Adaptive Machine Learning: An application to credit risk modeling

Hui Chen
MIT Sloan School of Management
huichen@mit.edu

Antoine Didisheim
Department of Finance
HEC Lausanne, University of Lausanne
antoine.didisheim@unil.ch

Simon Scheidegger
Department of Finance
HEC Lausanne, University of Lausanne
simon.scheidegger@unil.ch

March 27, 2020

Abstract

We investigate a machine learning decision-making problem in which the data generating process itself is adaptive to the algorithm. We consider a classification problem in which agents (borrowers) can perturb their data, trading off improved classification outcome by the algorithm against the potential cost of cheating. We propose a robustness scheme through a repeated game played between lenders and borrowers. Specifically, the lender proposes an algorithm to screen loan applications, while borrowers adapt to the model by falsifying their data. Perturbation by the borrower can potentially lead to an improvement in her application outcome. The cost of cheating is being caught, where the probability of fraud detection is increasing in the magnitude of the perturbation. We show that the repeated game converges to a robust decision-making function under various set-ups: including non-linear input and heterogeneous agents' risk aversion. If agents falsify their input to improve classification (linear perturbation) systematically, robust algorithms can reconstruct the original information from perturbed data. However, if the incentive to cheat is non-linear, for example, if only agents who can change their final classification do falsify their input (non-linear perturbation), some information is irrevocably lost by this perturbation process. We also investigate the robustness of the algorithm with respect to endogenous perturbations when the model is underspecified or overspecified. We do so by applying a simple logistic model to non-linear data and a universal approximator (neural network) to linear inputs. We show that underspecified models can converge to a stable, robust solution even when the input is highly non-linear. On the other hand, overspecified models are shown to be less robust to non-linear perturbation schemes.

Keywords: Deep Learning,

JEL classification:

1 Introduction

Recently, quantitative loan screening has evolved from using linear discriminant and logistic classifiers to complex models like deep neural networks. Recently, the role of these quantitative tools has started to shift from solely being a support to human decisions to the one of being the centerpiece of fully automated decision processes (Zhao, 2017). More generally, machine learning-based decision-making in finance is on the rise, with activities ranging from robot financial advisors to automatic trading algorithms.

In this context, the question of model robustness to endogenous perturbations of the inputs'[1] distribution is particularly relevant, especially since the adversarial machine learning literature has shown that neural networks can be very sensitive to adversarial attacks (see, e.g., Szegedy et al., 2013; Goodfellow et al., 2014; Papernot et al., 2017).[2] This weakness of neural networks can be seen as especially important in financial contexts. Indeed, in economic theory, robust decision rules are often defined in terms of an equilibrium between competing rational agents maximizing their respective utilities (see, e.g., Hansen and Sargent, 2008). If a prospective borrower's utility can be increased by receiving a loan, and if this borrower is fully rational, she should adapt her behavior to the introduction of a new screening function by the lender. If we further assume that the lender would lose utility by according a loan to a borrower solely because she adapted to his decision rule, it also follows that the lender should, in turn, adapt to the borrower's behavior. In such a context, a robust screening function is defined as one which would satisfy a Walrasian equilibrium between a lender and his borrowers.

In this paper, we aim at connecting these two strands of literature. We study the robustness of machine learning models when *all* agents rationally adapt to the algorithm's introduction into the economy. We simulate a lender and some borrowers playing a dynamic game. The lender wants to screen out *bad loans*[3], and the borrowers strategically adapt to the introduction of the screening rule. With these simulations, we study the consequences of either using linear logistic classifiers or deep neural networks to predict *bad loans* when the borrowers' data is endogenously perturbed by a strategic reaction to the algorithm. Next, we use data from the Lending Club[4]—a peer to peer online lending platform—to estimate the economic and welfare costs of non-robust algorithms. Specifically, we wish to answer the following four strands of questions:

i) Can a repeated game between the borrowers who adapt to the lender's model and the lender who adapts in turn to the perturbed input converge towards a stable equilibrium? Can this equilibrium be reached when the lender uses a neural network or only when he uses a linear logistic classifier? If there is an equilibrium, under which conditions is the adaptive behavior of the borrowers costly to the lender?

ii) How does the adaptive behavior of the borrowers impact the perturbed variables' predictive power?[5] Moreover, under which conditions can one variable loose all its predicting power as a

[1]We define input here as the set of predictive variables used by a quantitative model to estimate credit-score risk or similar quantities used to make a financial decision.

