accuracy of the model meaning that the cutoff point optimized on the existing data would still be suboptimal for the live data. Moreover, as was stressed by Wu and Hand (2007) and later by Dey (2010), the bias increases when the ratio of rejected to accepted clients rises, which is the case for consumer credit industry that targets subprime high-risk customers.

Another important issue that influences a credit scoring model's performance in practice is so-called population drift. As noted by Nikolaidis (2017), most scoring models assume the relationship between the estimated probability of default and the subsequent performance of a customer to be the same as it was for the data the model was trained on. However, the actual data generation process has a dynamic nature and depends on the changes in the general population, economic environment or market conditions. Thus, another branch of credit scoring

[7] A selection bias example from the company's practice is loan applications of unemployed people. The firm's rules state that unemployed people's applications can be accepted only if they are considered low-risk according to other characteristics, such as client's positive credit history with the company, healthy bank account history, no debts according to credit bureaus etc. It means that only unemployed clients' applications with the lowest default risk level will be accepted and most of them will pay in time. If a credit scoring model is then trained on a dataset of accepted applications, it might learn that unemployed clients' applications have relatively low risk to go overdue and will assign them a higher credit score than others. In the general population of (both accepted and rejected) loan applications, however, unemployed clients have much higher probability to go overdue on their loans, meaning that the trained model would be biased. The same issue applies to any kind of policy rules used by the lender (rejection of clients with any amount of debt, with no credit history, with gambling transactions) and it is hard to account for every one of them when building a credit scoring model. Thus, all those biases add up distorting the scoring model's performance and shifting the optimal credit score acceptance threshold for the live data.

research is aimed at the inclusion of economic dynamics into the credit scoring model (Crook et al., 1992; Bellotti and Crook, 2013). Since the population drift is usually assumed to affect the whole population of clients rather than some particular groups (Sousa et al., 2013; Bellotti and Crook, 2013), it does not bias the classification function itself, but it shifts the overall default rate and as a result, the optimal acceptance threshold of the model (see illustration on the Figure A-2 of Appendix A). This is evidenced by Sousa et al. (2013) who find that adjusting scores for systemic risk change in the form of central tendency of default is the best improvement for their scoring model. Finally, Thomas (2010) makes an important point that in consumer credit business, the score needs to have dynamic nature and ability to quickly respond to changes in economic and market behavior and to immediate changes in borrower behavior and circumstances.

The discussion above highlights a significant research gap in the cutoff point optimization methodology in credit scoring. There is a clear need for a system that could dynamically adjust the acceptance threshold in response to the changes in the live data generation process maximizing a set of business objectives. To the best of our knowledge[8], no research has been done yet to develop an adaptive system like this. In this paper, we propose a novel approach to the problem developing a dynamic decision support system based on the reinforcement learning algorithm.

2.2. The Essence of Reinforcement Learning

The reinforcement learning is a rather recent but intensively developing branch of machine learning theory. For the basics of RL discussed from now on we would refer to a textbook by its founding fathers, Sutton and Barto (2017). RL algorithms (or RL agents) are designed to optimize decision-maker's actions in the environment that could be described as a Markov Decision Process (MDP). The learning and optimization could be thought of as a natural learning process of a person (see Figure 5). Let's assume that the agent is designed to optimize acceptance threshold of a credit company. Then it operates in a credit business environment that generates a state (or observation) for a particular point in time, S_t, which describes current characteristics of the credit portfolio, market conditions, economy, assumed to be reflected in the loan applications acceptance rate. The state characteristics are transformed into a set of features (or variables) that could be understood by the agent. Based on the current state

[8] According to the search in Google Scholar and Science Direct databases with various combinations of keywords: 'credit scoring', 'dynamic', 'adaptive', 'threshold', 'cutoff', 'optimization', 'adaptation', there are no papers investigating a similar problem as we do.

description, current policy[9] and some prior knowledge (if available) the agent takes an action, A_t, i.e. adjusts the cutoff point of the credit scoring model. In the next point in time, based on this action (and previous actions) the environment generates a new state, $S_{t+1} = S(S_t, A_t)$. Additionally, it generates a reward, R_t[10], (e.g. firm's profit in the current period), which works like a feedback mechanism for the agent. From the reward, the agent learns how good were its previous actions and adjusts its actions trying to make them more rewarding or more optimal.

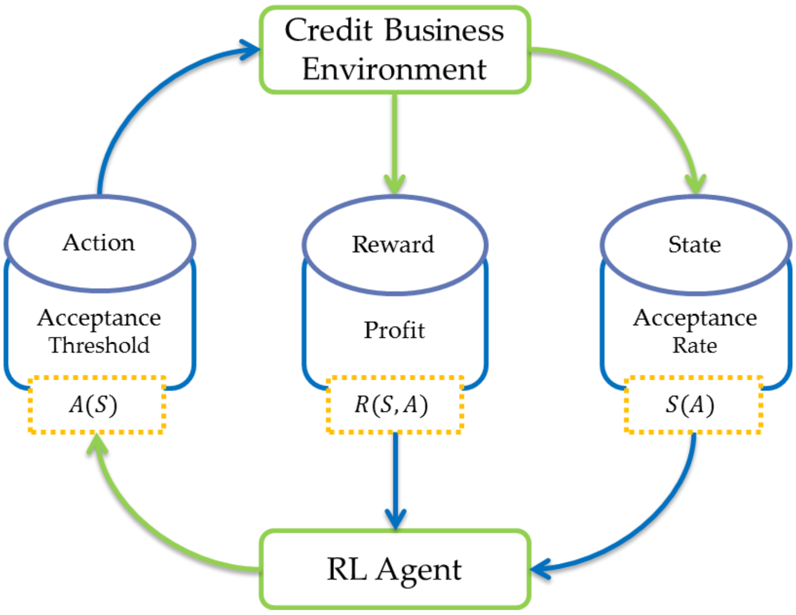

Figure 5. General reinforcement learning algorithm.

Notes: The figure shows the scheme of the interaction loop between the environment and reinforcement learning agent. The loop reflects state – action – reward exchange between the two.

The essential entity that the agent needs to learn from the interaction with the environment is thus the link between the state characteristics and the most appropriate action in this kind of state. This link is called the value function. State value function describes an expected discounted reward of being in a state s and following a policy π (also known as the Bellman equation):

$$V_\pi(s) = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots | S_t = s], \tag{6}$$

where γ is a discount rate. Similarly, one can define the action value function (also called Q-function) that describes an expected discounted reward of taking action a in a state s and following a policy π thereafter as follows:

[9] Policy is a function mapping from state to action which could be defined as deterministic: $a = \pi(s)$, or stochastic: $\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$.
[10] Even though the reward is generated in period $t + 1$, there is a convention to denote it as R_t, since it reflects the reward for the action taken in period t.

$$Q_\pi(s,\, a) = \mathbb{E}_\pi[R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots | S_t = s,\, A_t = a,] \tag{7}$$

In case the Q-function can be accurately approximated by the agent, the optimal policy then simply becomes:

$$\pi^*(s) = \underset{a}{\operatorname{argmax}}\, Q_\pi(s,\, a) \tag{8}$$

RL algorithms that aim to learn the action value function are called Q-learning techniques. Basically, all the agent needs to do to learn the true value function is to correct its learned value function in each period according to the following update rule:

$$Q(S_t,\, A_t) = \; Q(S_t,\, A_t) + \alpha[R_t + \gamma \max_a Q(S_{t+1},\, a) - Q(S_t,\, A_t)], \tag{9}$$

where α is the learning rate parameter. Watkins (1989) proves that Q-learning converges to an optimal policy if all the state-action pairs are visited an infinite number of times. Sutton and Barto (2017) note that in some cases the convergence could be achieved surprisingly fast in just a couple of iterations. Besides, there exist a number of improvements to the original Q-learning, like Speedy Q-learning by Azar et al. (2011) or Zap Q-learning by Devraj and Meyn (2017) that help to speed up the convergence of the algorithm.

It is important to note, however, some of the flaws in Q-learning. One well-known problem in RL applications is so-called curse of dimensionality due to Bellman (1957). The problem is that in case of discrete state spaces[11], the computational requirements grow exponentially with the number of state variables meaning that the implementation could be feasible only for small state spaces. The widespread solution is to consider continuous state spaces together with a more sophisticated value function approximation method. For instance, in the survey of reinforcement learning Kaelbling et al. (1996) describe different function approximation techniques that use simple stochastic gradient descent[12], decision tree, or artificial neural networks[13]. Some outstanding examples of using advanced value function approximators are shown in the Table A-2 of Appendix A.

Another drawback is the tendency of Q-learning algorithms to overestimate Q-values coming from the nature of the update rule that is based on the maximum Q-value (so-called

[11] A state space is the set of all possible configurations within the environment. A discrete state space can be described by discrete variables. For instance, in chess the state space could be described by $\binom{64}{32}$ binary variables (where 64 is the number of chess board positions and 32 is the number of chess pieces).
[12] Gradient descent is a first-order iterative optimization algorithm used to find optimal parameters of a model that minimizes its approximation error. Gradient descent methods are called stochastic when the optimization update is done on only a single example per iteration, which might have been selected stochastically. For a more detailed explanation refer to Kaelbling et al. (1996), Sutton and Barto (2017).
[13] An artificial neural network is a computational model based on the structure and functions of biological neural networks used for non-linear function approximation. For a more detailed explanation refer to Kaelbling et al. (1996), Sutton and Barto (2017).

maximization bias[14]). This issue was investigated carefully by Sutton and Barto (2017) and van Hasselt et al. (2016) and an alternative approach of double Q-learning was proposed[15]. Last but not least, using continuous action space[16] adds computational and mathematical complexity, that's why discrete action spaces are preferred in RL applications. On the other hand, discrete action space removes the link between similar actions making the agent lose significant information about the process. This problem was studied by Wang et al. (2015) and a solution in the form of dueling deep Q-networks was developed[17].

2.3. Finance and Business Applications of Reinforcement Learning

Modeling credit process as an MDP is far from new, however it has been mainly used to optimize credit limits (So, 2009; So and Thomas, 2017), credit prices (Trench et al., 2003), collection policies (Briat, 2006) or as a credit scoring system itself (Malik and Thomas, 2010; Regis and Artes, 2015). Nevertheless, to the best of our knowledge, there has been no attempt to neither use MDP to optimize acceptance threshold nor use RL algorithms in consumer credit business yet. Moreover, MDPs were always used by big financial institutions with a well-established scoring system and amount of behavioral data sufficient to accurately estimate transition probabilities, while in our case the MDP and optimal policy are going to be learned online.

A financial area where RL algorithms were adopted for quite some time is portfolio optimization (Neuneier, 1996; Gao and Chan, 2000; Lee, 2001; Moody and Saffell, 2001). We consider it to be close to the cutoff optimization task since the latter can be thought of as a portfolio optimization of different risk groups of loan applications. More examples of rather new RL applications come from the business management field. Some of these studies are briefly described in the Table A-3 of Appendix A. The main ideas from the RL applications review can be summarized in the following way:

- RL methodology was found effective in portfolio optimization problems (Neuneier, 1996, Du et al., 2016, Strydom, 2017);

[14] In Q-learning algorithms a maximum over estimated values is used implicitly as an estimate of the maximum value, which can lead to a significant positive bias in Q-values. See Sutton and Barto (2017) for detailed explanation.

[15] Double Q-learning RL algorithm uses one action value function to determine a maximizing action and a different one to estimate its value. It was proven to solve the problem of double maximization and assure unbiased Q-values. See Sutton and Barto (2017) and van Hasselt et al. (2016) for more information.

[16] An action space is the set of all possible actions in a decision-making problem.

[17] Dueling deep Q-network RL algorithm uses one neural network to estimate state value function and another network for the state-dependent action advantage function. It lets the agent generalize learning across actions without imposing any change to the underlying RL algorithm. See Wang et al. (2015) for more information.

- RL algorithms are able to solve optimization problems with little or no prior information available about the environment (Tesauro et al., 2006; Kim et al., 2016);
- It is possible to avoid suffering potential costly poor performance in live online training by letting the RL agent learn offline, following safe expert-defined policy or train in a simulated environment (Tesauro et al., 2006; Aihe and Gonzalez, 2015);
- The RL agent learning from the real-time data optimizes more accurately and efficiently than traditional approaches based on theoretical modeling (Rana and Oliveira, 2015; Strydom, 2017);
- A reinforcement learning system is able to satisfy contradictive performance goals outperforming reasonable heuristic or hand-coded approaches (Huang et al., 2010; Varela et al., 2016);
- RL mechanisms outperform other intelligent systems in their ability to dynamically adjust the policy over the learning period adapting to environmental changes (Abe et al., 2010; Rana and Oliveira, 2015, Strydom, 2017);
- Using artificial neural networks in the value function approximation for RL solves the curse of dimensionality and provides faster learning (Moody and Saffell, 2001; Darian and Moradi, 2016).

Finally, it is worth noting that lots of papers employ a Monte Carlo simulation to train and test the RL agent (Huang et al., 2010; Darian and Moradi, 2016; Sato, 2016; Strydom, 2017). The main reasons are to avoid costly learning errors in the real environment, perform stress-testing and compare algorithm performance to the one produced by traditional policies. It is important to remember, however, that in this case, the actual performance of optimal policy is highly dependent on the accuracy of the simulation. While the simulation is just a simplified approximation of reality, the adaptive nature of the RL agent lets it correct for the bias and inaccuracies when operating in the real environment.

Summing up the discussion above, the main benefits of RL compared to classic dynamic programming and traditional optimization techniques are that the former does not need the exact knowledge of the MDP structure and that it has a dynamic nature, i.e. is able to adapt to changes in the environment. The objective function that the RL agent aims to maximize or minimize is designed to reflect the preferences of the decision-maker. Thus, in case of the acceptance threshold optimization for credit scoring, this kind of system can adapt to the changes in the population dealing with selection bias and population drift starting with little prior information (e.g. results from the test dataset) and it can be customized to optimize the accurately specified objective function of the lender.

3. Methods and Data

The following section describes the methodology and the data used to build the reinforcement learning system. In particular, it formulates the acceptance threshold optimization problem as a reinforcement learning task, describes the components of the constructed RL algorithm and the interrelation between them, explains the mathematical modeling details behind the algorithm, discusses the nature of the data used to train the system and lays out the structure of the training and evaluation experiments.

3.1. Problem Setup and Data Generating Process Description

The scheme of the RL algorithm is shown in Figure 6. As was explained before, the RL algorithm consists of two major entities: the environment and the agent, which interact with each other by exchanging state, action and reward objects (see the outer loop in Figure 6). The environment was created using a Monte Carlo simulation (from now on referred to as the simulation) built based on the historical data of an international consumer credit company Creditstar Group for one of its target markets. The firm specializes in consumer loans of sums in a range of 24-1200 EUR for durations in a range of 5-90 days with no collateral. We further explain the general idea behind the simulation so that it is sufficient for the reader to follow, how the aim of the thesis is reached. Due to confidentiality reasons, we do not disclose all the details and parameters of the simulation.

The simulation generates weekly data based on the historical period of 24.05.2015 till 09.07.2017 (112 observations). In each simulated week (from now on week or iteration)[18] it starts by generating the numbers of new and repeat customers' loan applications received. Next, for each application, it generates loan sum and duration, potential profit, whether the loan is going to go overdue and whether it is going to pay back after the collection procedure in this case. Based on these variables the event dates are calculated (maturation date, overdue date, delinquent repayment date). Finally, for each loan application, the company's credit scoring model prediction is generated[19].

Next part of the simulation marks the accepted applications based on the company's acceptance threshold for the application credit score (see an example of a weekly generated dataset in Table B-1 of Appendix B). This data is used to calculate major loan portfolio characteristics, such as

[18] Simulated 'week' and 'iteration' terms are used interchangably.
[19] The credit scoring model is an ensemble of binary and clustering classifiers trained on historical loan application data using more than 300 uncorrelated variables. It outputs an inverse of the loan default probability adjusted for firm's policy rules (referred to as the credit score).

application volumes, new-to-repeat application ratios, acceptance rates, default rates, profits (see an example of generated time series in Figure B-1 of Appendix B).

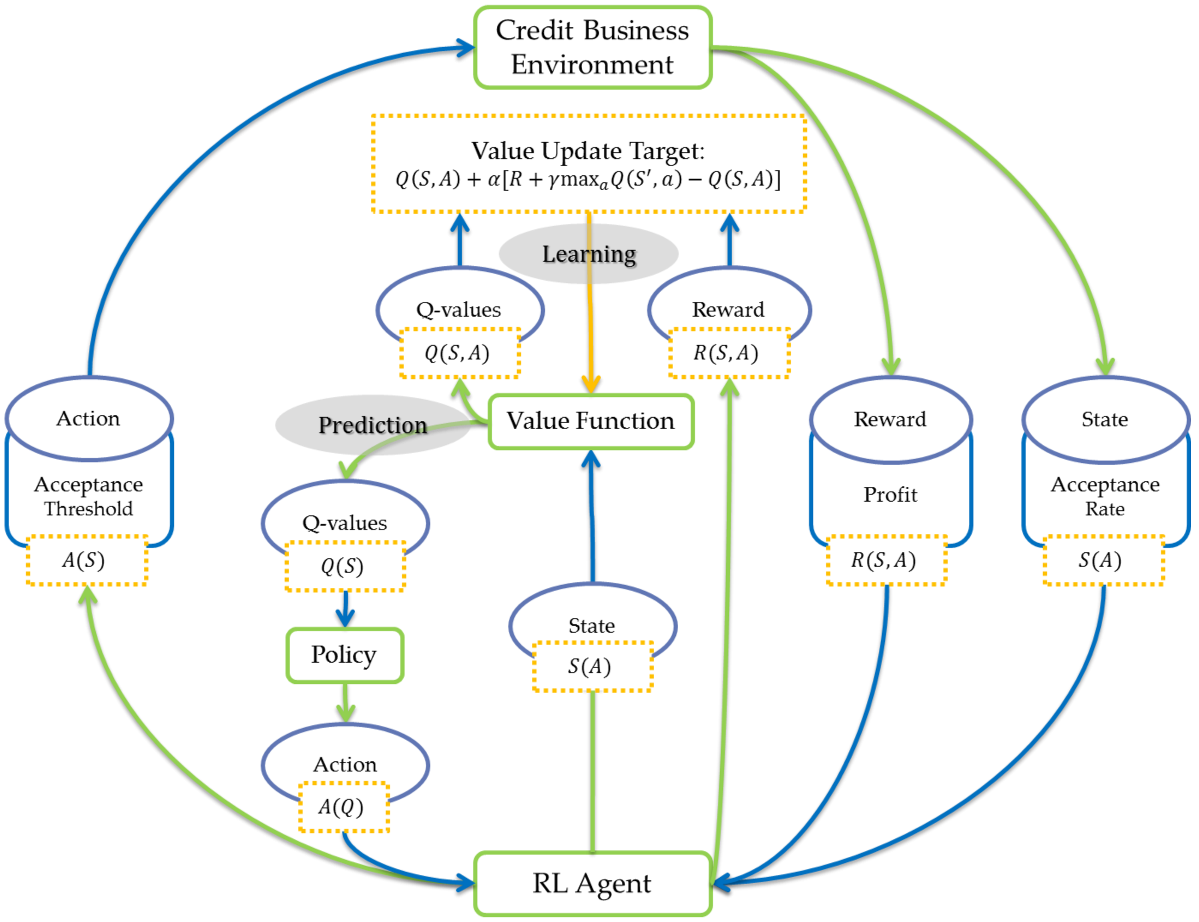

Figure 6. The scheme of reinforcement learning algorithm for a credit scoring acceptance threshold optimization in a consumer credit company.