[2]These attacks are defined as small modifications of a neural network's input in order to fool the network into a misclassification. As an example, in machine vision—that is, machine learning applied to visual inputs—an adversarial attack can be defined as a slight modification of a picture which would not impede human identification of its content, but would fool a deep neural network. In this particular context, it has been shown that deep neural networks could be tricked into misclassification by modifying a single pixel (Su et al., 2019).

[3]Bad loans are usually defined in the literature as loans with a high probability of default, partial default, or delayed payments.

[4]https://www.lendingclub.com/

[5]We define a borrower's variables as some inputs, specific to a borrower which has some predictive power over her probability of being a *bad loan* from the lender's point of view.

consequence of borrowers' rational adaptation?

iii) In equilibrium, can underspecified models such as a linear logistic classifier outperform overspecified models like deep neural networks? If the true *pre-adaptation*[6] relationship between the borrowers' variables to their probability of default is non-linear, then linear logistic classifiers are underspecified, whereas deep neural networks[7] are overspecified. While it is clear that overspecified models will outperform underspecified ones when the borrowers do not adapt to the algorithm, the question of what will happen if the borrowers do strategically adapt is non-trivial.

iv) What would be the economic and welfare costs of using non-robust algorithms to automate decisions?

To answer questions i)-iii), we use simulated data, as the repeated game with logistic regressions or neural networks cannot be trivially reduced to closed-form solutions. To carry out our numerical experiments, we assume that the borrowers' cost of perturbing their variables—that is to say, the cost of adaptive behavior, is a function of the magnitude of perturbation multiplied by the inverse covariance matrix of all borrowers' unperturbed variables.[8] This assumption allows us to model the cost of an increased probability for the borrower to be detected. Indeed, the said probability is likely to be high when the borrowers' perturbed variables are less likely to come from the same distribution as the unperturbed ones. With the same reasoning, this assumption can also be used to model the costs of behaving in a way that would create negative utility for a borrower in order to obtain a higher probability of obtaining a loan. An example of such a trade-off would be a borrower using social media in order to project the image of a good creditor instead of merely enjoying the activity for its own sake (Wei et al., 2016). Next, we assume that, for each loan application, the lender uses a model to estimate a probability of *bad loan*. A *bad loan* corresponds to a high likelihood of costing some utility to the lender, through default, partial default, or delayed payments. The lender then decides on a specific threshold, which defines what *bad loan* probability is large enough for him to screen out the applicants. Next, the lender receives a positive payoff for every good borrower who received a loan and a negative payoff for every bad one. The lender chooses his model's parameters and decision threshold in order to maximize his profit.

We model two distinct adaptive strategies for the borrower, which we name from here onward the *all* and the *improve* perturbation scheme. In the *all* perturbation scheme, borrowers solve a maximization problem in which they trade off the cost of perturbing their inputs versus a lowered *bad loan* predicted probability. In the *improve* perturbation scheme, borrowers solve the same maximization but then compare their new predicted probability to their original one. If the new one allows the borrowers to obtain a previously unobtainable loan, they perturb their variables. Otherwise, they submit their original unperturbed data to the lender. In other words, in the *improve* perturbation scheme, borrowers submit perturbed variables if and only if these perturbed data both solves the maximization problem of the *all* perturbation scheme and move their final classification from *bad* to *good*. Importantly, while in the *all* perturbation scheme, every borrower will perturb it's variables following the same rule, in the *improve* perturbation scheme, some agents end up perturbing their data, while others do not.