Notes: The figure shows the scheme of the interaction loop between the environment (the simulation) and reinforcement learning agent. The outer loop reflects state – action – reward exchange between the two. The inner loop shows the state – action evaluation and learning from reward by the RL agent. S – state or weekly acceptance rate, A – action or acceptance threshold, Q – action-state value prediction, R – reward or profits, α – learning rate, γ – discount parameter, S' - following state.

We define the decision process state as weekly application acceptance rate variable[20] ($S(A)$ in Figure 6). After experimenting with various sets of variables as state definition it was concluded that using solely acceptance rate variable is the best for the current problem. First, keeping the state dimensionality as small as possible lets the algorithm converge faster and more accurately, makes results more interpretable and easier to debug, thus making it a better choice for the initial version of the model. Second, compared to all the other loan portfolio characteristics acceptance rate has a direct connection with the others. For instance, it defines the contingency matrix values (as well as derived from it type I and type II errors) together with corresponding

[20] 'State' and 'acceptance rate' are used interchangeably.

credit volumes and thus, embodies a direct link between the acceptance threshold and the resulting overall profit. Third, since acceptance rate is defined by the acceptance threshold it will be easier for the RL agent to plan and interpolate its policy to other states once again letting the algorithm to converge faster and more confidently. Finally, the acceptance rate is stationary and lies between 0 and 1 as opposed to profits or accepted applications number variables, which makes it more suitable for the MDP state definition.

The action space of the decision process is defined with the acceptance threshold variable[21] ($A(S)$ in Figure 6). Because of the reasons discussed in the literature review part of the work, we use discrete action space rather than continuous one. The agent has a choice of 20 binary actions reflecting credit score acceptance thresholds from 5 to 100 evenly spaced by 5. The reward is defined as the weekly profit value[22] ($R(S, A)$ in Figure 6). The reward variable thus accounts for the actual distribution of the misclassification costs without drawing any simplifying assumptions. Since we use aggregate profit rather than profit per loan, the reward also takes into account credit volumes.

The RL agent is designed as an entity separate from the environment. The only links it has with the environment are: 1) it uses environment's state object (acceptance rate value) as input to decide on the action; 2) it outputs the action (acceptance threshold value) to be used by the environment to calculate loan portfolio characteristics; 3) it uses environment's reward object (profit value) to learn the value function; 4) it uses state history to calculate 'would be' rewards from alternative actions (to be discussed later). Thus, its information set is included in and, in fact, is significantly smaller than the actual decision maker's information set.

3.2. Reinforcement Learning Agent Specification

To perform the optimization task the agent uses a value function model and a set of policies (see the inner loop in Figure 6 and Figure 7). These are defined as was discussed in the literature review section. After experimenting with various function approximation models, such as simple stochastic gradient descent (SGD), polynomial SGD, simple artificial neural networks (one to three fully connected layers), it was decided to use a set of Gaussian Radial Basis Functions (RBFs)[23] as a value function approximator. The latter represents a compromise

[21] 'Action' and 'acceptance threshold' are used interchangeably.

[22] 'Reward' and 'profit' are used interchangeably.

[23] A radial basis function is a real-valued function whose value depends only on the distance from the origin or some other point c, called a center, so that $\varphi(x, c) = \varphi(\|x - c\|)$. Sums of radial basis functions are typically used as function approximators. A Gaussian radial basis function is $\varphi(r) = e^{-(\varepsilon r)^2}$ where $r = \|x - x_i\|$. The intuition behind the transformation is actually quite simple. Similar to creating a polynomial transform, when one takes nth power of input variables resulting in additional polynomial features in order to account for non-linear form of the relation in a linear regression model, Gaussian RBFs produce features that reflect the non-linear form

between simplicity and flexibility needed for the problem. Compared to simpler gradient descent algorithms, RBFs provide a more flexible shape of the approximated function and fit the expectation that the value function is bell-shaped both in states and actions. As opposed to complex artificial neural network models, RBFs converge faster, are much easier to set up and work with and are computationally cheaper. Finally, RBFs have been found to perform well in small state spaces (Lane et al., 1992).

We use the RBFs transformation function as implemented in the Python library Scikit-learn based on Rahimi and Recht (2008). It can be summed up in a transformation of type:

$$x = \frac{\sqrt{2}}{\sqrt{k}}\cos(w^{RBF}s + c^{RBF}),\, w^{RBF} \sim N\left(0, \sqrt{2\gamma^{RBF}}\right), c^{RBF} \sim U(0, 2\pi), \tag{10}$$

where x is the resulting transformed feature vector, s is the input state variable, k is the number of Monte Carlo samples per original feature, w^{RBF} is a k-element vector of randomly generated RBF weights, c^{RBF} is a k-element vector of randomly generated RBF offset values and γ^{RBF} is the variance parameter of a normal distribution. We employ four Gaussian RBFs with γ^{RBF} parameters equal to 5, 2, 1, 0.5 and numbers of components k equal to 500 each.

Thus, transforming a state variable with RBFs generates 2000 features that are later normalized and used in Stochastic Gradient Descent models. The latter can be thought of as a simple linear regression model of the form:

$$Q(w_a, s) = w_a RBF(s) = w_a x, \tag{11}$$

where w_a is a vector of regression weights for action a, s is the state variable, RBF is the RBFs transformation function, x is the resulting vector of features and Q is the value of action a in state s corresponding to the feature vector x.

There are 20 SGD models, one for each action. Each model takes the generated vector of features as input and outputs a value of the corresponding action ($Q(S)$ in Figure 6), which leaves the agent with 20 action values for the current state. At this point, the application of the value function model is finalized and the agent can choose an action based on the current policy ($A(Q)$ in Figure 6). The policy differs depending on the agent's regime. In the test regime the agent just exploits the value function model it learned so far by taking the most rewarding action (so called 'greedy' policy):

$$\pi^{Greedy}(s) = \underset{a}{\operatorname{argmax}}\, Q(s, a) \tag{12}$$

of the Normal distribution. Using several RBFs with various scaling parameters lets the approximation function have more flexible non-linear form. For more information refer to Buhmann (2003).

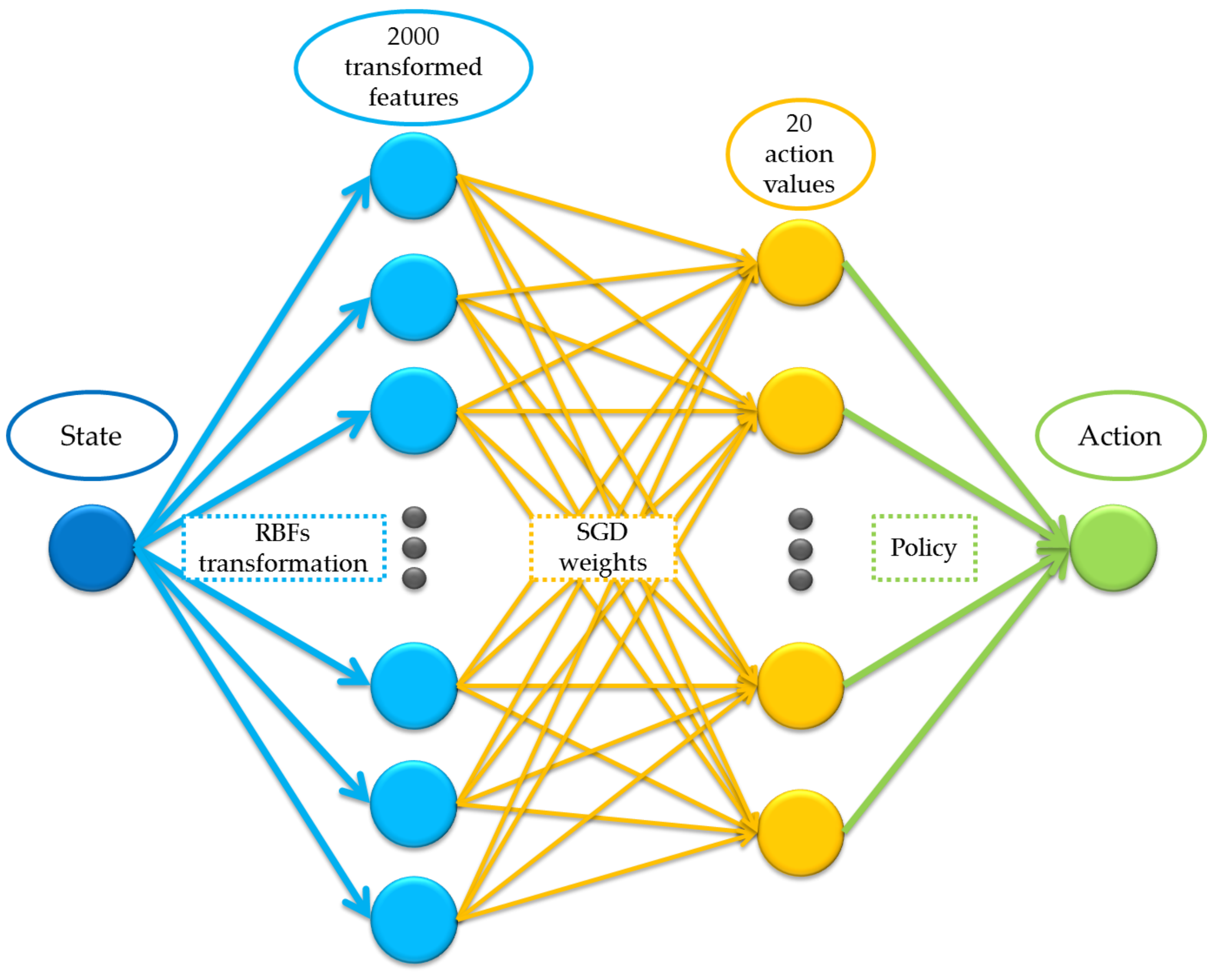

Figure 7. Value function model and policy architecture of the reinforcement learning agent.

Notes: The figure shows the steps of state (acceptance rate) transformation into action (acceptance threshold) using value function model and a policy. Radial basis functions (RBFs) transform the initial state value into 2000 normally distributed features. Those are used as input to 20 stochastic gradient descent (SGD) models. The latter generate a Q-value for each of the 20 actions. Values are then used according to the policy (greedy or Boltzmann-Q) to choose the final action (acceptance threshold).

In the training regime the agent needs to explore the environment, for which we use the Boltzmann-Q policy defined as follows:

$$\pi^{Boltzmann}(a|s) = \mathbb{P}[A_t = a \mid S_t = s] = \frac{e^{\frac{Q(s,a)}{\tau}}}{\sum_{a' \in \mathcal{A}} e^{\frac{Q(s,a')}{\tau}}}, \tag{13}$$

where $\mathcal{A}$ is the set of all actions, a' is any action except action a and τ is the temperature parameter of the Boltzmann distribution. As τ goes to infinity, the probability of taking any action becomes the same:

$$\mathbb{P}[A_t = a \mid S_t = s] = \frac{1}{\mathcal{A}} \, \forall \, a \in \mathcal{A}, \tag{14}$$

while as τ approaches zero, the probability of taking the most rewarding action increases up to 1, turning it into a ‘greedy’ policy (see illustration in Figure B-2 of Appendix B). Thus, τ regulates the explorative behavior of the agent. The goal is to set τ in a way that at first, the agent explores the environment learning the value function model, but as the latter starts converging to the optimum, the agent takes more rewarding actions and visits more rewarding states with a higher probability, exploring mainly near-optimal states and actions.

While the feedforward mechanism of the agent is quite simple, the feedback part where the learning happens is more complicated. Once agent chooses an action and sets the acceptance threshold for the following week, the applications with the credit score higher than the chosen cutoff are considered accepted by the environment. However, the reward for the action taken is revealed only when a corresponding loan pays back or defaults, which might happen months after the loan issue. Since loans vary in duration, not only the reward is delayed, but also distributed among various weeks. To keep the learning process smooth, we do the following: in each week we calculate weekly profit and divide it between the (week – action – state) tuples it corresponds to based on the loans that generated it. The segmented reward is then passed to the agent and a separate value function update is performed for each (state – action – reward – following state) tuple (value update target in Figure 6).

Another peculiarity of our problem is that once the acceptance threshold was applied, the decision-maker can easily calculate profit for higher cutoffs since outcomes of applications with higher scores are revealed. Similarly, the state that follows any threshold can be accurately predicted based on the applications number and score distribution in the previous week and acceptance threshold of interest. Thus, the (state – action – reward – following state) tuple can be generated for any action higher than the actual one and additional value function updates can be done.

The value target is defined as in equation (9). Thus, the approximation error[24] is:

$$R_t + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t) \tag{15}$$

To adjust the SGD model weights in the direction of the steepest error descent we use the following update rule:

$$w_a \leftarrow w_a + \alpha[R_t + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \frac{\partial Q(s,a)}{\partial w_a}, \tag{16}$$

[24] Also known as Bellman residual.

which under the assumption that $\gamma max_a Q(S_{t+1}, a)$ does not depend on w_a[25] simplifies to the general SGD update rule:

$$w_a \leftarrow w_a + \alpha[R_t + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]S_t, \tag{17}$$

where $Q(S_t, A_t)$ can be thought of as current model prediction, $R_t + \gamma max_a Q(S_{t+1}, a)$ – the target and S_t – the gradient of the weights.

Given that the agent visits different states and uses various actions, and that the learning rate is decreasing[26], with every update, the value function model is expected to produce a better approximation than before meaning that the estimated value function will converge to the true value function of the problem (Sutton and Barto, 2017).

3.3. Experiment Structure

The structure of the training experiment is the following. We define an episode as a full simulation run. The historical data for the first 52 weeks corresponds to the company's entrance to a new market and has some irregular properties (such as intense marketing campaigns and operation interruptions) that we would not want the agent to learn. That's why we drop those from the learning phase. Thus, one episode consists of 60 weeks (simulated 420 days) and generates 60 states in response to the agent's actions. A training episode scheme is described in Figure 8.

The training algorithm of the RL agent follows the scheme in Figure 6. The loop shown in the figure repeats each simulated week, 82 times per episode. The algorithm can be described in the following way. Starting from the first week:

1. A state (weekly acceptance rate) is generated by the environment and passed to the agent.
2. Based on the passed state the Q-values of actions (acceptance thresholds) are predicted using the value function model.
3. An action (acceptance threshold for the following week) is chosen based on the predicted Q-values and the training (Boltzmann-Q) policy and passed to the environment.

[25] So called direct update rule. See Baird (1999) for more detail.

[26] To implement the decreasing learning rate we use the inverse scaling learning rate that follows the formula: $\alpha_t = \frac{\alpha_0}{t^{power\_t}}$, where α_t is a learning rate parameter at time t, α_0 is the initial learning rate and $power\_t$ is the inverse scaling paramter of the learning rate.

4. Next state (acceptance rate for the following week) is generated by the environment based on the chosen action. Rewards (profits) for all the previous weeks are generated by the environment. State and rewards are passed to the agent.
5. The value function model is updated based on revealed rewards, state, and action history and predicted (state – action – reward – following state) tuples.
6. Return to the point 2 and continue the loop until the end of the episode.

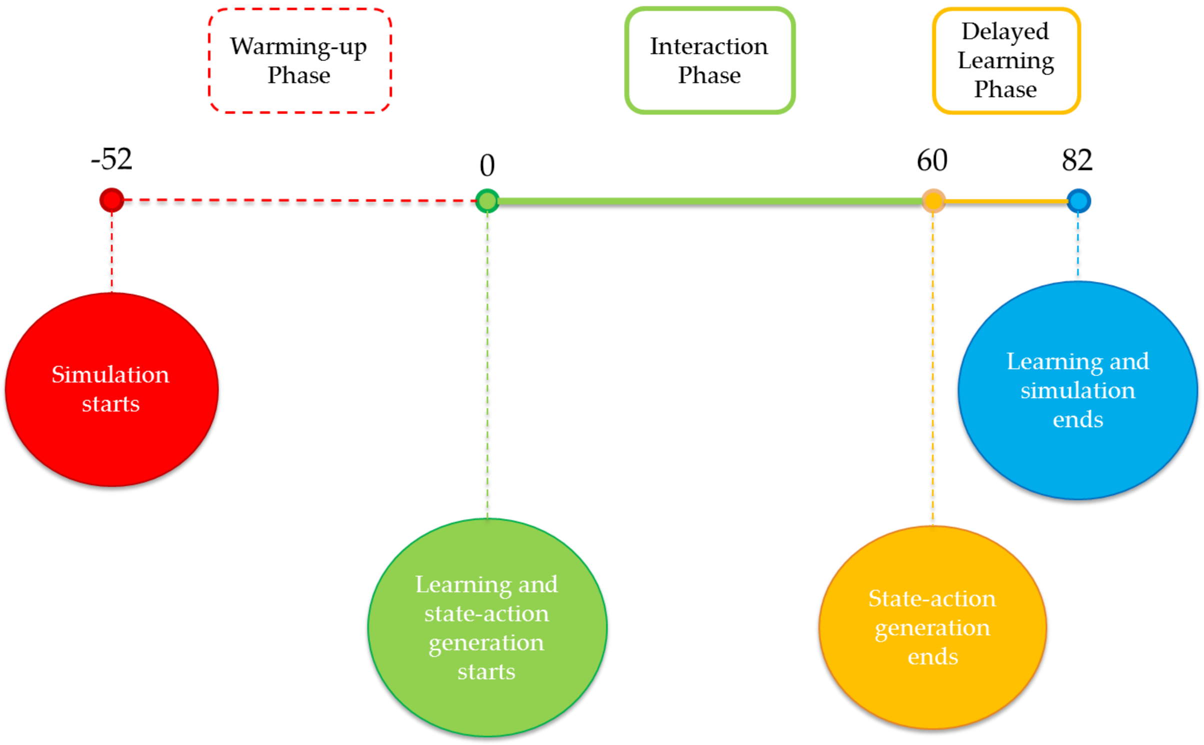
Warming-up Phase
Interaction Phase
Delayed Learning Phase
-52
0
60
82
Simulation starts
Learning and state-action generation starts
State-action generation ends
Learning and simulation ends

Figure 8. Training episode structure of the reinforcement learning system.

Notes: The figure shows the structure of a training episode. Out of 112 weeks generated by the simulation the first 52 are used as a warm-up with no interaction or learning. During the next 60 weeks, every week environment generates a state (weekly acceptance rate) and reward (realized profit), while agent responds by taking an action (acceptance threshold for the following week) and learning from the actual reward. During the last 22 weeks, the environment only generates delayed rewards for the loans issued during the interaction phase, and the agent learns by adjusting the value function model according to the realized rewards.

After the 60th week the environment no longer generates states, but still outputs delayed rewards for the agent to learn up until the 82nd week when outcomes of all the accepted loans are known and the episode rewards are completely defined.

The training experiment consists of 100 training episodes where the agent learns the value function model using training Boltzmann-Q policy. Every 5 training episodes a set of 5 test episodes is run where the agent uses greedy policy to track the performance improvement of the optimized policy. The parameters for the training experiment were optimized using grid search for the highest test episode rewards, the fastest convergence to the optimal policy and the most intuitive value function model shape (see sample grid search results for the learning rate optimization in Figures C-1 to C-3 of Appendix C). The resulting parameters are summarized in Table 2.

Table 2. Reinforcement learning agent parameterization for the training experiment.