Finally, we define two underlying mappings between the borrower's original variables and their probability of being *bad loans*, a linear and a non-linear one. We investigate two different algorithms,

[6]We define the pre-adaptation in this paper as the static case, when the borrowers do not strategically adapt to the introduction of the lender's decision function.

[7]Neural networks are universal approximators and can, given enough data, approximate any function (see, e.g., Hornik et al., 1989; Hornik, 1991).

[8]By borrower's variables, we mean the input used by the lender's model to make his screening decision. These variables could range from the borrower's history of credit-card usage to his social media footprint, to the text submitted with his loan application.

two perturbation schemes, and two underlying mappings, totaling in eight combinations. For each of these, we simulate a repeated game between the lender and his borrowers, where the former first proposes a decision rule, and the latter adapts to it. We repeat these two steps until convergence.

We tackle question i) by showing that in all eight combinations mentioned, the repeated game does converge to a stable profit for the lender. We show that in the *all* perturbation scheme, the lender can perfectly adapt to the perturbations and get a profit equal to the one he would have collected without adaptive behavior. In the *improve* perturbation scheme, a portion of the lender's profit is definitely lost, i.e., the profit of the lender in equilibrium is lower than in the static case when borrowers do not strategically adapt to his decision function.

We answer question ii) in a two-fold fashion. First, we derive closed-form solutions for some simplified versions of the repeated game defined above to foster the intuition of the economic mechanism at work. We show that in the specific case where the lender uses an ordinary least square regression, and the borrowers follow the *all* perturbation scheme, the importance of a predicting variable is not impacted by the borrowers' adaptive behavior. The resulting perturbations simply move the variable linearly across all borrowers. To answer the more general cases, we use in a second step again our simulations. We show that a variable's predicting power will drop to zero if and only if the cost of perturbation is exceptionally high, or its original predicting power is extremely low.

We address iii) by comparing the lender's equilibrium profits in all eighth combinations of parameters described above. When the true underlying mapping is non-linear, and the borrowers follow the *all* perturbation scheme, overspecified models do outperform underspecified ones. However, when borrowers follow the *improve* perturbation scheme, the underspecified model seems to outperform the overspecified one. We attribute this weakness of sophisticated models to the well-documented sensitivity of deep neural networks to adversarial attacks (Szegedy et al., 2013). Moreover, we provide evidence for this hypothesis by showing (a) that the equilibrium profits in our simulation are noisier with neural networks than with logistic classifiers, and (b) that the way each individual borrower perturbs her input from one round to the next varies significantly more when the lender uses deep neural networks to screen out *bad loans*. We verify that our findings are stable across various neural network architectures by changing the number of layers and the neurons' activation type in our simulations.

Finally, we tackle iv) by using real data from the Lenders club (TODO: finish here when we have results).

The questions studied in this paper rely on the assumption that borrowers do adapt to quantitative credit-scoring rules or, more generally, to automate decision-making algorithms. Anecdotal pieces of evidence support this assumption, even before the rise of big data. The proliferation of online pages suggest that *tricks* and behavior changes to increase FICO credit score provides evidence of interest from borrowers to adapt to the quantitative methods.[9] Nonetheless, big data and improved computing capabilities are likely to increase the importance of this adaptive behavior further. While new data sources and ever-rising computing power allow the lenders to use larger sources of information to estimate probabilities of default, these data can be legally perturbed by the borrowers at both high and low costs. For example, Óskarsdóttir et al. (2019) showed how the phone call history of borrowers can be used to improve credit scoring, whereas Netzer et al. (2019) suggested that textual analysis performed on loan applications could be used to improve default predictions. Phone call habits provide a good example of a predicting variable which can be legally *perturbed* by calling people with a bad credit score less often. Obviously, such behavior would come at a high utility