Parameter	Notation	Value
Q-learning update rule discount rate	γ	0.95
SGD initial learning rate	α_0	0.0001
SGD inverse scaling parameter	$power\_t$	0.25
Boltzmann-Q policy temperature parameter	τ	1

In order to account for the dynamic nature of the real environment compared to the simulation and for potential inaccuracies of the latter we perform the test experiment. We adjust the simulation parameters to add distortions to the environment[27], such as score distribution or prior default probabilities shifts reflecting credit scoring model's performance uncertainty, population drift, and selection bias issues.

The test experiment consists of 100 episodes for each type of distorted environment. In each episode, the agent starts with the value function model learned in the training experiment and adapts to the distorted environment by interacting with it. Since in the test experiment the agent needs to be more responsive to be able to reshape the estimated value function consistently and less explorative to avoid extra costs, we adjust the RL parameters as denoted in Table 3.

Table 3. Reinforcement learning agent parameterization for the test experiment.

Parameter	Notation	Value
Q-learning update rule discount rate	γ	0.5
SGD initial learning rate	α_0	0.001
SGD inverse scaling parameter	$power\_t$	0
Boltzmann-Q policy temperature parameter	τ	0.5

Finally, to assess the agent's performance on the real data it has never seen before we let it operate with the real loan applications received since 10.07.2017 to 25.12.2017 (24 weeks) and scored with the credit scoring model used in the simulation[28]. To cope with the selection bias issue present in the data, low-scored application sampling was performed during this period, meaning that a percentage of low-scored applications were accepted in order to gather information on low-quality applications that would normally be rejected by the model. Based on outcomes of those applications we estimate the prior default rates for applications in low-score bins. Based on those we extrapolate the outcomes for all the rejected applications,

[27] Referred to as a 'distorted episode' from now on.

[28] We refer to this experiment as the 'real episode'.

getting a stream of data of the same structure as in the simulation but coming from the real environment.

We compare the results of the acceptance threshold optimization using the RL algorithm to the results of the traditional cost-sensitive cutoff optimization. The latter is implemented as described in the literature review section following the methodology used in Verbraken et al. (2014) and Skarestad (2017). We simply calculate the optimal threshold value using equation (5) based on the test dataset credit score distribution. From now on we refer to the traditional approach as the baseline[29].

4. Results

The following section presents the results of the experiment. It starts with showing the baseline results of the traditional approach to acceptance threshold optimization. It then describes the training process of the constructed reinforcement learning system. Next, it evaluates the RL algorithm's adaptive ability on various simulated scenarios and on the real recent data of the company and compares the results to the baseline.

4.1. Baseline Results and Reinforcement Learning Algorithm Convergence

We start with describing the results for the baseline threshold optimization approach (see Figure 9). If we consider the same action set as for the reinforcement learning problem (cutoffs from 5 to 100 by 5) and compute the potential profits for each action based on the test dataset for the current credit scoring model, we can see that actions below 60 lead to a large negative profit, while actions over 80 lead to near-zero profit. The potential profit function is hump-shaped with an optimal cutoff point of 65 (corresponding to the baseline action). One can notice that being too liberal is much more harmful than being too strict, as it usually happens in the credit industry[30] (Thomas et al., 2017; Lessmann, et al., 2015). The optimal acceptance rate calculated based on the test set is around 15%.

Turning to the RL optimization results, we first describe the value function model convergence dynamics during the training experiment. The timing details of the experiment are shown in Table C-1 of Appendix C. Since the learning rate of the RL agent has a decreasing schedule,

[29] An action chosen according to the baseline policy is referred to as the 'baseline action' or 'baseline acceptance threshold' and reward received performing the baseline action is referred to as the 'baseline reward' or 'baseline profit'. The difference between another action and the baseline action is referred to as the 'action difference' or 'acceptance threshold difference' and the difference between another reward and the baseline reward is referred to as the 'reward difference' or 'profit difference'.

[30] In credit scoring the cost of misclassifying a bad application is usually more than 5 times higher than the cost of misclassifying a good one, which becomes even more severe in case of consumer loan providers.

the major part of learning happens during the first episode of training (see Figure 10). The actual actions taken by the agent during the first episode and the action that the agent considers optimal according to the value function model together with received rewards for each simulated week are shown in Figures 11 and 12.

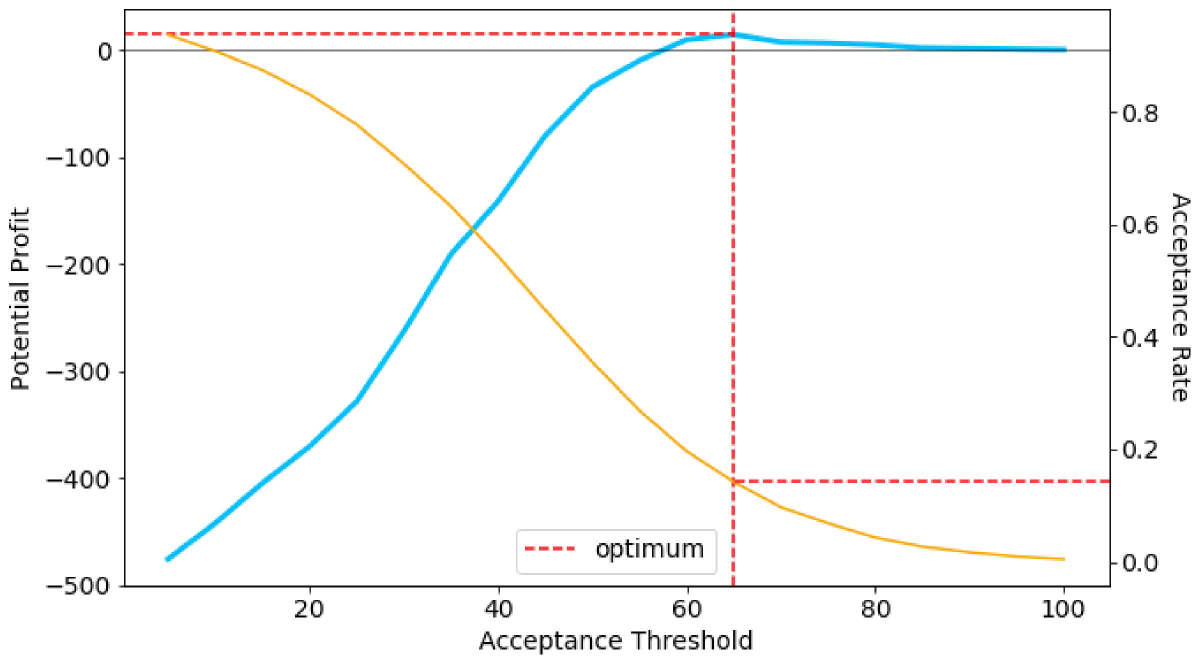

Figure 9. Baseline threshold optimization results: potential profit and corresponding acceptance rate for every acceptance threshold based on the test dataset

Note: Profit is measured in thousands of euros.

The agent starts without any knowledge about the environment (flat value function model shape) and performs random actions to explore it. Once it gets its first rewards for the actions taken, it starts adjusting the model making the value predictions more accurate. As the difference between values for various actions grows, the exploration rate decreases, and the agent’s action range slowly converges towards the optimum predicted by the value function model. In most cases the RL agent finishes the first episode with a relatively well-approximated value function model, meaning that the latter predicts optimal action to be no more than 10 points away from the baseline action. Expectedly, the rewards received are consistently lower than or around the baseline rewards due to the initial exploration of the environment. Consequently, the agent finishes its first training episode with a total profit significantly lower than in the baseline case.

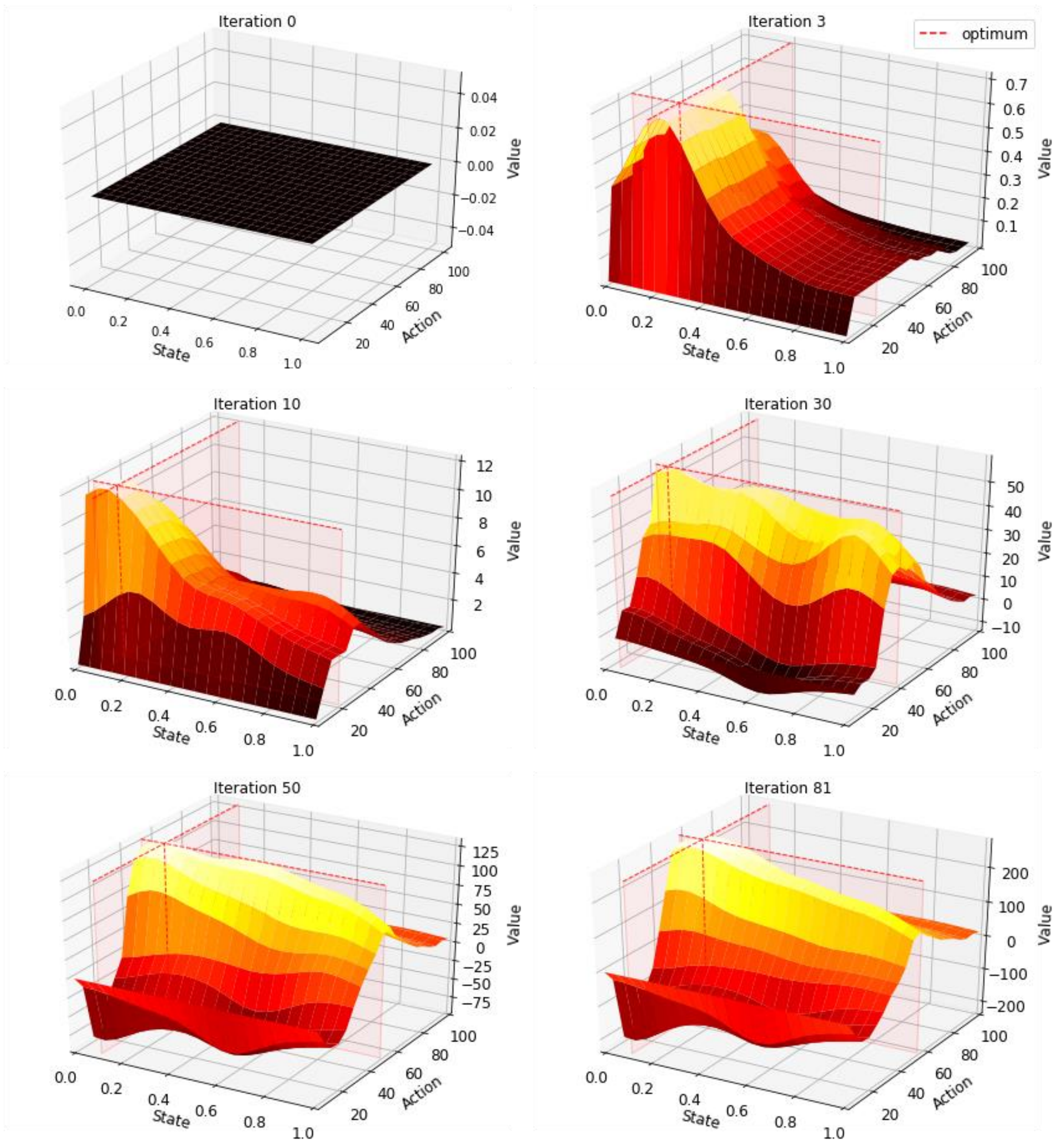

Figure 10. Value function model convergence during the first training episode

Note: State denotes the application acceptance rate during the previous week, action denotes the acceptance threshold for the following week, value is the prediction of the value function model for a particular state-action pair, optimum shows the state-action pair that corresponds to the highest value in the state-action space.

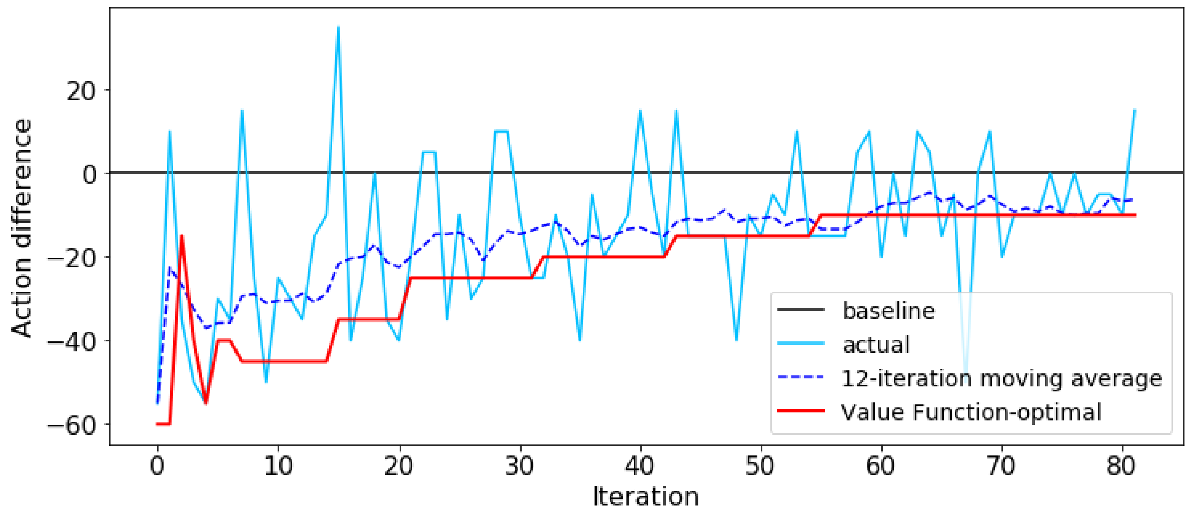

Figure 11. The dynamics of the action difference between the actual action taken and the baseline action, value function-optimized action and the baseline action during the first training episode

Note: the figure shows the difference between action variables and the baseline action during interaction and delayed learning phases. Baseline denotes the acceptance threshold optimized using the traditional approach, actual denotes the one used by the RL agent, value function-optimal denotes the one optimal according to the value function model.

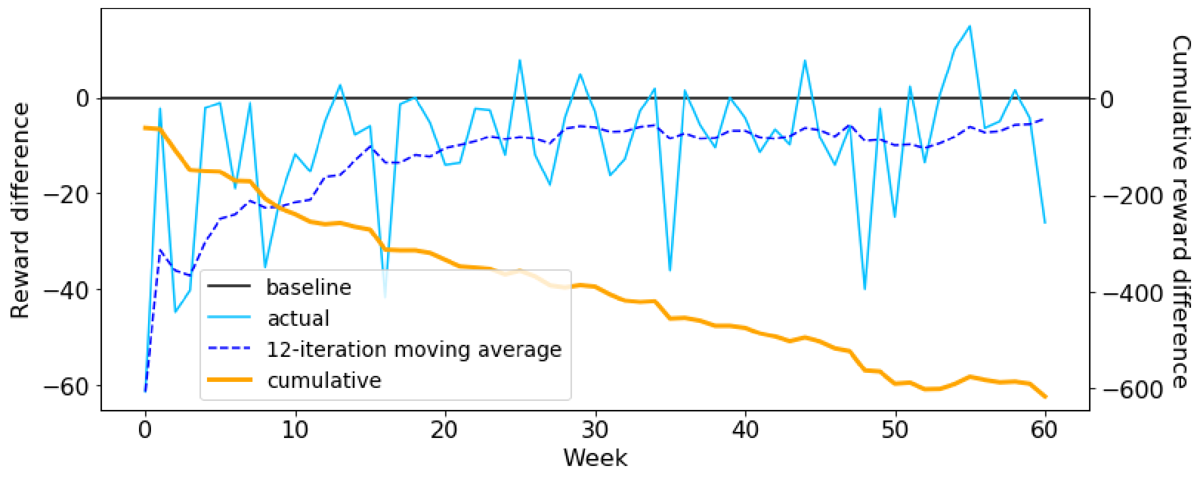

Figure 12. The dynamics of the (cumulative) reward difference between the actual reward received and the baseline reward during the first training episode

Note: the figure shows the difference between reward variables and the baseline reward during interaction phase. Baseline denotes the profits received with the acceptance threshold optimized using the traditional approach, actual and cumulative denote profits received by the RL agent. For easier perception rewards are calculated for the week the corresponding loans were issued in (for instance, the reward for week 0 shows profits generated by loan applications issued in week 0). Profit is measured in thousands of euros.

While most learning happens during the first couple of episodes, all the following episodes help the agent to make the value function model smooth and well-defined (see Figure 13). The final model shape is displayed in Figure 14. One can notice that low acceptance thresholds and high acceptance rates have the lowest values[31], meaning that accepting most of the applications is the least profitable (and in fact, very detrimental) business policy. On the other hand, high acceptance cutoffs and low acceptance rates have relatively higher but still suboptimal values, meaning that rejecting all the applications is better than accepting too many but it is a zero-profit business policy. As expected, the value function is bell-shaped and there is an optimal state-action pair in between two extremes, which is setting the acceptance threshold to 65 and operating at an acceptance rate of approximately 5%.

Comparing current results to the baseline case one can notice a lot of resemblance. The RL-optimized acceptance threshold has converged to the baseline action, while the shape of the learned value function model both in actions and states is very similar to the shape of the potential profit function in Figure 9. The only difference is the optimal acceptance rate which is 10% lower in case of the RL approach. It can be explained by the selection bias issue. Since the test dataset consists only of accepted applications (that are better than the ones that were rejected), its score distribution is shifted towards higher scores compared to the one of the whole population. This means that if we include all the rejected applications into the test dataset, the average score will decrease, thus decreasing the acceptance rate for the optimal acceptance threshold[32].

The action and reward dynamics through 100 training episodes is shown in Figures 15 and 16. In the beginning, the agent's exploration range covers the whole action set while narrowing down as the agent becomes more and more confident that the baseline action is optimal. The agent converges to the baseline action fairly quickly after the 6th episode. After the 10th episode, the average actual action rises up to 5 points over the baseline action showing that the exploration range shifts upwards. It reflects the fact that lower suboptimal acceptance thresholds are considered much less rewarding than higher suboptimal ones, that's why the latter are preferred following the Boltzmann-Q policy.

During the learning phase, the total profit per episode increases until the 6th episode when the value function model has converged to some extent and stays significantly below the baseline episode profit for the rest of the experiment. The gap can be explained by the constant exploration of suboptimal state-action pairs by the agent while in the training regime, which leads to stable suboptimal profits. However, if one looks at rewards during test episodes, the

[31] As was discussed in previous Sections, a value of a state-action pair reflects how good it is to perform a particular action in a particular state and follow an optimal policy afterwards. Since the optimal policy leads to positive reward, all the values are positive even though some state-action pairs actually lead to negative rewards.

[32] For instance, for the current test dataset an acceptance rate of 15% for previously accepted applications corresponds to the acceptance rate of around 7% for all the previously received applications.

profit difference gets close to zero already after the 5th episode and oscillates around it for the rest of the experiment.

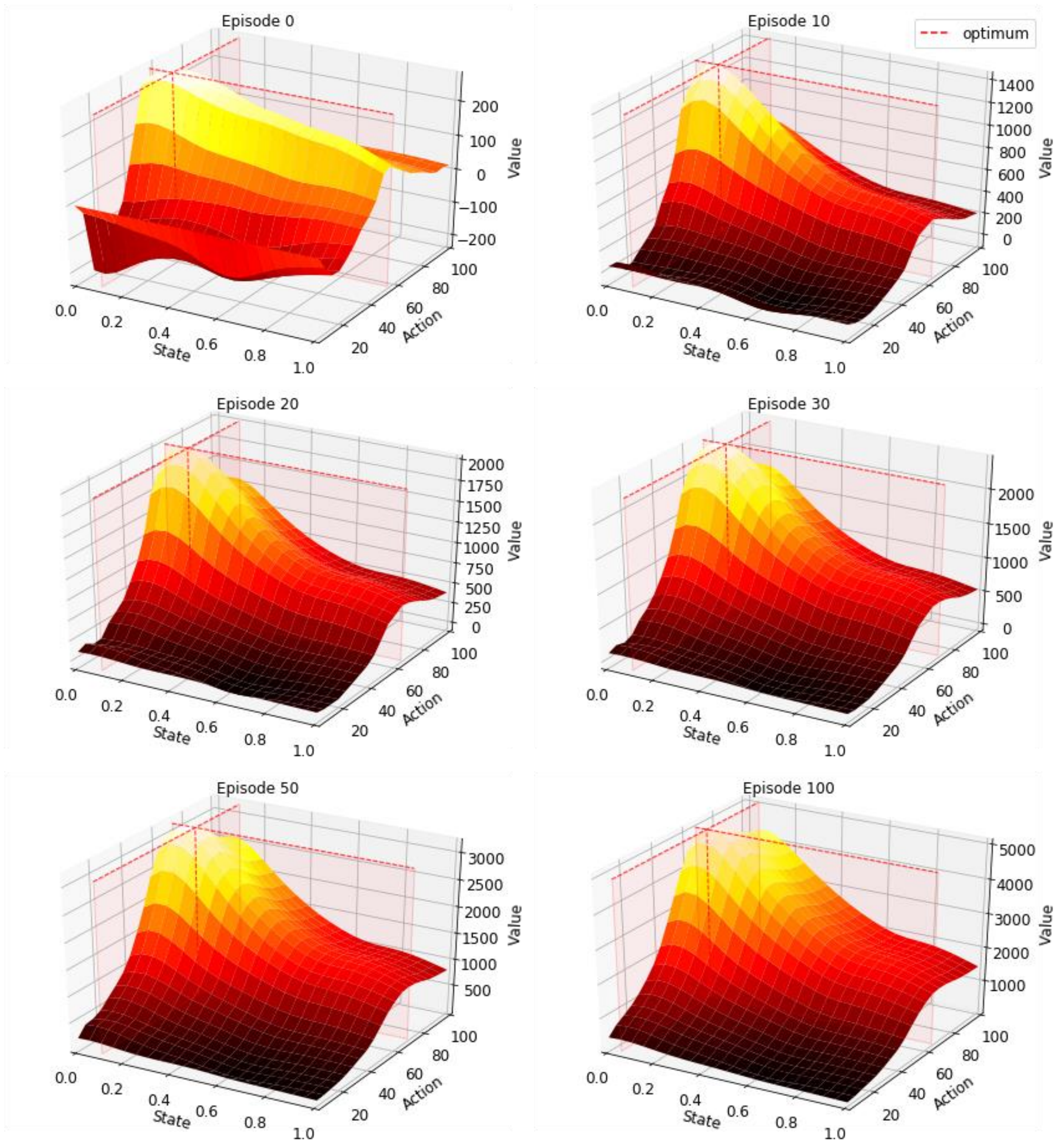

Figure 13. Value function model convergence during the training experiment

Note: State denotes the application acceptance rate during the previous week, action denotes the acceptance threshold for the following week, value is the prediction of the value function model for a particular state-action pair, optimum shows the state-action pair that corresponds to the highest value in the state-action space.

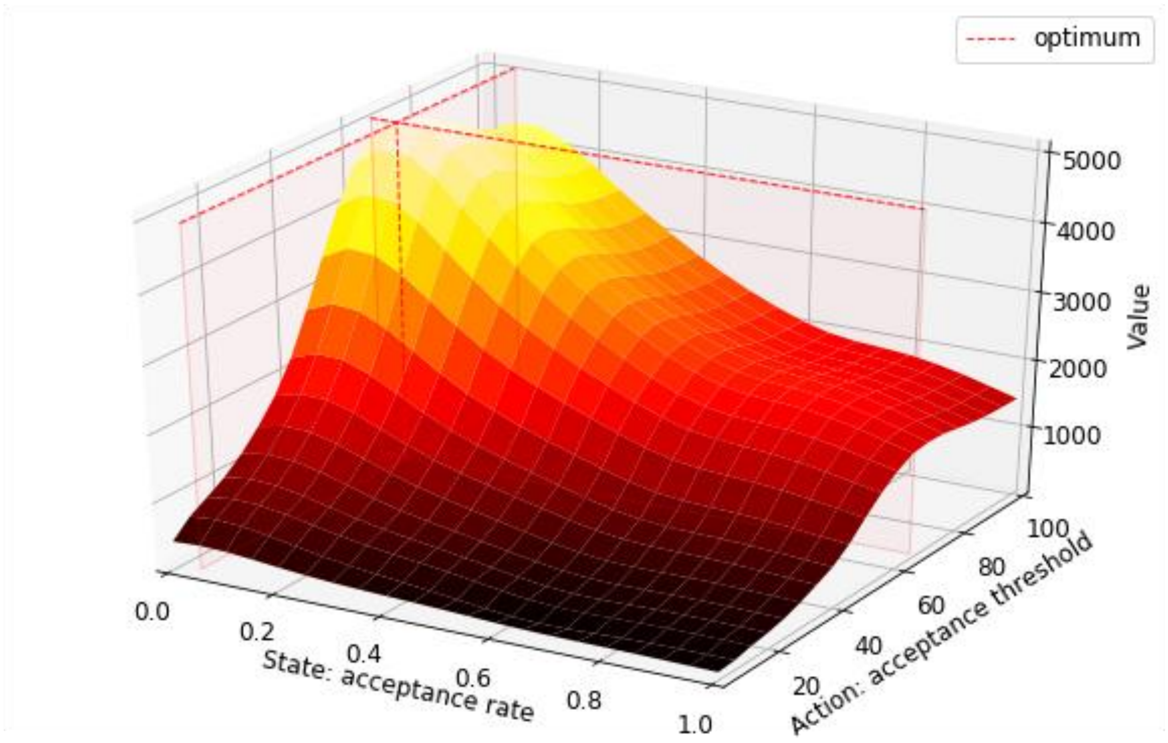

Figure 14. Final value function model shape

Note: State denotes the application acceptance rate during the previous week, action denotes the acceptance threshold for the following week, value is the prediction of the value function model for a particular state-action pair, optimum shows the state-action pair that corresponds to the highest value in the state-action space.

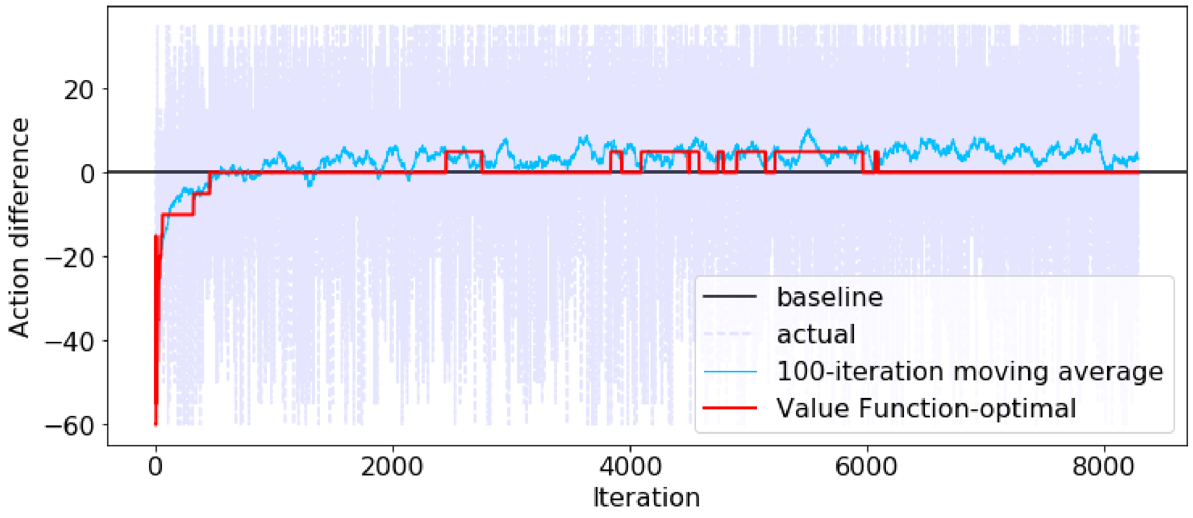

Figure 15. The dynamics of the action difference between the actual action taken and the baseline action, value function-optimized action and the baseline action during the training experiment

Note: the figure shows the difference between action variables and the baseline action during interaction and delayed learning phases. Baseline denotes the acceptance threshold optimized using the traditional approach, actual denotes the one used by the RL agent, value function-optimal denotes the one optimal according to the value function model.

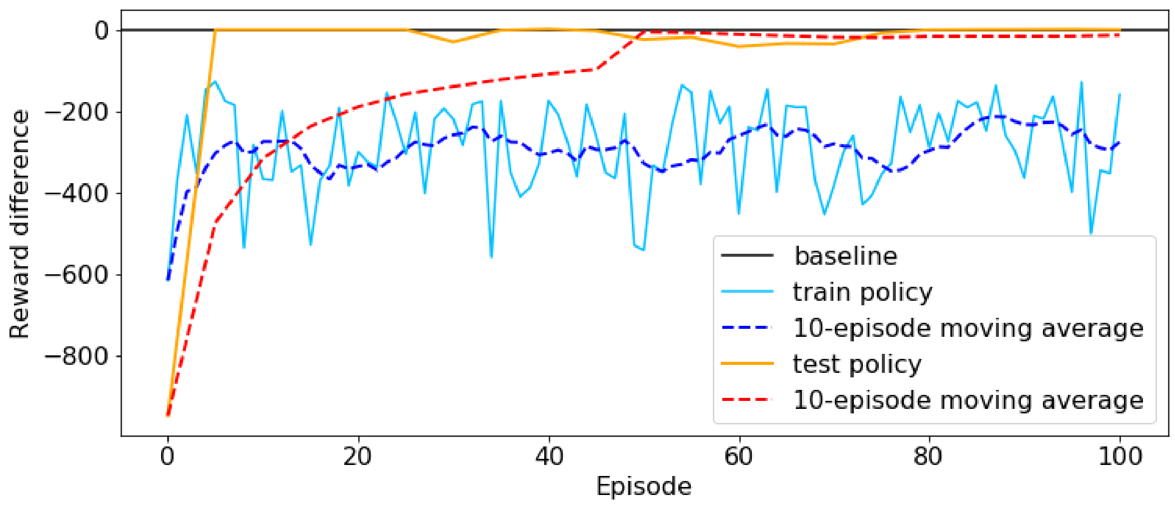

Figure 16. The dynamics of the episode reward difference between the actual episode reward received and the baseline episode reward during the training experiment for training and test episodes

Note: the figure shows the difference between reward variables and the baseline reward during interaction phase. Baseline denotes the profits received with the acceptance threshold optimized using the traditional approach, train policy and test policy denote profits received by the RL agent following training and test policy respectively. Profit is measured in thousands of euros.

To sum up, during the training experiment the RL agent finds optimal baseline acceptance threshold and is able to receive baseline profit relatively quickly, after just a couple of training episodes. Up to 50 episodes are needed to sufficiently approximate the value function model for the whole state-action space. One obvious drawback is the tendency of the value function model predictions (or Q-values) to grow indefinitely, meaning that it cannot converge without decreasing learning rate, as was discussed in the methodological section. Thus, for various sets of parameters (learning rate, discount rate, learning rate's inverse scaling parameter) the final state-action values predicted are going to be different and there is no way to determine the best parameter set. Consequently, predicted values cannot be interpreted concretely.

4.2. Test Experiment Results

In the test experiment, we let the RL agent compete with the traditional approach in distorted environments, created using the simulation with adjusted parameters. The first kind of distorted environment simulates a credit business process where the score distribution of new applications is shifted by approximately 20 points down and the score distribution of repeat applications is shifted by around 10 points down (around 15 points downward shift in the general score distribution). This kind of distortion reflects the inaccuracy in the credit scoring model assessment and/or population drift. Compared to the test dataset, the optimal acceptance threshold shifts down making traditionally optimized cutoff policy too strict for the distorted environment.

We first briefly describe the convergence of the value function model in a sample distorted episode (see Figures 17-19). The agent starts the episode with the shape of the model it has learned so far. Lower Boltzmann-Q policy temperature parameter τ lets the agent explore only most promising neighboring state-action pairs. Lower discount rate and higher learning rate let the agent adjust the value function model parameters faster than during the training experiment. Once the agent gets enough positive reinforcement about a neighboring action, the model's optimum shifts and the agent starts exploring around the new optimal action. The process continues until alternative actions stop outperforming current optimum.

The episode starts with reward difference being close to or below zero due to exploration. However, as the agent shifts to a new optimum, the average reward difference becomes positive and grows until the end of the episode allowing for rare low rewards due to exploration. The agent starts outperforming the traditional approach in terms of accumulated profit already after 20 weeks of adjustments, meaning that it responds to the first wave of matured (overdue) loans right away, and the reward difference keeps growing afterward.

The results of the test experiment are shown in Figure 20, Scenario 1. The RL agent outperforms the traditional approach in all episodes and the average episode profit received by the agent is significantly larger than the baseline. The distribution of profit differences is significantly higher than zero according to the one-tailed t-test (see Table 4). Thus, in case of the credit score distribution being significantly lower than what is estimated based on the test dataset, a dynamic RL system would successfully adapt to real environment outperforming the traditional approach.

Similar test experiments were performed for the following types of distorted environment: upwards shift in the score distribution of new and repeat applications by approximately 20 and 10 points respectively (or by around 15 points overall); an increase in default rates by 10%; a decrease in prior default rates by 10%. In case of the score distribution being significantly higher than what is estimated based on the test dataset, a dynamic RL system would successfully adapt to the distorted environment outperforming the traditional approach (see results in Figure 20, Scenario 2).

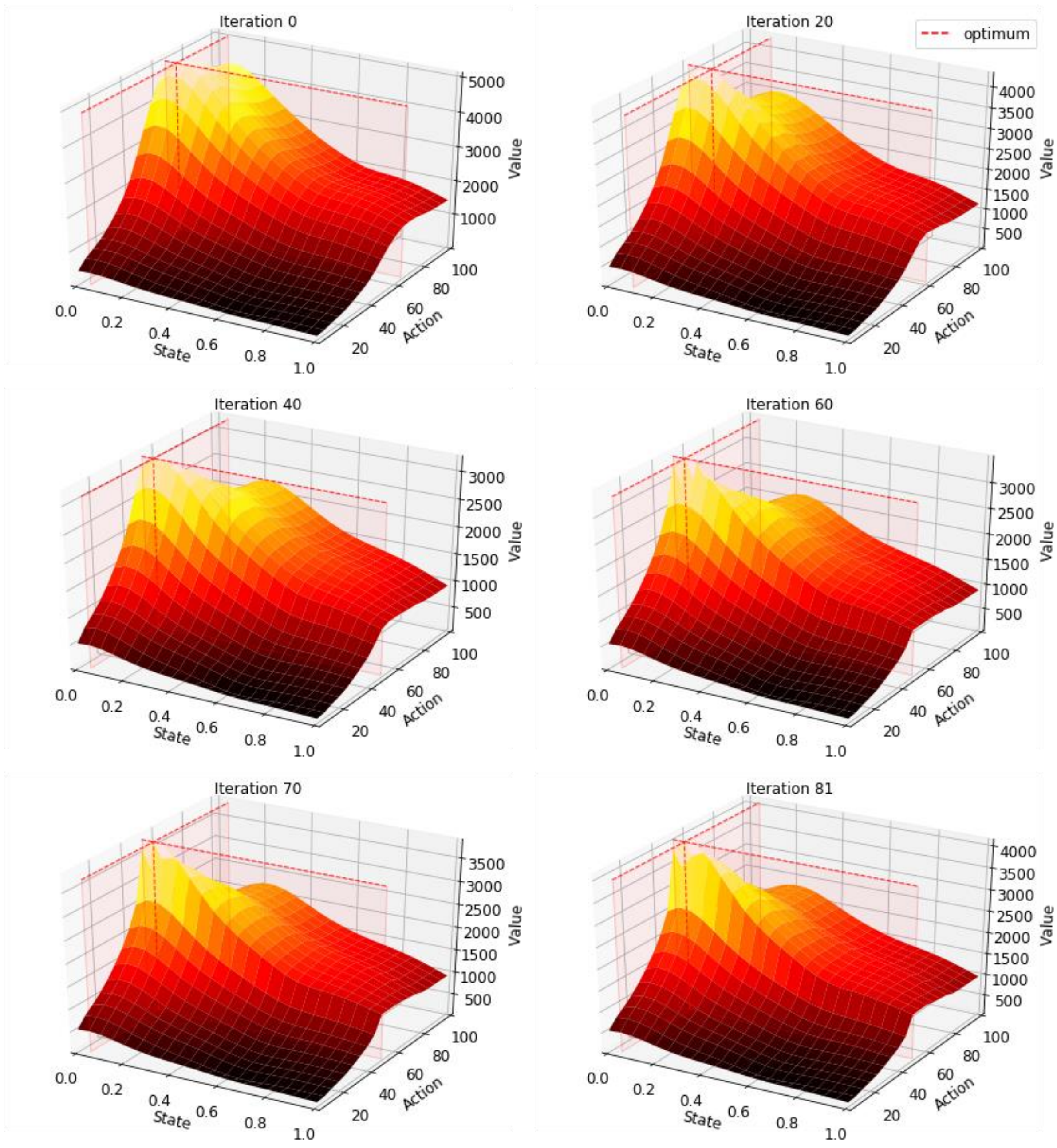

Figure 17. Value function model convergence during a sample distorted episode

Note: State denotes the application acceptance rate during the previous week, action denotes the acceptance threshold for the following week, value is the prediction of the value function model for a particular state-action pair, optimum shows the state-action pair that corresponds to the highest value in the state-action space.

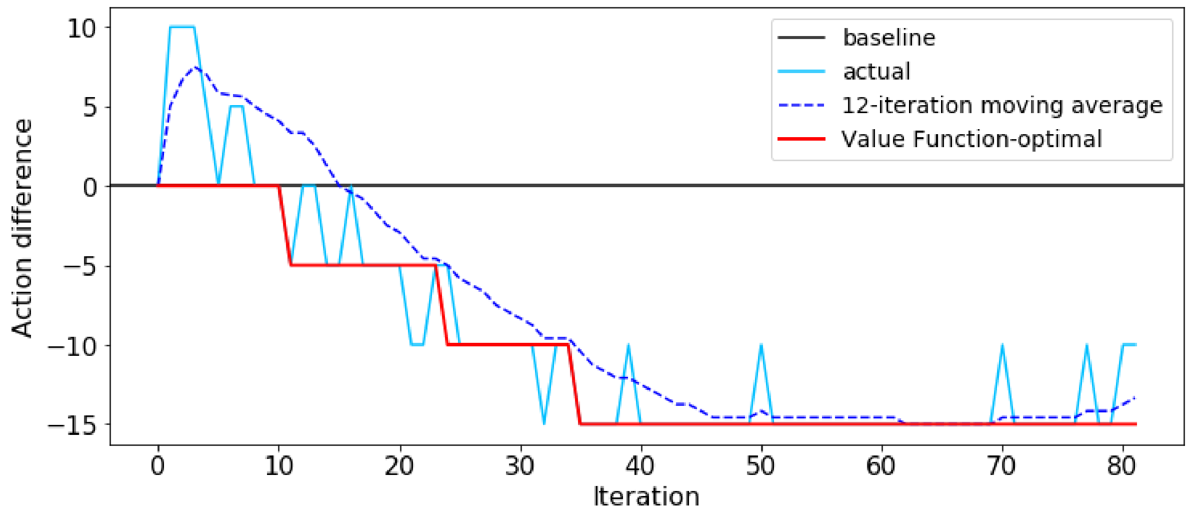

Figure 18. The dynamics of the action difference between the actual action taken and the baseline action, value function-optimized action and the baseline action during a sample distorted episode

Note: the figure shows the difference between action variables and the baseline action during interaction and delayed learning phases. Baseline denotes the acceptance threshold optimized using the traditional approach, actual denotes the one used by the RL agent, value function-optimal denotes the one optimal according to the value function model.