[9] See e.g. https://www.myfico.com/credit-education/improve-your-credit-score, https://bettercreditblog.org/5-amazingly-simple-techniques-to-optimize-your-credit-score, or https://www.debt.org/credit/improving-your-score.

cost. Concurrently, the texts in loan applications provide a good example of a variable which can be perturbed at almost zero costs for borrowers.[10]

To summarize, we contribute to the current literature in a fourfold fashion. First, we highlight the importance of including rational adaptive behavior in the definition of robustness when dealing with automated financial decisions. This is especially relevant in an economy moving towards more automation and more common usage of big data. Second, we show that variables which may be perturbed by fully rational agents are likely to contain still predicting power as long as perturbation is costly. Third, we show that algorithms that may appear to outperform on a static data set may underperform once the model is implemented because of high sensitivities to adversarial attacks. Fourth, we use real data to quantify the economic and welfare costs of poorly estimating the risks linked to rational adaptive behavior. In all, and to the extent of our knowledge, we are the first to investigate the risks associated with the use of machine learning when the agents creating the data can rationally adapt to the algorithm's introduction into the economy.

The remainder of this paper is organized as follows. In section 2 we discuss this paper relationship to previous literature. In section 3, we present the dynamic of the repeated games used both on the simulated and real data. In section **??**, we present the simulated and real data. Section 5 discusses the results, and Section 6 concludes.

2 Literature

This paper is related to three main strands of literature. The first is the quantitative credit-scoring and loan screening literature, which develops and analyses statistical or machine learning techniques to estimate a borrower's credit quality. The second is the economic literature on asymmetric information in the context of lending and banking. The third is the machine learning literature on adversarial attacks, which studies neural networks and other sophisticated models' sensitivity to strategic perturbation of the models' inputs. In the subsections to follow, we describe these three streams in greater detail.

2.1 Credit scoring and loan screening

Both academics and as well as practitioners have studied the use of quantitative approaches to estimate credit risk ever since Durand (1941) first pointed out that the statistical tools which were introduced in Fisher (1936) could be used to discriminate between good and bad loans. Earlier work developed the idea of *credit scoring* with the aim of providing quantitative support to human decision-makers (see, e.g., Rosenberg and Gleit, 1994; Hand and Henley, 1997; Thomas, 2000). The usage of credit score rapidly became mainstream in credit applications. They first became popular in the late 1960s with the arrival of credit cards (Thomas, 2000), but quickly generalized to other areas of finance. For example, Carleton and Lerner (1969) investigated how credit scoring tools could be used to rate municipal bonds' risks, whereas Orgler (1970) showed how credit scoring, estimated with ordinary least square regressions, can be used in the context of commercial loans. The tools used in credit scoring quickly spread over a wide range of quantitative techniques, ranging from linear discriminant and logistic classifiers up to algorithmic approaches that including decision trees and neural networks (see, e.g., Rosenberg and Gleit, 1994; Bastani et al., 2016; Hand and Henley, 1997; Thomas, 2000). Most of these models proved to be useful tools to assist human decisions; however, they were not immediately considered to be as robust enough to develop fully automated

[10]Liang et al. (2017) showed that adversarial attacks, while usually researched in machine vision, are also efficient in the context of natural language processing.

decision processes. Indeed, Wiginton (1980) concluded the analysis of logistic classifiers and linear discriminants by noting that if *"the logit function performed considerably better than chance, but does not appear to make a significantly high proportion of correct classifications to warrant use for unaided decision-making"*. Similarly, in the late 2000s, comparatively simple techniques were still preferred over more sophisticated models. West (2000), who investigated the efficiency of neural networks in credit scoring, concluded that linear logistic classifiers were still the most accurate classifier.

More recently, significant improvements in computational power renewed practitioners' and academics' interest in the use of sophisticated machine learning techniques. Huang et al. (2007), for instance, investigated the efficiency of support vector machines for credit-scoring, whereas Gao et al. (2006) showed that with a proper training process, deep neural networks can significantly outperform simpler linear models in credit scoring applications.