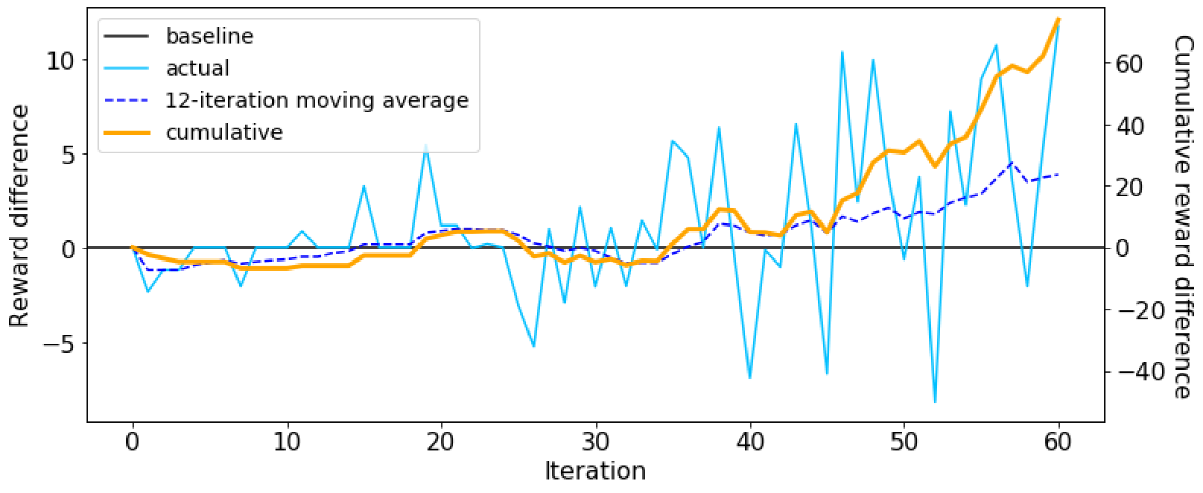

Figure 19. The dynamics of the (cumulative) reward difference between the actual reward received and the baseline reward during a sample distorted episode

Note: the figure shows the difference between reward variables and the baseline reward during interaction phase. Baseline denotes the profits received with the acceptance threshold optimized using the traditional approach, actual and cumulative denote profits received by the RL agent. For easier perception rewards are calculated for the week the corresponding loans were issued in (for instance, the reward for week 0 shows profits generated by loan applications issued in week 0). Profit is measured in thousands of euros.

In case of environments with distorted prior default rates, however, the RL agent significantly underperforms the traditional static policy when using the same parameterization as for previous experiments. The reasoning is that these environments have different reward structure from the one the value function model was trained for. Since the value function updates asymmetrically (higher actions are updated much more frequently than lower ones), learning from different reward structure might create discontinuities in the shape of the value function model. Thus, trying to adapt in some cases distorts the shape of the latter to a degree, where it diverges from the optimum.

The effectiveness of the optimization could be improved using a different set of parameters, most suitable for the problem at hand. Particularly, the agent successfully outperforms the traditional approach in a distorted environment with lower prior default rates (Figure 20, Scenario 3) if using higher learning rate and lower discount rate (being more sensitive), while in a distorted environment with higher prior default rates (Figure 20, Scenario 3) – if using lower learning rate and higher discount rate (being less sensitive). Such a contradiction makes the RL approach less able to generalize and require careful maintenance when operating in the real environment.

It is important to note that consistent exploration and adaptation of the RL agent has its cost. Consequently, the dynamic nature of the algorithm lets it outperform the traditional static approach in a highly dynamic environment, but it also might make the agent less profitable in a rather stable environment even if the agent's policy successfully converges.

Table 4. Result of the t-test for various distortion scenarios

Scenario	**t-statistic**	**p-value**
Scenario 1: downwards shift in score distribution	29.56631	1.55E-51
Scenario 2: upwards shift in score distribution	42.72066	2.45E-66
Scenario 3: downwards shift in default rates	5.172688	5.95E-07
Scenario 4: upwards shift in default rates	4.600158	6.20E-06

Note: the t-test null hypothesis is that the mean difference between the episode reward received by the RL agent and the episode reward received using the traditional approach throughout 100 episodes is equal to or lower than zero.

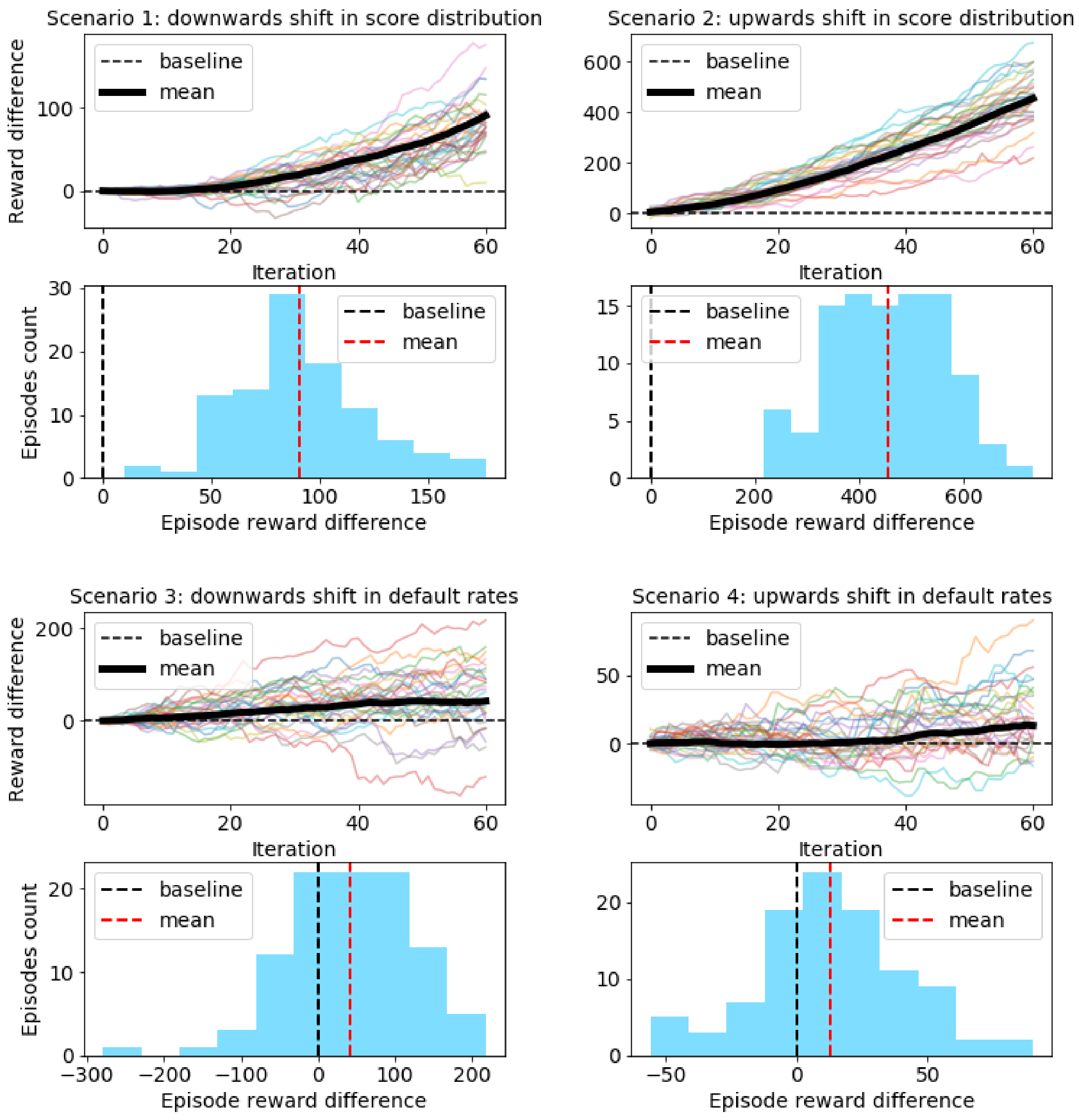

Figure 20. Test experiment results

Note: the figure shows the difference between reward variables and the baseline reward during interaction phase. Baseline denotes the profits received with the acceptance threshold optimized using the traditional approach. The upper figures for each scenario show cumulative profits for each out of 100 simulated episodes and their mean. For easier perception rewards are calculated for the week the corresponding loans were issued in (for instance, the reward for week 0 shows profits generated by loan applications issued in week 0). The lower figures show the distribution of episode profits for each out of 100 simulated episodes and their mean. Profit is measured in thousands of euros.

4.3. Results on the Real Data

Finally, we present the results of the RL agent operating on the real data. The value function model convergence, action and reward dynamics during the real episode are shown in Figures 21-23. Like in a distorted episode, the agent adjusts the value function model relatively fast. After 12 weeks of exploration, it decides that it is more rewarding to be stricter in the current environment ending up with optimal acceptance threshold of 75 and optimal acceptance rate of 10%. The average action-state value increases considerably, adapting to the reward structure of the environment.

Like in a distorted episode, weekly profit received by the agent happens to be below the baseline one during the first weeks due to the exploration. After a couple of weeks, however, the agent learns that acceptance thresholds below 65 are much less rewarding than the higher ones and stops exploring those. From then on, weekly profit difference consistently stays above or equal to zero leading to a significantly higher cumulative profit at the end of the real episode. The optimal acceptance rate starts at 5% going up to 10% during the adaptation phase and stays at that level until the end of the episode. This behavior clearly shows that the real environment score distribution is shifted upwards and using the baseline acceptance threshold leads to a higher acceptance rate than the one estimated on the historical data. Thus, an upward shift in the acceptance threshold is necessary to adjust for this bias. Positive change in the optimal acceptance rate value shows that real default rates are lower than those estimated on the historical data reflecting a slight selection bias of the dataset. Nevertheless, the verification period is still too small to draw final conclusions.

To sum up, the results show that the RL agent learns optimal policy relatively fast: the value function model correctly predicts the optimal state-action pair after 6 training episodes. The agent successfully adapts its policy in environments with distorted score distributions and prior default rates. The main drawback of the current system is that the value function model is updated asymmetrically due to the application selection issue, which might lead to the divergence of the model in an alien environment. The secondary drawback is the constant exploration cost, which makes dynamic RL system perform worse than the static traditional approach in a stable environment. Finally, the agent successfully identifies an upward score distribution shift in the real environment and significantly outperforms the traditional approach by following a stricter policy.

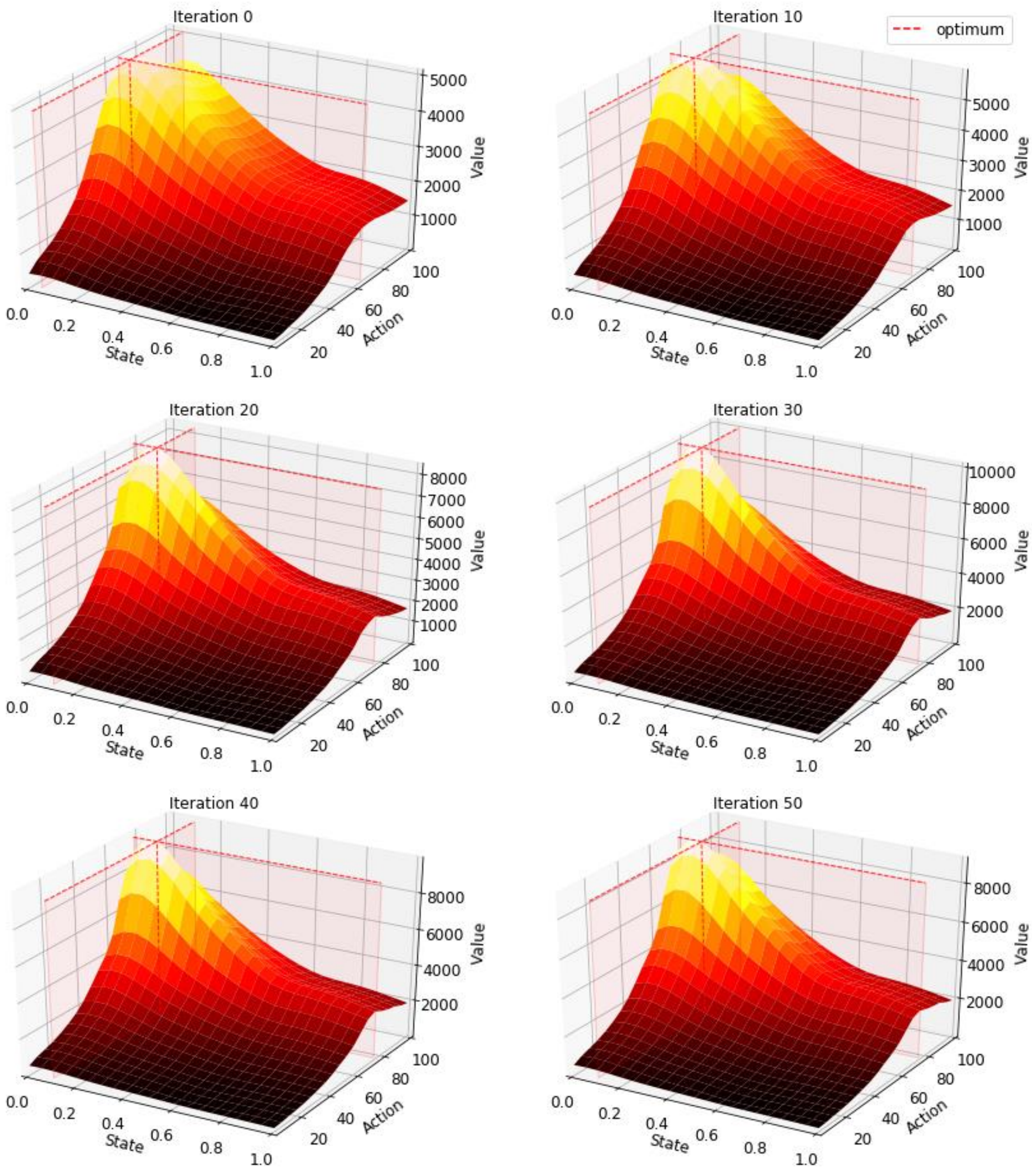

Figure 21. Value function model convergence during the real episode

Note: State denotes the application acceptance rate during the previous week, action denotes the acceptance threshold for the following week, value is the prediction of the value function model for a particular state-action pair, optimum shows the state-action pair that corresponds to the highest value in the state-action space.

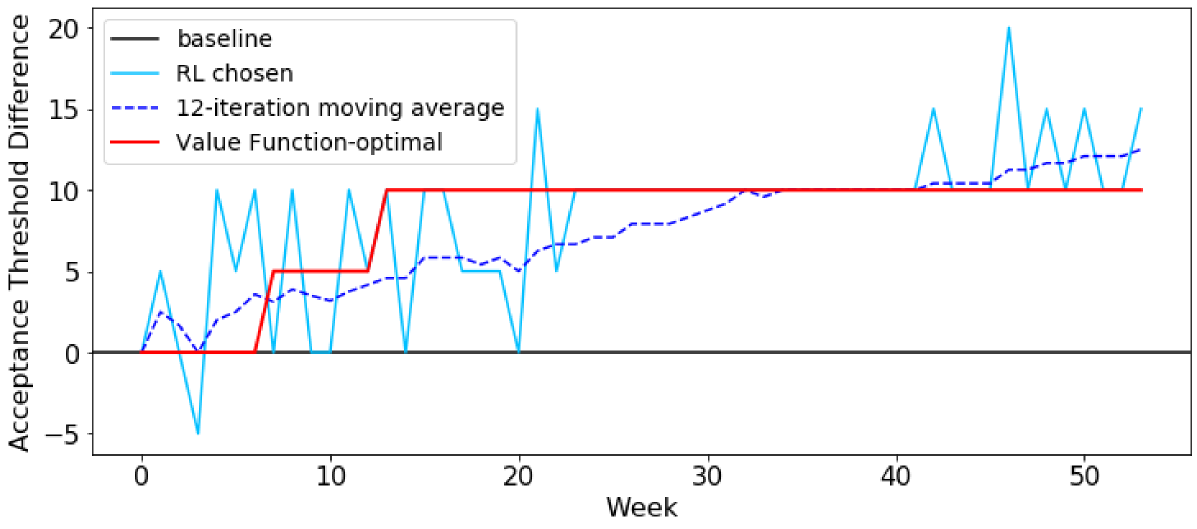

Figure 22. The dynamics of the action difference between the actual action taken and the baseline action, value function-optimized action and the baseline action during the real episode

Note: the figure shows the difference between action variables and the baseline action during interaction and delayed learning phases. Baseline denotes the acceptance threshold optimized using the traditional approach, actual denotes the one used by the RL agent, value function-optimal denotes the one optimal according to the value function model.

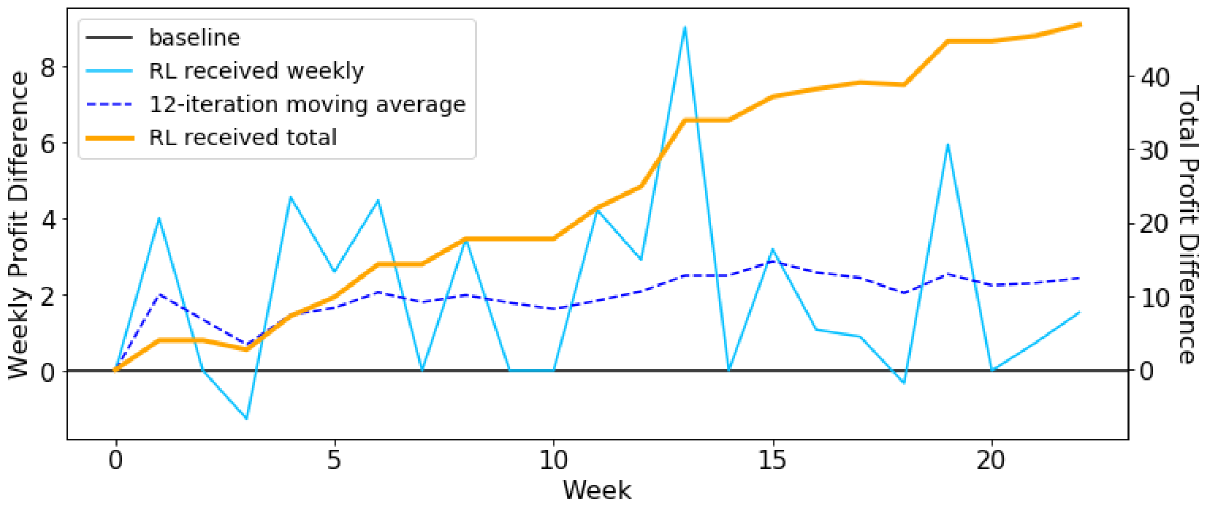

Figure 23. The dynamics of the (cumulative) reward difference between the actual reward received and the baseline reward during the real episode

Note: the figure shows the difference between reward variables and the baseline reward during interaction phase. Baseline denotes the profits received with the acceptance threshold optimized using the traditional approach, actual and cumulative denote profits received by the RL agent. For easier perception rewards are calculated for the week the corresponding loans were issued in (for instance, the reward for week 0 shows profits generated by loan applications issued in week 0). Profit is measured in thousands of euros.