The interest for sophisticated methods increased further with easier access to a large quantity of potentially relevant information: big data.[11] Wei et al. (2016) showed in a theoretical set-up how networks-based observations, as can be obtained from social media's historical data, could be used to improve a lender's predictions of his borrowers' default probability. Óskarsdóttir et al. (2019) showed empirically how integrating network information about the borrowers' phone calls could be used to significantly improve the machine learning models' ability to classify good and bad credit risks. Similarly, Netzer et al. (2019) used natural language processing to show that textual information contained in loan applications could be used by a machine to improve credit scoring.

Big data, combined with significantly improved computational power, finally allowed large companies like Ant Financial[12] to fully automate lending decisions (Zhao, 2017).

We contribute to the credit scoring literature in general, and in particular, to the use of big data and modern algorithms to loan screening by investigating the risks of using sophisticated machine learning model when borrowers can strategically adapt to the introduction of the new model.

We further contribute to this strand of literature by including concepts from game theory and the partial equilibrium literature. To the best of our knowledge, very few researchers so far attempted to use these concepts in the context of credit scoring. Bravo et al. (2015) used game theory to define a third class of credit risk, but did not investigate adaptive behavior and equilibria. To improve their estimation of credit risks, they split *bad* loans into two categories; *can't pay* and *won't pay*. They defined *won't pay* as agents for whom it is optimal to default, even though they could repay the loan. Wei et al. (2016), in their theoretical analysis of social network data in credit scoring, did study an equilibrium in which borrowers rationally react to the lender's decision rule. While this work is related to ours, they focus their analysis on a theoretical equilibrium, while we discuss how different statistical models would behave and perform when facing endogenous input manipulation by the borrowers.

2.2 Banking and lending with asymmetric information

Over the past few decades, economist have studied the effect of asymmetric information on lending activities. In the context of banking, the seminal work by Stiglitz and Weiss (1981) discusses how imperfect information can create *credit rationing*. A vast literature has expanded this idea to include, among other things, screening costs in partial equilibrium models (see, e.g., De Meza and Webb, 1987; Broecker, 1990; Freixas et al., 2007). The bulk of this literature modeled the said costs as an

[11] We define big data in this paper as some large data-set —usually in the order of terabytes—composed of data previously unusable or which were believed to be irrelevant to the financial problem: data about user's social media activities, phone call habits, or texts which could not be quantitatively analyzed before improvements in natural language processing, etc.

[12] Formerly Alipay, Ant Financial is an affiliate company of the Alibaba Group.

exogenous parameter or a function in a complex model. We are, to the extent of our knowledge, the first to study loan screening in a Walrasian equilibrium. We contribute to this second stream of literature by showing how an equilibrium can arise even when the lender uses a neural network to screen-out bad loans, or when the borrowers follow a non-linear perturbation scheme.

2.3 Machine learning and adversarial learning

The machine learning literature on adversarial attacks studied how deep neural networks can be fooled by a relatively small modification of the model's inputs. In machine vision, it has been shown that a well-performing neural network classifier could be fooled by strategically modifying a single pixel of a photo (see, e.g., Su et al., 2019), while Liang et al. (2017) showed that adversarial attacks are equally efficient in natural language processing.

Szegedy et al. (2013) were the first to report that neural networks were highly sensitive to these kinds of attacks. The authors suggested that this weakness might be explained by the highly non-linear nature neural networks. This view was later challenged and, to the best of our knowledge, no clear consensus on this matter has yet emerged (see, e.g., Goodfellow et al., 2014). Szegedy et al. (2013) also noted that the same attacks worked on similar network trained on a different subsample of the same data set. This result strongly implies that the network attacks were *transferable* from one network to another. This *transferability* property of adversarial attacks was later confirmed by several papers (see, e.g., Goodfellow et al., 2014; Papernot et al., 2017).

The machine learning literature has split adversarial attacks into multiple categories. Poisoning attacks are defined as malicious attempts to introduce falsified data into the data set that is used to train the model (see, e.g., Yang et al., 2017; Jagielski et al., 2018). The attacks more relevant to this paper are the so-called white-box and black-box attacks. Both attacks occurred at inference, i.e., by manipulating the input given to the model when it is queried for a prediction. However, while white-box attacks assume the assailant knows both the machine learning model and its parameters' values, black-box attacks occur when the aggressors only know the model type (i.e., if the model is a neural network or a logistic classifier, etc.). In the context of loan screening, black-box attacks are more realistic as strategically adapting borrowers are not likely to know in detail the algorithm used by the lender. However, a growing body of research has shown that deep neural networks are surprisingly vulnerable to black-boxes adversarial attacks. Tramèr et al. (2016), Juuti et al. (2019), and Takemura et al. (2020), for example, studied extraction attacks in which an adversary obtain a new model whose performance is equivalent to that of a target model by querying it. Fredrikson et al. (2015), and Zhang et al. (2019) studied model inversion attacks in which access to the model is maliciously used to infer information about the training data.

Various attempts have been made to make deep neural networks robust to adversarial behavior. Perhaps the most intuitive approach was the so-called "sampled enhancement", in which the training sample is enhanced by adding adversarial examples to it (see, e.g., Huang et al., 2015; Tramèr et al., 2017; Hosseini et al., 2017). In order to make this process efficient, a vast literature has developed computationally efficient ways to create adversarial samples (see, e.g., Moosavi-Dezfooli et al., 2016; Kurakin et al., 2016; Papernot et al., 2016; Cisse et al., 2017; Su et al., 2019). Various other methods have been suggested to improve the neural networks' resistance to other adversarial attacks including specialized batch normalization (Rozsa et al., 2016), input compression (see, e.g., Dziugaite et al., 2016; Gao et al., 2017), detection of adversarial sample (Xu et al., 2017), and pre-processing inputs with a generative adversarial network (Kurakin et al., 2018). While all of these methods have shown promising results, at the time of writing, no method was perceived by the literature as a panacea against adversarial attacks.

We contribute to this third strand of literature by being the first, to the extent of our knowledge, to link this well-documented sensitivity to adversarial attacks to economic applications where agents can rationally adapt to the algorithm implementation.

3 Model

We model a competitive game between a lender and his borrowers. A borrower is defined by the quality of his credit risk (either *good* or *bad*)[13] and a set of variables that are mapped by some function to a probability of *bad* credit risk. The lender aims at filtering out *bad* borrowers in order to maximize his profits. To do so, he screens out borrowers for whom he estimates that the probability of *bad* is above some threshold. The lender has to commit both to a model to estimate the probability of *bad* from the borrowers' variables and the threshold above which he considers the risk to be too high. Both *good* and *bad* borrowers can rationally adapt to the chosen model and threshold by perturbing their variables at a given cost.

In this section, we define the borrower's and lenders' optimization problems. Next, we will present the borrowers' characteristics before defining the repeated game used to make the lender's algorithm robust to the borrower's strategic perturbations.

3.1 Lender's Problem

We start the exposition of the problem with the lender, who aims at maximizing his profits by screening out *bad* risks.

Let $\phi(x_i|\tau,\theta)$ be the lender's decision function. $\phi(x_i|\tau,\theta)$ takes as an input the variables of a borrower i and yields back a binary variable equal to 1 if the lender agrees to lend to borrower i. This screening function can be split into two parts. In the first part, the probability of *bad* is estimated by a function $f(x_i|\theta)$, where θ is a set of parameters. In a second step, the lender defines his decision threshold τ. Every borrower with an estimated probability of *bad* higher than τ is screened out, while all others receive a loan. The function $\phi(x_i)$ is defined as:

$$\phi(x_i|\tau,\theta) = \mathbb{1}_{f(x_i|\theta)<\tau}, \tag{1}$$

where $\mathbb{1}_x$ is a binary variable equal to 1 if the condition x is satisfied, 0 otherwise.