5. Discussion of Results

The following section performs the analysis of the experiment results pointing out benefits and limitations of the proposed approach in comparison to the previous literature. It further summarizes the scientific and practical implications of the conducted work and draws directions for further improvements and research.

The results above show that an adaptive reinforcement learning system can be effectively used to optimize credit scoring acceptance threshold in a dynamic consumer credit environment, outperforming traditional static approaches, such as cost-sensitive optimization used in Verbraken et al. (2014) and Skarestad (2017). The RL algorithm presented in this work lets decision-maker adapt their credit scoring model to the real environment solving problems of the uncertainty of the credit scoring model's performance on the live data (discussed in Thomas et al., 2017), selection bias (investigated in Banasik et al., 2003; Wu and Hand, 2007) and population drift (studied in Thomas, 2010; Nikolaidis, 2017). Since the RL agent adapts its actions in response to actual data, the optimized policy accounts for the real misclassification costs and profit distributions rather than approximated ones as in the traditional method, solving the problem outlined in Thomas et al. (2005), Hand (2006) and Hernandez-Orallo et al. (2011).

The experiment provides more evidence of the ability of RL algorithms to outperform traditional approaches in portfolio optimization problems, as in Du et al. (2016) and Strydom (2017). It further finds the approach to perform efficient optimization given limited prior information, in line with Tesauro et al. (2006) and Kim et al. (2016), and contradicting objectives, confirming results in Huang et al. (2010) and Varela et al. (2016). Finally, the experiment shows that the RL method successfully uses simulation training to avoid initial costly poor performance in real environment, similar to findings in Tesauro et al. (2006) and Aihe and Gonzalez (2015).

On the downside, since the project is rather a proof of concept, it still has some flaws. First, since the RL agent is trained and verified on a simulation, its performance in the real environment directly depends on the goodness of fit of the simulation. In case the latter is oversimplified or biased, the deployment of the RL system might bring additional costs until it adapts to the real environment and corrects for the flawed knowledge (Aihe and Gonzalez, 2015). Thus, it is essential to develop a well-approximated simulation to make the transition of the RL agent between the training and real environments as smooth as possible.

Another drawback concerns the optimization algorithm used. Q-learning is relatively simple to understand, implement and provide the intuition on, but it comes at its cost. Since Q-values approximated by the value function model converge only in case of decreasing learning rate, eventually the model either diverges with a fixed learning rate, gradually loses its adaptive ability with a decreasing learning rate or needs constant maintenance of a human expert to control the learning rate and convergence dynamics (confirming observations made by Sutton and Barto, 2017; Baird, 1995). This disadvantage could be cured by using alternative classes of reinforcement learning algorithms, such as temporal difference (Sutton and Barto, 2017), policy gradient (Williams, 1992) and residual algorithms (Baird, 1995).

Furthermore, Q-learning was initially designed for discrete action spaces, that's why the developed RL algorithm had to use separate value function models for various actions. The disadvantage of this approach is that the system ignores the relation between similar actions and does not generalize throughout the action space (as was previously noted by Kaelbling et al., 1996). Ideally, one would use a continuous action space instead together with action-wise generalization, which could be achieved by using more advanced RL algorithms such as Asynchronous Advantage Actor-Critic (Mnih et al., 2016).

One of the drawbacks inherent to most machine learning algorithms is the parameterization problem (outlined in Farahmand and Szepesvári, 2011; Sutton and Barto, 2017). The current relatively simple system already needed a lot of experimentation to choose optimal value function model, discount rate, learning rate, exploration policy etc. Each combination of parameters provides different Q-values and convergence properties and there is no fixed rule on which combination is the best. It leads to the lack of generality and interpretability of the approach compared to the traditional methodology. The flaw can be solved with more complex RL model selection techniques (Farahmand and Szepesvári, 2011).

One of the significant flaws of the current system is the asymmetric update of the value function model caused by the application selection bias, which is new for the RL literature to the best of our knowledge. As was shown in the test experiment, although it lets the agent learn much quicker, it might sometimes cause the model to diverge in an alien environment leading to significant losses. The problem can be overcome by randomly accepting a small percentage of low-scored applications, estimating the default rates of corresponding score bins and extrapolating the outcomes for the rejected applications. This way the agent symmetrically updates the model using extrapolated outcomes covering the whole action space. Thus, the adaptation to a new reward structure will not create discontinuities in the value function model shape.

Finally, since the RL algorithm uses exploration to adapt to an environment, it has constant costs for taking suboptimal actions. It makes it underperform the static traditional approach in relatively stable conditions (the curse of online learning as discussed in Sutton and Barto, 2017). This weakness can be solved with the random low-scored application sampling. As was explained before, the latter lets the agent get an estimate of default rates in high-risk application bins, extrapolate the outcomes for rejected applications and learn from resulting estimated rewards. Thus, this approach eliminates the necessity of action exploration in the sense that it was defined before and offers much lower exploration costs and greater benefits in the form of high-risk groups default rate estimates. The low-scored application sampling is also important to make the scoring model less vulnerable to the selection bias after retraining it on the new data. That is why the related costs could be thought of as indispensable for credit scoring model building itself.

Consequently, further steps in the RL algorithm development, first of all, include the introduction of the random low-scored application sampling and experimentation with alternative RL approaches to fix the existing flaws described above. Next, the extension of the current setup is planned. The action set of the RL agent is planned to be made more detailed by differentiating between new and repeat client application acceptance thresholds, as these groups are known to have different repayment behaviors. The state space of the environment is planned to be extended with new-to-repeat loan application ratios. The next step in this direction would be a similar extension in terms of short and long-term loan applications. Finally, the reward function is planned to be extended to account for policy rules for credit volumes and new-to-repeat loans ratio.

Although the current setup of the developed RL agent is still too simple to perform the decision making independently, it is ready to be used as a decision support system by human experts. As an example, it could be utilized to optimize the acceptance threshold level based on the real environment feedback after a credit scoring model is deployed on the new market or after significant changes to the existing credit scoring model are made. As was shown in the previous section, using the agent right after the deployment of the current credit scoring model would lead to a significant increase in profit.

Additionally, the algorithm can be used together with the Monte Carlo simulation to test various what-if scenarios and perform the stress-testing as was demonstrated in the test experiment in the previous section. As international financial regulation becomes stricter demanding credit businesses to plan ahead and be ready to potential economic disturbances (Thomas, 2010), a framework developed in the thesis would help the company to correspond to regulators' requirements.

From the scientific point of view, the work offers a real example of problems when implementing credit scoring modeling in practice that are usually omitted in the literature on the topic. In particular, the acceptance threshold optimized on the test dataset following the traditional cost-sensitive optimization approach is shown to suffer from a significant credit score distribution bias most likely related to the uncertainty of the model's performance on the live data and selection bias. The developed solution to the problem contributes to existing literature on practical issues in credit scoring listed in Table A-1 in Appendix A.

As a contribution to the reinforcement learning literature, current work shows a successful proof of concept of using reinforcement learning algorithms in credit scoring for consumer credit business complementing the list of novel RL application papers in Table A-3 in Appendix A. As the problem setup considerably differs from more usual practical RL applications, a set of useful discoveries was made in terms of environment, RL algorithm and reward specification, such as using asymmetric reward learning to boost the value function model approximation process and using random low-scored application sampling as an alternative exploration policy.

6. Conclusions

To sum up, the thesis addresses some common issues of the practical implementation of credit scoring, such as scoring model's performance uncertainty, selection bias, population drift and the specification of business objectives. As opposed to numerous papers on the topic, we focus on credit score acceptance threshold optimization rather than the scoring model itself. We discuss and empirically show significant drawbacks of the traditional approach to cost-sensitive cutoff point optimization, which are mainly caused by its static nature. Instead, we propose a dynamic reinforcement learning framework that can correct for the flawed prior knowledge and adapt to the real environment based on the live feedback rather than oversimplifying theoretical assumptions. To the best of our knowledge, it is the first attempt of applying reinforcement learning to the problem of credit scoring in consumer credit business. Additionally, we conduct the research using real data of an international consumer loans provider, which adds practical sense to the work.

The results show that the traditional cutoff optimization approach does not ensure the optimality of the acceptance threshold, which might lead to biased conclusions and significant losses. The proposed dynamic reinforcement learning system manages to outperform the traditional method both in a simulated and real credit business environment leading to significantly higher total profits of the credit company. Consequently, the main scientific contribution of the current work is the proof of concept of applying a reinforcement learning algorithm in credit scoring for consumer credit business. Additionally, a set of useful discoveries was made in terms of the environment, RL algorithm and reward specification in the credit scoring context, such as using asymmetric reward learning to boost the value function model approximation process and using random low-scored application sampling as an alternative exploration policy. Finally, a complete programming package was developed to implement the designed RL system[33].

The current system could be improved by experimenting with the random low-scored application sampling and alternative RL algorithms such as temporal difference, residual algorithms and policy gradient (such as Asynchronous Advantage Actor-Critic). Next, the extension of the current setup is planned. The state-action space of the RL problem can be extended to account for the applications heterogeneity. For instance, due to the difference between new and repeat client loan application groups the optimization of distinct acceptance thresholds for those could certainly improve the agent's performance. The corresponding extension of the state space of the environment would be new-to-repeat application and loan ratios variables. Another differentiation could be done by the duration of the loan applied for, i.e. into short-term and long-term loan applications. Finally, the reward function is planned to be extended to account for policy rules for credit volumes and new-to-repeat loans ratio.

Overall, due to the novelty of the approach application and intensive research in the field, there is a lot of room for experimentation and improvement. However, it is important to remember that with massive machine learning systems like reinforcement learning agents a 'step-by-step' rather than 'all-at-once' improvement strategy should be preferred in order to go through the experimentation process as smoothly and quickly as possible.

[33] The project repository can be found by following the link: https://github.com/MykolaGerasymovych/Optimizing-Acceptance-Threshold-in-Credit-Scoring-using-Reinforcement-Learning.git

References

1. Abdou, H.A. and Pointon, J., 2011. Credit scoring, statistical techniques and evaluation criteria: A review of the literature. *Intelligent Systems in Accounting, Finance and Management*, *18*(2-3), pp.59-88.
2. Abe, N., Melville, P., Pendus, C., Reddy, C.K., Jensen, D.L., Thomas, V.P., Bennett, J.J., Anderson, G.F., Cooley, B.R., Kowalczyk, M. and Domick, M., 2010, July. Optimizing debt collections using constrained reinforcement learning. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 75-84). ACM.
3. Aihe, D.O. and Gonzalez, A.J., 2015. Correcting flawed expert knowledge through reinforcement learning. *Expert Systems with Applications*, *42*(17-18), pp.6457-6471.
4. Azar, M.G., Munos, R., Ghavamzadeh, M. and Kappen, H., 2011. Reinforcement learning with a near optimal rate of convergence.
5. Baird, L.C., 1999. Reinforcement learning through gradient descent. *Robotics Institute*, p.227.
6. Banasik, J., Crook, J. and Thomas, L., 2003. Sample selection bias in credit scoring models. *Journal of the Operational Research Society*, *54*(8), pp.822-832.
7. Barisitz, S., 2013. Nonperforming Loans in CESEE–An Even Deeper Definitional Comparison. *Focus on European Economic Integration Q*, *3*, pp.64-81.
8. Beling, P., Covaliu, Z. and Oliver, R.M., 2005. Optimal scoring cutoff policies and efficient frontiers. *Journal of the Operational Research Society*, *56*(9), pp.1016-1029.
9. Bellman, R., 2013. *Dynamic programming*. Courier Corporation.
10. Bellotti, T. and Crook, J., 2013. Forecasting and stress testing credit card default using dynamic models. *International Journal of Forecasting*, *29*(4), pp.563-574.
11. Blöchlinger, A. and Leippold, M., 2006. Economic benefit of powerful credit scoring. *Journal of Banking & Finance*, *30*(3), pp.851-873.
12. BRIAT, P., 2006. *A Markovian approach to the analysis and optimization of a portfolio of credit card accounts* (Doctoral dissertation).
13. Buhmann, M.D., 2003. *Radial basis functions: theory and implementations* (Vol. 12). Cambridge university press.

14. Crook, J. and Banasik, J., 2004. Does reject inference really improve the performance of application scoring models?. *Journal of Banking & Finance*, *28*(4), pp.857-874.

15. Crook, J.N., Edelman, D.B. and Thomas, L.C., 2007. Recent developments in consumer credit risk assessment. *European Journal of Operational Research*, *183*(3), pp.1447-1465.

16. Crook, J.N., Thomas, L.C. and Hamilton, R., 1992. The degradation of the scorecard over the business cycle. *IMA Journal of Management Mathematics*, *4*(1), pp.111-123.

17. Dariane, A.B. and Moradi, A.M., 2016. Comparative analysis of evolving artificial neural network and reinforcement learning in stochastic optimization of multireservoir systems. *Hydrological Sciences Journal*, *61*(6), pp.1141-1156.

18. Deep Think Team, "Report for the PAKDD 2009 Data Mining Competition," http://sede.neurotech.com.br/PAKDD2009/files/47.PDF, 2009

19. Devraj, A.M. and Meyn, S., 2017. Zap Q-Learning. In *Advances in Neural Information Processing Systems* (pp. 2232-2241).

20. Dey, S., 2010, June. Credit limit management using action-effect models. In *Financial Theory and Engineering (ICFTE), 2010 International Conference on* (pp. 112-115). IEEE.

21. Dey, S., 2010. Modeling the Combined Effects of Credit Limit Management and Pricing Actions on Profitability of Credit Card Operations. *International Journal of Business and Management*, *5*(4), p.168.

22. Du, X., Zhai, J. and Lv, K., 2016. Algorithm Trading using Q-Learning and Recurrent Reinforcement Learning. *positions*, *1*, p.1.

23. Eisenbeis, R.A., 1978. Problems in applying discriminant analysis in credit scoring models. *Journal of Banking & Finance*, *2*(3), pp.205-219.

24. Farahmand, Amir-massoud, and Csaba Szepesvári. "Model selection in reinforcement learning." *Machine learning* 85.3 (2011): 299-332.

25. Finlay, S., 2010. Credit scoring for profitability objectives. *European Journal of Operational Research*, *202*(2), pp.528-537.

26. Fluss, R., Faraggi, D. and Reiser, B., 2005. Estimation of the Youden Index and its associated cutoff point. *Biometrical journal*, *47*(4), pp.458-472.

27. Gao, X. and Chan, L., 2000. An algorithm for trading and portfolio management using Q-learning and sharpe ratio maximization. In *Proceedings of the international conference on neural information processing* (pp. 832-837).

28. Hand, D.J., 2006. Classifier technology and the illusion of progress. *Statistical science*, pp.1-14.

29. Hand, D.J., 2009. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine learning*, *77*(1), pp.103-123.

30. Hernández-Orallo, J., Flach, P.A. and Ramirez, C.F., 2011, June. Brier Curves: a New Cost-Based Visualisation of Classifier Performance. In *ICML* (pp. 585-592).

31. Huang, Z., van der Aalst, W.M., Lu, X. and Duan, H., 2010. An adaptive work distribution mechanism based on reinforcement learning. *Expert Systems with Applications*, *37*(12), pp.7533-7541.

32. Kaelbling, L.P., Littman, M.L. and Moore, A.W., 1996. Reinforcement learning: A survey. *Journal of artificial intelligence research*, *4*, pp.237-285.

33. Kim, B.G., Zhang, Y., van der Schaar, M. and Lee, J.W., 2016. Dynamic pricing and energy consumption scheduling with reinforcement learning. *IEEE Transactions on Smart Grid*, *7*(5), pp.2187-2198.

34. Kober, J., Bagnell, J.A. and Peters, J., 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, *32*(11), pp.1238-1274.

35. Lane, S.H., Handelman, D.A. and Gelfand, J.J., 1992. Theory and development of higher-order CMAC neural networks. *IEEE Control Systems*, *12*(2), pp.23-30.

36. Lee, J.W., 2001. Stock price prediction using reinforcement learning. In *Industrial Electronics, 2001. Proceedings. ISIE 2001. IEEE International Symposium on* (Vol. 1, pp. 690-695). IEEE.

37. Lessmann, S., Baesens, B., Seow, H.V. and Thomas, L.C., 2015. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, *247*(1), pp.124-136.

38. Louzada, F., Ara, A. and Fernandes, G.B., 2016. Classification methods applied to credit scoring: Systematic review and overall comparison. *Surveys in Operations Research and Management Science*, *21*(2), pp.117-134.

39. Malik, M. and Thomas, L.C., 2012. Transition matrix models of consumer credit ratings. *International Journal of Forecasting*, *28*(1), pp.261-272.

40. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D. and Kavukcuoglu, K., 2016, June. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning* (pp. 1928-1937).

41. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *Nature*, *518*(7540), p.529.

42. Moody, J. and Saffell, M., 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, *12*(4), pp.875-889.

43. Moody, J. and Saffell, M., 2001. Learning to trade via direct reinforcement. *IEEE transactions on neural Networks*, *12*(4), pp.875-889.

44. Neuneier, R., 1996. Optimal asset allocation using adaptive dynamic programming. In *Advances in Neural Information Processing Systems* (pp. 952-958).

45. Nikolaidis, D., Doumpos, M. and Zopounidis, C., 2017. Exploring Population Drift on Consumer Credit Behavioral Scoring. In *Operational Research in Business and Economics*(pp. 145-165). Springer, Cham.

46. Oliver, R.M. and Thomas, L.C., 2009. Optimal score cutoffs and pricing in regulatory capital in retail credit portfolios.

47. Oliver, R.M. and Wells, E., 2001. Efficient frontier cutoff policies in credit portfolios. *Journal of the Operational Research Society*, *52*(9), pp.1025-1033.

48. OpenAI. OpenAI Dota 2 1v1 bot, 2017. URL https://openai.com/the-international/

49. Oreski, S., Oreski, D. and Oreski, G., 2012. Hybrid system with genetic algorithm and artificial neural networks and its application to retail credit risk assessment. *Expert systems with applications*, *39*(16), pp.12605-12617.

50. Rahimi, A. and Recht, B., 2008. Random features for large-scale kernel machines. In *Advances in neural information processing systems* (pp. 1177-1184).

51. Rana, R. and Oliveira, F.S., 2015. Dynamic pricing policies for interdependent perishable products or services using reinforcement learning. *Expert Systems with Applications*, *42*(1), pp.426-436.

52. Régis, D.E. and Artes, R., 2016. Using multi-state markov models to identify credit card risk. *Production*, *26*(2), pp.330-344.

53. Sato, M., 2016. Quantitative Realization of Behavioral Economic Heuristics by Cognitive Category: Consumer Behavior Marketing with Reinforcement Learning.

54. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M. and Dieleman, S., 2016. Mastering the game of Go with deep neural networks and tree search. *nature*, *529*(7587), pp.484-489.

55. Skarestad, H., 2017. *A Novel Profit Scoring Method for Classifying Credit Card Applications* (Master's thesis, NTNU).

56. So, M.C., 2009. *Optimizing credit limit policy by Markov Decision Process Models* (Doctoral dissertation, University of Southampton).

57. So, M.M. and Thomas, L.C., 2011. Modelling the profitability of credit cards by Markov decision processes. *European Journal of Operational Research*, *212*(1), pp.123-130.

58. Sousa, M.R., Gama, J. and Brandão, E., 2013. Introducing time-changing economics into credit scoring. *Universidade do Porto, Faculdade de Economia do Porto*.