When he estimates the parameters in θ and choose the threshold τ, the lender observes a set of historical data consisting of N pairs $[x_i; y_i]$, where y_i is a binary variable equal to 1 if the borrower i proved to be a *bad* risk, and 0 otherwise. x_i is a vector of variables of dimension 3 with some predictive power of y_i. In other words, the true probability of being *bad* is defined by $f_{true}(x_i) = p_i + \epsilon$, where $f_{true}(\cdot)$ is the true function mapping the borrowers unperturbed input to its probability of being a *bad* risk. p_i is the true probability that borrower i represents a *bad* risk for the lender, while ϵ is some uncorrelated random noise.

A naive lender—that is, a lender who assumes the borrowers would not falsify their input once the function $\psi(\cdot)$ is defined— would first estimate the set of parameters θ to best map the borrowers' variables to a probability of default. In this paper, this estimation is done by minimizing the cross-entropy loss (see, e.g., Goodfellow et al., 2014).

$$\hat{\theta} = \arg\min_{\theta} - \sum_{i=1}^{N} y_i \log(f(x_i|\theta)) + (1-y_i)\log(1-f(x_i|\theta)). \tag{2}$$

[13]We define a *bad* credit-risk as any loan where the lender, on average, lose money. This can be caused by default, partial default or delayed payments.

Once the set of parameters $\hat{\theta}$ is estimated, the lender has to define his decision threshold τ. We assume a risk-neutral lender who maximize his expected profits. Therefore, he estimates his optimal τ as:

$$\hat{\tau} = \arg\max_{\tau} \sum_{i=1}^{N} \mathbb{1}_{f(x_i|\hat{\theta})<\tau} G - \mathbb{1}_{f(x_i|\hat{\theta})<\tau} L, \tag{3}$$

where G represents the profit made when lending to a *good* borrower, and L the loss occurred when lending a *bad* one.

3.1.1 Lender's sophistication

We want to distinguish between simple models and deep neural networks in order to study their sensitivity to adversarial attacks in this context. To do so, we define two levels of the lender's sophistication.

We define the lender's sophistication as the complexity of the functional form $f(\cdot|\theta)$ used to approximate the true mapping $f_{true}(\cdot)$. A *simple* lender is defined as using a linear logistic classifier. In this case, θ represents the comparatively small set of linear weights and the constant term. A *complex* lender uses a neural network, and θ represents a very large set of weights and constants.

3.2 The borrower's problem

We define now how and when a borrower strategically adapts to the introduction of the lender's decision function $\phi(\cdot|\theta, \tau)$. First, we define how a borrower with an initial vector of variables x_i finds her optimal perturbed vector m_i. Second, we discuss *perturbation schemes* which define the conditions under which a borrower decides to submit her perturbed input m_i or her initial variable vector x_i.

3.2.1 Optimal perturbed variable

A borrower observes the lender's function $\phi(\cdot|\theta, \tau)$ and can strategically adapt to it by falsifying her original x_i into a perturbed vector m_i. The optimal values of m_i are defined by a trade-off between maximizing the borrower's chances of getting a loan and a cost proportional to the probability of getting caught.That is,

$$\min_{m_i} \left[f(m_i|\theta) + \lambda_i \left(m_i - x_i\right)' \Lambda^{-1} \left(m_i - x_i\right) \right]. \tag{4}$$

Equation (4) represents the minimization problem which the borrower i solves to find her optimal falsified variable vector m_i. λ_i represents the borrower's aversion to risk, while Λ^{-1} is the inverse covariance matrix of the full input matrix composed of all individual x_i. The first part of equation (4) is the probability estimated by the lender that the borrower i is a *bad* credit risk. The second part of equation (4), $\lambda_i \left(m_i - x_i\right)' \Lambda^{-1} \left(m_i - x_i\right)$, increases proportionally to the magnitude of falsification $m_i - x_i$. Furthermore, the inverted covariance Λ^{-1} insures that this cost increase is sharper when the final variable vector m_i is less likely to have originated from the same distribution as x_i.