59. Strydom, P., 2017. Funding optimization for a bank integrating credit and liquidity risk. *Journal of Applied Finance and Banking*, *7*(2), p.1.

60. Sutton, R.S. and Barto, A.G., 2017. *Reinforcement learning: An introduction*. Cambridge: MIT press.

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

62. Tesauro, G., Gondek, D.C., Lenchner, J., Fan, J. and Prager, J.M., 2013. Analysis of watson's strategies for playing Jeopardy!. *Journal of Artificial Intelligence Research*.

63. Tesauro, G., Jong, N.K., Das, R. and Bennani, M.N., 2006, June. A hybrid reinforcement learning approach to autonomic resource allocation. In *Autonomic Computing, 2006. ICAC'06. IEEE International Conference on* (pp. 65-73). IEEE.

64. Thomas, L., Crook, J. and Edelman, D., 2017. *Credit scoring and its applications* (Vol. 2). Siam.

65. Thomas, L.C., 2010. Consumer finance: Challenges for operational research. *Journal of the Operational Research Society*, *61*(1), pp.41-52.

66. Thomas, L.C., Oliver, R.W. and Hand, D.J., 2005. A survey of the issues in consumer credit modelling research. *Journal of the Operational Research Society*, *56*(9), pp.1006-1015.

67. Trench, M.S., Pederson, S.P., Lau, E.T., Ma, L., Wang, H. and Nair, S.K., 2003. Managing credit lines and prices for bank one credit cards. *Interfaces*, *33*(5), pp.4-21.

68. Van Hasselt, H., Guez, A. and Silver, D., 2016, February. Deep Reinforcement Learning with Double Q-Learning. In *AAAI* (Vol. 16, pp. 2094-2100).

69. Vanhulsel, M., Janssens, D., Wets, G. and Vanhoof, K., 2009. Simulation of sequential data: An enhanced reinforcement learning approach. *Expert Systems with Applications*, *36*(4), pp.8032-8039.

70. Varela, M., Viera, O. and Robledo, F., 2016. A q-learning approach for investment decisions. In *Trends in Mathematical Economics* (pp. 347-368). Springer, Cham.

71. Verbraken, T., Verbeke, W. and Baesens, B., 2013. A novel profit maximizing metric for measuring classification performance of customer churn prediction models. *IEEE Transactions on Knowledge and Data Engineering*, *25*(5), pp.961-973.

72. Viaene, S. and Dedene, G., 2005. Cost-sensitive learning and decision making revisited. *European journal of operational research*, *166*(1), pp.212-220.

73. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M. and De Freitas, N., 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.

74. Watkins, C.J.C.H., 1989. *Learning from delayed rewards*(Doctoral dissertation, King's College, Cambridge).

75. Williams, R.J., 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Reinforcement Learning* (pp. 5-32). Springer, Boston, MA.

76. Wu, I.D. and Hand, D.J., 2007. Handling selection bias when choosing actions in retail credit applications. *European journal of operational research*, *183*(3), pp.1560-1568.

Appendices

Appendix A

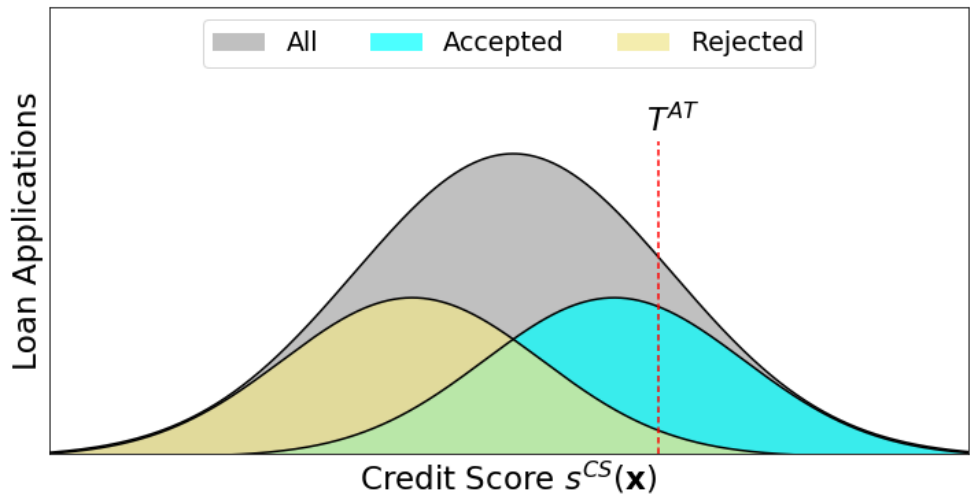

Figure A-1. Selection bias in credit scoring.

Notes: $s^{CS}(\boldsymbol{x})$ – application's credit score estimated based on the application data $\boldsymbol{x}$**;** red dotted line is the estimated optimal acceptance threshold T^{AT}. The latter is optimized for the dataset of accepted loan applications. However, they make only a part of the overall loan applications set. Another part is the rejected applications that usually have on average lower credit score than the accepted ones. As long as the credit score distributions of accepted and rejected applications are not perfectly aligned, the traditionally optimized acceptance threshold would be suboptimal for the overall loan applications dataset.

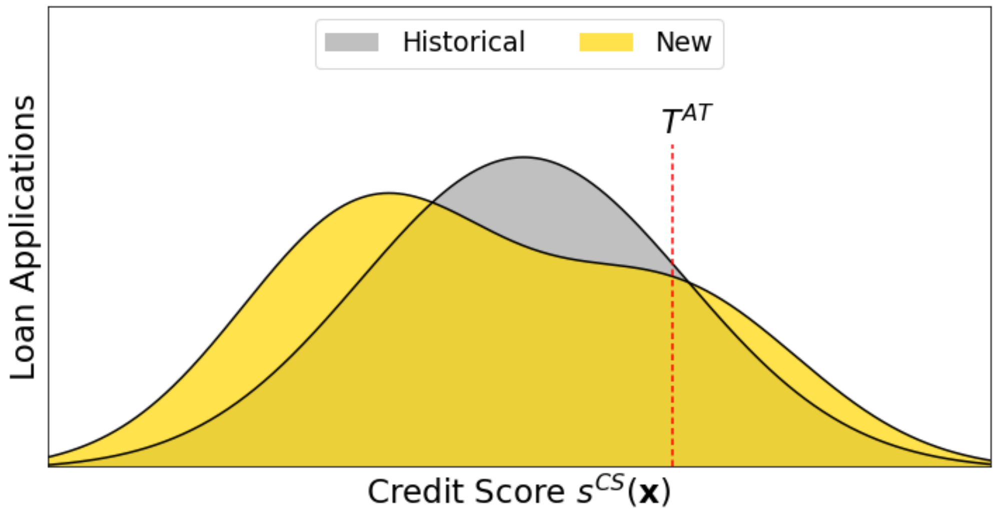

Figure A-2. Population drift in credit scoring.

Notes: $s^{CS}(\boldsymbol{x})$ – application's credit score estimated based on the application data $\boldsymbol{x}$**;** red dotted line is the estimated optimal acceptance threshold T^{AT}. The latter is optimized for the dataset of historical accepted loan applications. However, new loan applications might have a different distribution due to changes in business policy, legal rules, market conditions or macroeconomic environment. As long as the credit score distributions of historically accepted and new applications are not perfectly aligned, the traditionally optimized acceptance threshold would be suboptimal for the incoming loan applications.

Table A-1. Practical issues in credit scoring.

Paper	Issue				Main conclusions
	PS	PD	SB	BO	
Eisenbeis (1978)	X		X		Selection bias significantly affects estimates of the group means and dispersions, the true cutoff point, and true error rates. Reclassification and augmentation reject inference do not solve the issue.
Oliver and Wells (2001)	X			X	The optimal cutoff policy significantly depends on the nature of business objectives. Those should at least include risk, volume and profit tradeoffs apart from the tradeoffs between expected losses and expected profit.
Banasik et al. (2003)			X		Training credit scoring model on high-risk application dataset leads to high selection bias and model's performance uncertainty. These issues cannot be fully solved with traditional probit model with selection technique.
Crook and Banasik (2004)			X		Selection bias significantly affects loan application datasets where a large proportion of cases are rejected. Extrapolation techniques are useless and harmless for the model's performance.
Thomas et al. (2005)	X		X	X	The critical issue in credit scoring research is to develop a system that can: cope with selection bias; produce profit-based scoring; accurately reflect business objectives; relate scoring of individual applications to risk at the portfolio and macro levels.
Blochlinger and Leippold (2005)	X	X			Competitors' policies and quality of scoring models produce lead to population drift in the company's customer base affecting the performance of the firm's credit scoring model.
Viaene and Dedene (2005)	X				Cost-sensitive learning and cutoff optimization assumes that the probability classifiers' estimates are unbiased and well-calibrated. This is not true for many error-based learners like decision trees, ensemble methods or transformed scores.
Beling et al. (2005)	X			X	Credit scoring models should account for multiple business objectives including profit, market share, and loss. The traditional approach is limited to satisfying only two objectives at a time.
Oliver and Thomas (2009)	X			X	The credit scoring cutoff policies' objective should account for constraints of regulatory capital requirements, such as those of the Basel Capital Accord.
Hand (2006)	X	X	X		The traditional assumptions of credit scoring are oversimplifying. 1. Misclassification costs are dynamic and not accurately known. 2. Selectivity bias significantly affects classifier's performance on the live data. 3. Changes in economic and market environment, marketing, and advertising practices cause population drift. These false assumptions make credit scoring biased and traditionally optimized cutoff point suboptimal.
Wu and Hand (2007)			X		Selection bias severely affects credit scoring model's performance. When most previous customers received the same action, Heckman's method does not help.
PAKDD (2009)		X	X		In real credit scoring selection bias strongly affects the training dataset, while changes in the economy and the market significantly affect the performance of the scoring model through population drift.
Thomas (2010)		X			Payday and short-term loans with extremely high interest rates are strongly affected by population drift reflected in changes in economic and market environment, borrower behavior, and circumstances. The credit model needs to be able to quickly adapt to those.
Finlay (2010)	X				In consumer credit scoring accurate specification of business objectives using bad debt, revenue and profits rather than more general loss functions like sum of squared errors, improves scoring model's performance.

Note: PS - Profit Scoring, PD - Population Drift, SB - Selection Bias, BO - Business Objectives; main conclusions are summarized by the author.

Table A-1 Continued.

Paper	Issue				Main conclusions
	PS	PD	SB	BO	
Dey (2010)	X		X	X	Selection bias strongly affects credit scoring models. Additionally, the objective function of a model has to be extended with various business constraints like resource limits, growth targets, bad debt limits, and legal constraints like APR caps.
Sousa et al. (2013)		X			A dynamic credit scoring model that accounts for time-changing economic factors using the central tendency of default adjustment is found to outperform the traditional static approach.
Bellotti and Crook (2013)		X			Dynamic credit scoring models that include behavioral data and macroeconomic variables significantly outperform static models. Macroeconomic variables affect all predicted default probabilities, rather than at the individual account level.
Verbraken et al. (2014)	X				Profit-based acceptance optimization technique that accounts for misclassification costs distribution outperforms alternative cost-sensitive approaches in terms of both accuracy and monetary value.
Thomas et al. (2017)	X	X	X		There are still major problems in incorporating business objectives into credit scoring models. Those should account for economic and market conditions, individual consumer's characteristics, attrition rates. Reject inference cannot be valid if the assumptions about accepted and rejected populations are incorrect.
Nikolaidis (2017)		X			Credit scoring environment has a dynamic nature, while scoring models are usually static. Authors show that severe deterioration in the macroeconomic environment does not affect the separation ability of credit scoring models but significantly changes client portfolio and population odds.
Skarestad (2017)	X				The credit scoring model approach aimed to balance the risk and the expected profits of an applicant outperforms conventional methods in terms of profitability. The profit-based cutoff selection procedure can also outperform conventional methods in terms of profitability as it utilizes the distribution of losses and average gain.

Note: PS - Profit Scoring, PD - Population Drift, SB - Selection Bias, BO - Business Objectives; main conclusions are summarized by the author.

Table A-2. Major reinforcement learning achievements in learning playing games.

Year	Authors	Game	Achievement
1995	Tesauro et al.	Backgammon	RL agent TD-Gammon could learn from playing backgammon against itself up to a grandmaster level
2013	Tesauro et al.	“Jeopardy!”	RL agent IBM Watson won the first prize in a TV quiz show “Jeopardy!” in an exhibition match against human champions
2015	DeepMind (Mnih et al.)	Atari 2600 video games	RL agents could learn to play Atari 2600 games better than human experts
2016	DeepMind (Silver et al.)	Go	RL agent AlphaGo managed to outperform an 18-time world champion player in a board game of Go considered the most complicated to master for an artificial intelligence machine
2017	OpenAI	Dota 2 (1 vs 1)	RL agent learned the game from scratch by self-play and outperformed top Dota 2 players.

Note: summarized by the author.

Table A-3. Reinforcement learning in finance and business management.

Paper	**Problem**	**Main conclusions**
Neuneier (1996)	Forex and stock exchange trading	A neural network-based Q-learning agent is trained and tested on the artificially simulated exchange rate and then on German stock index DAX. The agent is found to outperform the baseline index and the neural network model for stock price prediction.
Gao and Chan (2000)	Forex trading.	An RL algorithm with neural network value function approximator and Boltzmann policy is applied to trade USD/DM currency. The agent is found to make significantly higher profits than traditional approaches based on forecasts and labelled data.
Lee (2001)	Korean stock market trading.	An RL agent with value function approximation by an artificial neural network is trained to trade stocks on the Korean stock market. The agent is found to make positive profits.
Moody and Saffell (2001)	Forex, stocks and securities trading	Trading algorithms based on Q-learning and recurrent reinforcement learning (RRL) are trained on a simulated stock, USD/GBP exchange rate and S&P 500/T-Bill. Algorithms are found to outperform heuristic benchmarks, and RRL is found to perform better than Q-learning.
Tesauro et al. (2006)	Resource allocation in a multi-application prototype Data Center scenario.	An RL agent is found to learn high-quality management policies with little or no built-in system specific knowledge. By training offline it is possible to avoid initial poor performance costs. RL is found to deal effectively with both transients and switching delays.
Vanhulsel et al. (2009)	Simulation of sequential data in activity-based travel demand.	RL algorithm is shown to successfully simulate sequential data. Using a regression tree function approximator produces a more optimal solution much faster than traditional Q-learning approach.
Abe et al. (2010)	Tax and debt collection optimization.	An RL agent is found to effectively optimize debt collection policies. Basing the deployed system on data modeling lets it adapt to environment changes without significant extra labor or costs.
Huang et al. (2010)	Work distribution in business process management.	An adaptive work distribution mechanism based on reinforcement learning is shown to learn and reason suitable work distribution policies within the change of process conditions. A test learning-based simulation experiment shows that the mechanism outperforms reasonable heuristic or hand-coded approaches.
Aihe and Gonzalez (2015)	Improvement in the proficiency of software agents performing a tactical task.	An RL agent is tested to perform optimization given incorrect, incomplete or outdated prior knowledge about the system. It is found to successfully improve a flawed tactical agent by revising its knowledge through practice in a simulated version of its operational environment.
Rana and Oliveira (2015)	Pricing optimization of perishable interdependent products when demand is stochastic and its functional form unknown.	An RL model is shown to learns the relationship between price and demand without any prior knowledge or assumptions through the observation of realized demand to derive an optimal dynamic pricing policy. It is shown to faster learn and better approximate the optimal policy compared to traditional dynamic pricing models.
Darian and Moradi (2016)	Optimization of multireservoir systems.	An RL algorithm based on an artificial neural net is developed and tested using the simulation method. The neural network-based agent is shown to outperform Q-learning based algorithm.

Note: main conclusions are summarized by the author.

Table A-3 Continued.

Paper	**Problem**	**Main conclusions**
Kim et al. (2016)	Dynamic pricing and energy consumption scheduling problem in the microgrid.	An RL-based dynamic pricing algorithm is shown to effectively work without a priori information about the system dynamics. The proposed energy consumption scheduling algorithm further reduces the system cost thanks to the learning capability of each customer. The algorithm is shown to perform better than a myopic optimization approach that considers only the immediate reward.
Sato (2016)	Categorization in consumer behavior marketing.	An RL algorithm is applied to categorize consumers using a simulation model of the marketing process. The algorithm is found to effectively optimize the categorization process.
Varela et al. (2016)	Investment decision support system.	A Q-learning based RL system is found to allow continuous learning based on decisions proposed by the system itself. This technique has several advantages, like the capability of decision-making independently of the learning stage, the capacity of adaptation to the application domain, and a goal-oriented logic.
Strydom (2017)	Bank portfolio optimization given uncertainty in both credit and liquidity risk.	A recursive learning method is developed to provide the bank with a trading signal to dynamically adjust the wholesale funding mix as the macroeconomic environment changes. A simulation is used to train and test the algorithm. The RRL method is found to dynamically adjust the trading strategy over the projection period and provide a higher average return compared to more traditional stochastic linear programming method.

Note: main conclusions are summarized by the author.

Appendix B

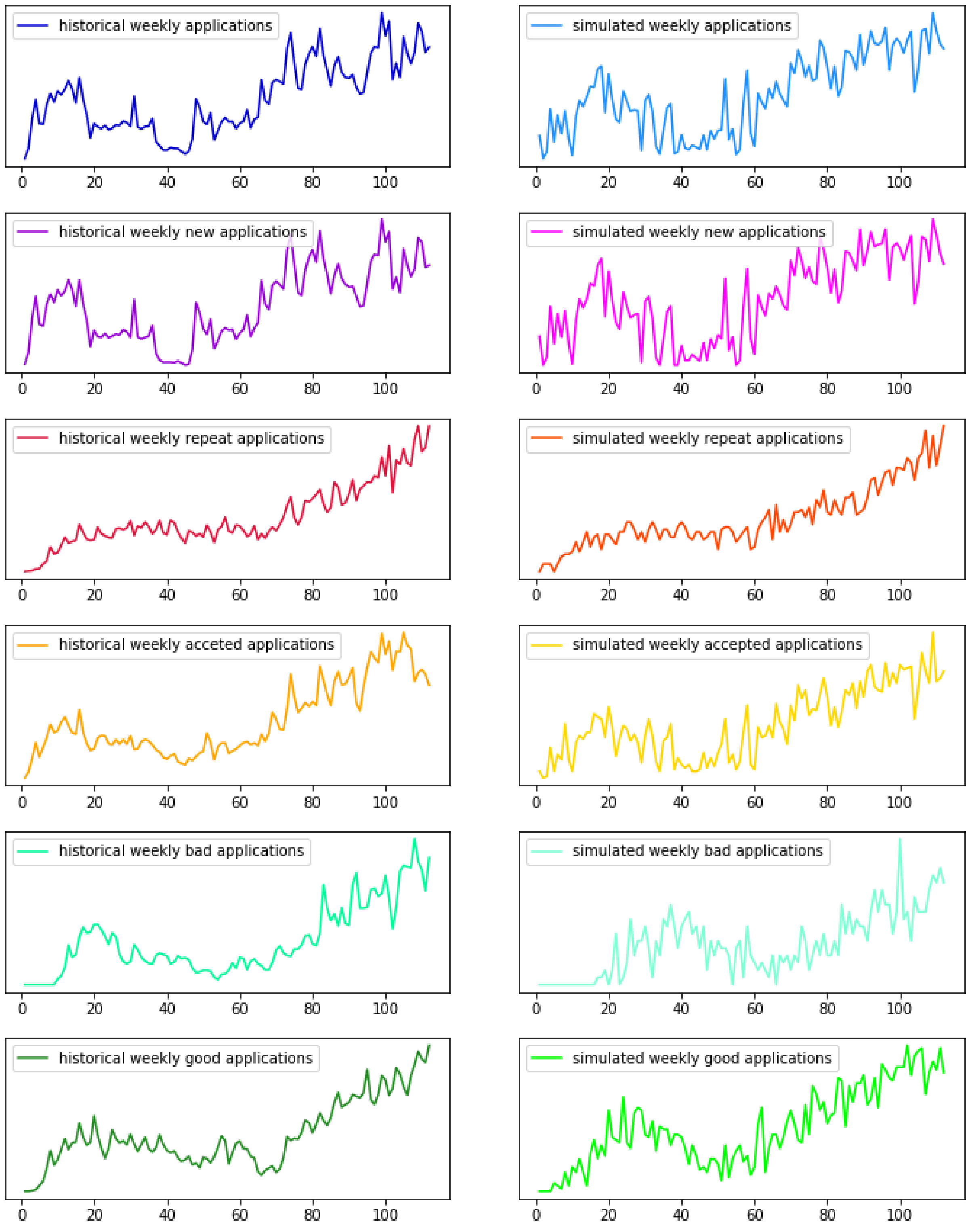

Figure B-1. Examples of historical and correspondent time series samples generated by the simulation.

Notes: the horizontal axis shows the number of the week; the vertical axis shows the number of credit applications; the tick marks of the vertical axis were removed because of confidentiality reasons.

Table B-1. Example of data generated every week of the simulation.

id	repeat	week	mature_at	debt	bad	bad_at	late_payment	late_payment_at	sum	duration	profit	score	accept
new_45_1	FALSE	45	57	0	FALSE	NA	FALSE	NA	214	12	69.26	53.02	TRUE
new_45_2	FALSE	45	57	0	FALSE	NA	FALSE	NA	354	12	117.62	27.89	FALSE
new_45_3	FALSE	45	49	0	TRUE	58	FALSE	NA	76	4	18.16	9.25	FALSE
new_45_4	FALSE	45	57	1200	FALSE	NA	FALSE	NA	214	12	69.26	41.32	FALSE
new_45_5	FALSE	45	57	0	TRUE	66	FALSE	NA	354	12	117.62	15.67	FALSE
new_45_6	FALSE	45	57	0	FALSE	NA	FALSE	NA	354	12	117.62	50.42	TRUE
new_45_7	FALSE	45	57	0	TRUE	66	FALSE	NA	354	12	117.62	65.53	TRUE
new_45_8	FALSE	45	49	1200	TRUE	58	FALSE	NA	76	4	18.16	39.38	FALSE
new_45_9	FALSE	45	49	0	TRUE	58	FALSE	NA	351	4	85.94	5.31	FAE
new_45_10	FALSE	45	57	0	TRUE	66	FALSE	NA	354	12	117.62	4.04	FALSE
new_45_11	FALSE	45	49	1200	FALSE	NA	FALSE	NA	351	4	85.94	-4.26	FALSE
new_45_12	FALSE	45	56	0	TRUE	65	FALSE	NA	87	11	30.10	-1.34	FALSE
new_45_13	FALSE	45	49	360	TRUE	58	FALSE	NA	351	4	85.94	43.75	FALSE
new_45_14	FALSE	45	49	252	FALSE	NA	FALSE	NA	76	4	18.16	41.18	FALSE
new_45_15	FALSE	45	49	1200	TRUE	58	FALSE	NA	351	4	85.94	13.24	FALSE
new_45_16	FALSE	45	57	0	FALSE	NA	FALSE	NA	354	12	117.62	56.01	TRUE
new_45_17	FALSE	45	57	144	TRUE	66	FALSE	NA	214	12	69.26	36.77	FALSE
new_45_18	FALSE	45	49	0	FALSE	NA	FALSE	NA	76	4	18.16	41.02	FALSE
repeat_45_1	TRUE	45	58	0	FALSE	NA	FALSE	NA	564	12	188.39	18.79	FALSE
repeat_45_2	TRUE	45	47	1200	FALSE	NA	FALSE	NA	92	2	18.68	13.18	FALSE
repeat_45_3	TRUE	45	49	720	TRUE	58	FALSE	NA	216	4	56.59	36.94	FALSE
repeat_45_4	TRUE	45	49	0	TRUE	58	TRUE	61	78	4	20.09	39.26	FALSE
repeat_45_5	TRUE	45	54	132	TRUE	63	FALSE	NA	346	13	117.31	39.66	FALSE
repeat_45_6	TRUE	45	54	720	TRUE	63	TRUE	72	346	13	117.31	54.70	TRUE
repeat_45_7	TRUE	45	48	0	FALSE	NA	FALSE	NA	75	3	19.28	48.06	FALSE
repeat_45_8	TRUE	45	58	48	FALSE	NA	FALSE	NA	1179	13	389.37	28.16	FALSE

Note: The acceptance threshold for the current example is 50. Variables week, mature_at, bad_at, late_payment_at mean the week (iteration) number of the event in the simulation, while the duration variable is defined in days. Profit measurements are not disclosed due to confidentiality issues.

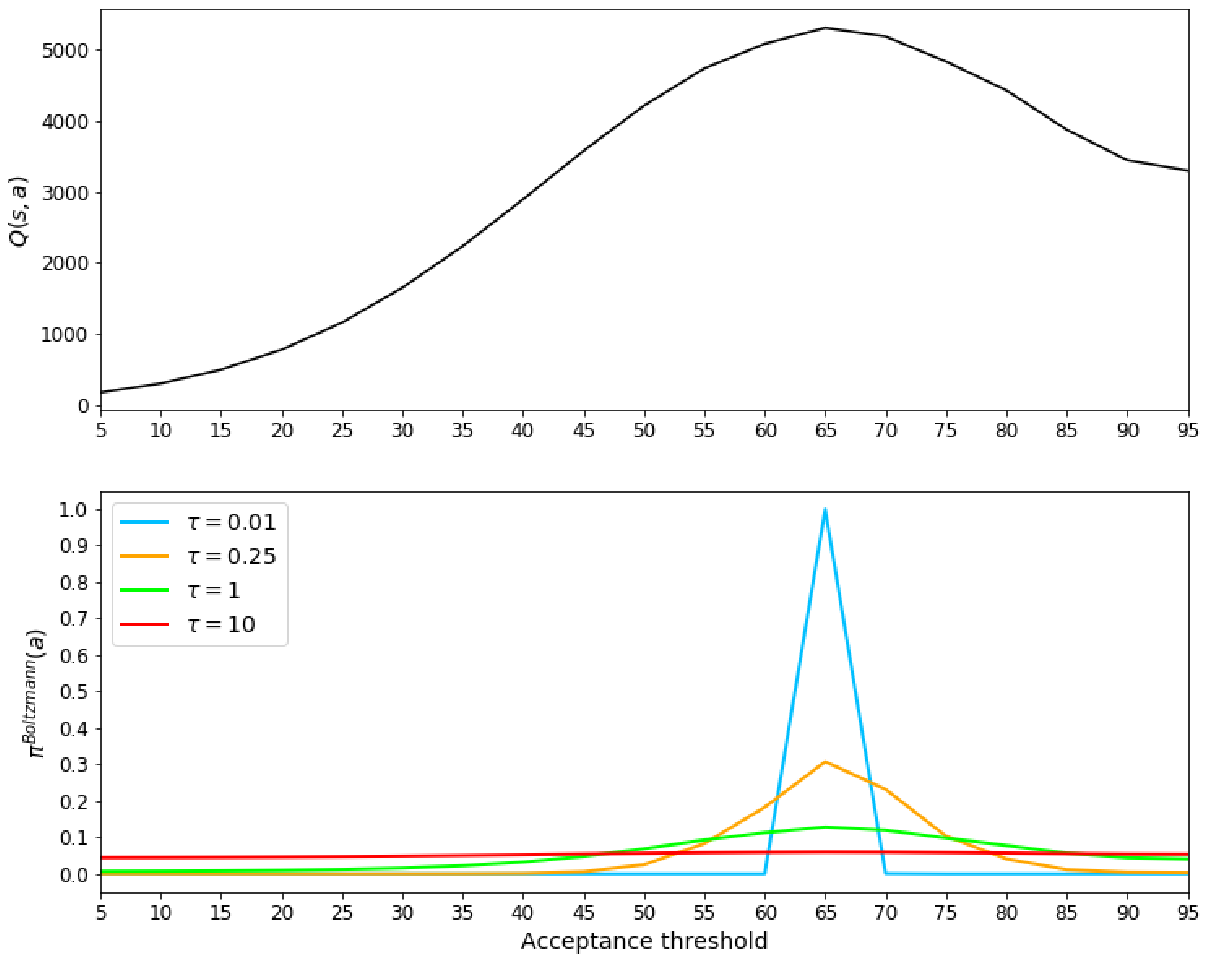

Figure B-2. Boltzmann-Q policy

Notes: the figure illustrates the Boltzmann-Q policy. The upper plot shows a Q-value function for a sample state (acceptance rate) s for every action (acceptance threshold) a. Those are used in equation (13) to generate a Boltzmann probability distribution based on a temperature parameter τ. The lower plot shows the generated probability distributions for a range of temperature parameters. One can see that low τ leads to a greedy policy: choosing the action with the highest value with probability 1, while high τ leads to a random policy: choosing any action with a probability of approximately 0.05. Parameter values in between lead to choosing actions with higher Q-values with higher probability compared to actions with lower Q-values.

Appendix C

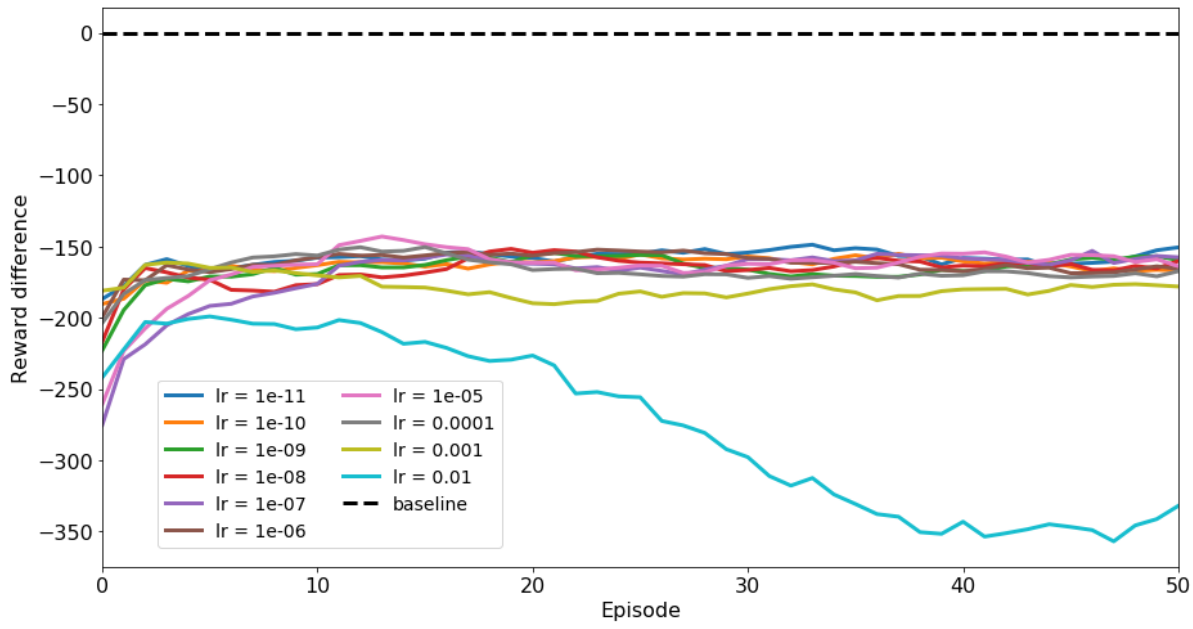

Figure C-1. Average training episode reward difference for various learning rate parameters

Notes: the graph shows the training reward difference (displayed on the vertical axis) – difference between episode profits received by the RL agent using train policy after training for n episodes (n displayed on the horizontal axis) using various learning rates (denoted by 'lr') and the baseline profits received using traditionally optimized policy. Profit is measured in thousands of euros. The reward difference is averaged across 5 runs of 50 episodes per each.

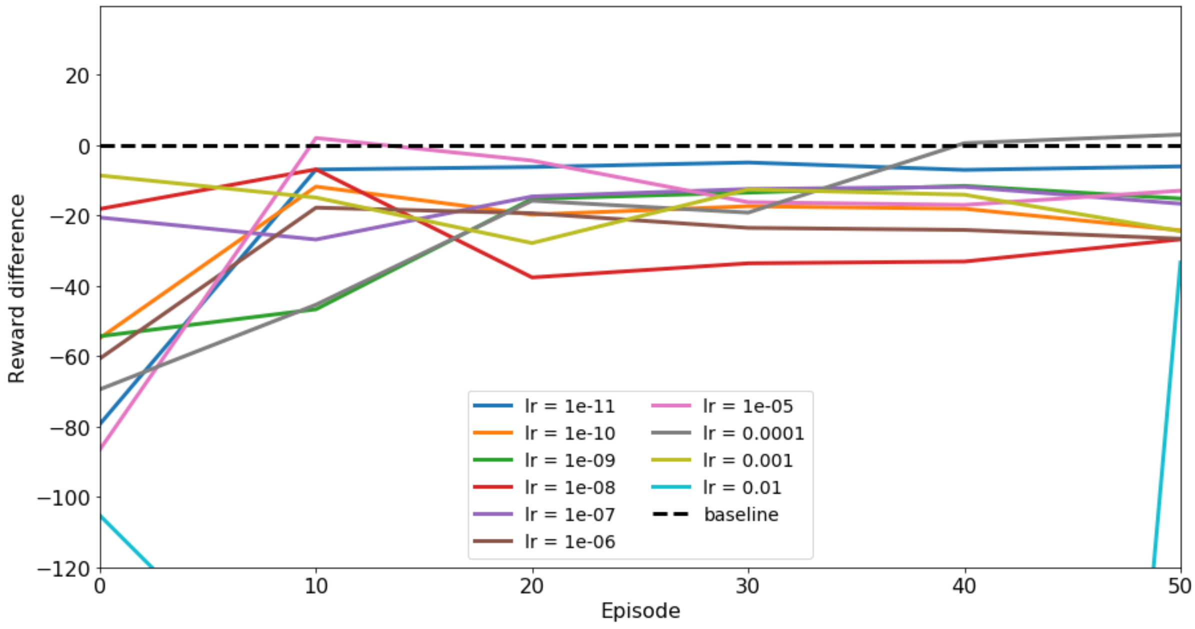

Figure C-2. Average test episode reward difference for various learning rate parameters

Notes: the graph shows the test reward difference (displayed on the vertical axis) – difference between episode profits received by the RL agent using test policy after training for n episodes (n displayed on the horizontal axis) using various learning rates (denoted by 'lr') and the baseline profits received using traditionally optimized policy. Profit is measured in thousands of euros. The reward difference is averaged across 5 runs of 50 episodes per each.

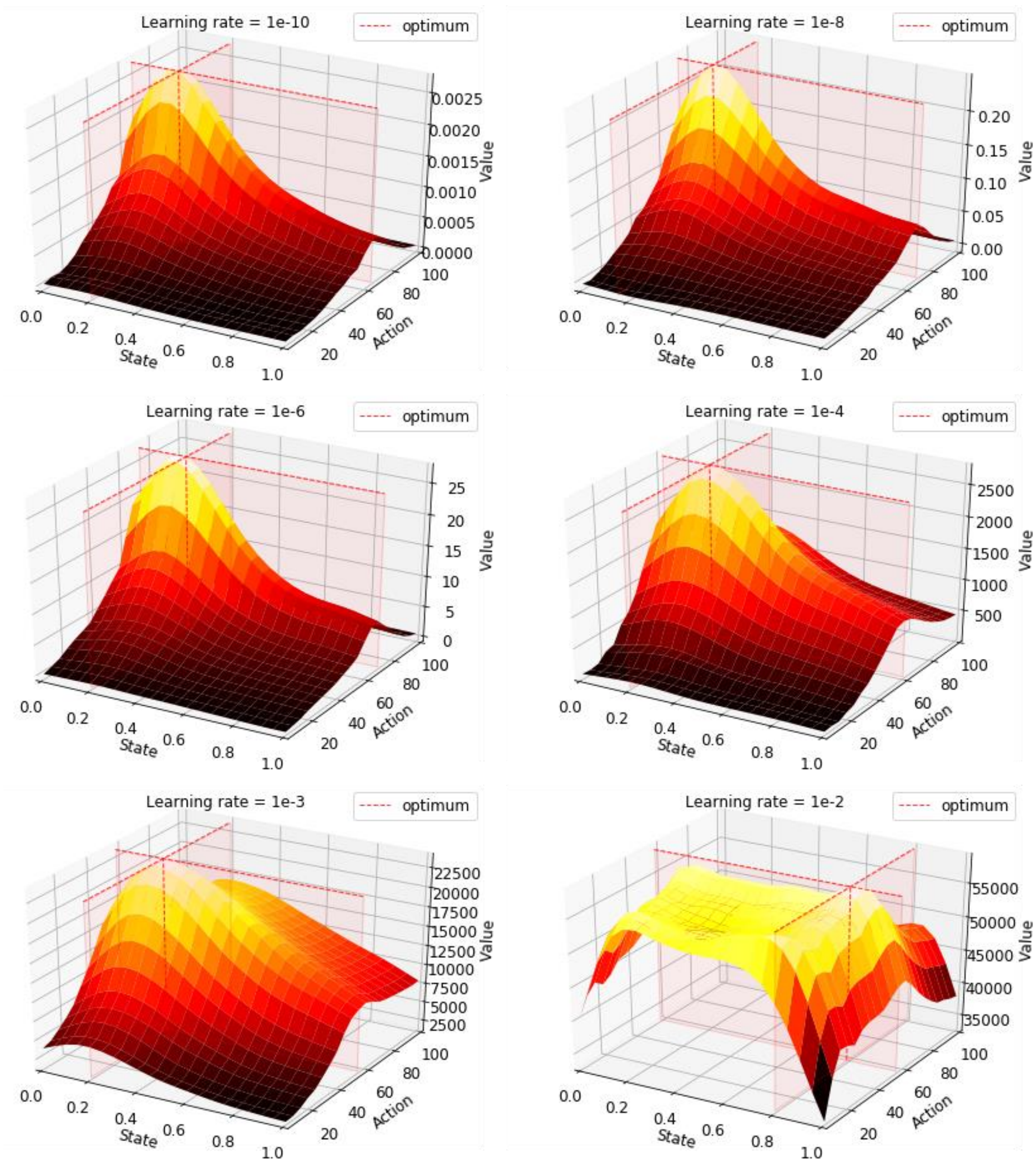

Figure C-3. Sample value function models after 50 episodes of training for various learning rate parameters

Note: State denotes the application acceptance rate during the previous week, action denotes the acceptance threshold for the following week, value is the prediction of the value function model for a particular state-action pair, optimum shows the state-action pair that corresponds to the highest value in the state-action space.

Table C-1. Reinforcement learning experiment timing

Computational chunk	Duration (hours:minutes:seconds)
Average week length	00:00:02.70
Average train episode length	00:05:26.25
Average test episode length	00:04:24.20
Average distorted episode length	00:06:05.13
Sample train experiment length	18:39:04.71
Sample test experiment length	11:13:35.94
Real episode length	02:36:08.50