3.2.2 Perturbation schemes

While the borrowers' true benefit of perturbations is discrete—i.e. either getting or no getting access to credit—the costs and benefits modeled in equation (4) are continuous. The benefits of the perturbations are based on the estimated probability of *bad*, $f(x_i|\theta)$, and not on the decision function, $\phi(x_i|\tau, \theta)$.

To remedy this shortfall, we introduce the concept of *perturbation schemes*. This scheme defines when a borrower decides to submit her optimal perturbed vector m_i or her original unperturbed vector of variables x_i. We define two perturbation schemes: *all* and *improve*.

Under the *all* perturbation scheme, all borrowers simply solve equation (4) and submit the perturbed input m_i to the lender. In this set-up, the perturbation is linear across borrowers, and all of them do perturb their inputs, even when their initial probability of *bad* is as high as 99.9% or as low as 0.1% and the decision threshold τ is set to 50%.

In the *improve* scheme, we assume that the agents only perturb their inputs if it improves their final classification, that is $\phi(m_i) > \phi(x_i)$. This implies that the borrowers' input perturbation is computed in two steps. First, every agent solves equation (4) to find their perturbed variable vector m_i. Since the borrowers know $f(\cdot|\theta)$, they can then estimate their probability of default if they report m_i to the lender. If this perturbed probability is below the decision threshold of the lender, they falsify their data and present m_i to the lender's algorithm. If their perturbed estimated probability of default is still above the threshold, they do not cheat and report their true x_i to the lender. Similarly, if a borrower estimated that the *bad* probability was already below the threshold τ without perturbing her input, she does not take any risk and simply submit the true vector x_i to the lender. In other words, if perturbation allows her to move her classification from *bad* to *good*, a borrower will perturb her input. Else she simply submits her original variable vector x_i. This perturbation scheme is the most consistent with our original assumptions that the benefits of perturbations are weighted against a cost of perturbation proportional to the probability of getting detected. Indeed, if the probability of being detected is non-zero for any $m \neq x_i$, adapting to the decision rule, $\phi(\cdot)$ only makes sense if it is counterbalanced by a financial gain.

3.3 Simulated data

In our simulations, we assign a set of initial variables x_i to each borrower. We will now discuss the underlying distribution from which x_i is drawn. Then we will present the true functions which map x_i to the probability of *bad* credit risk, as well as the true quality of credit of each simulated borrower.

3.3.1 The distribution of the borrowers' variables

In order to analyze how different perturbation costs do influence our repeated game—research question (ii), we define two underlying distributions from which to draw the original variables x_i: *σ-homogenous* and *σ-heterogeneous*.

In the *σ-homogenous* set-up, all 3 dimensions of x_i are drawn from a random normal variable with mean 0 and a standard deviation of 1—that is,

$$x_i \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right), \tag{5}$$

In the *σ-heterogeneous* set-up is similar to the homogenous set-up, but the standard deviation of the first variable has been set to 10.

$$x_i \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 10 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}\right) \tag{6}$$

3.3.2 The mapping function

Each one of our simulations is defined by an original function $f_{true}(\cdot)$ which maps the borrowers original unperturbed variable vector x_i to their respective probability of being *bad* loans.

We explore two different $f_{true}(\cdot)$, a *linear* and a *non-linear* one. In the *linear* case, the true function is defined as

$$f_{true}(x_i) = \frac{1}{1 + \exp -x_i w}, \tag{7}$$

where w is a vector containing one weight per input. All weights' values are set to 1 for simplicity.

In the *non-linear* set-up, we introduce a non-linearity to the first and second dimension of x_i. Let $x_i^{(d)}$ be the scalar value in the dimension d of borrowers' variable vector x_i:

$$x_i^{(1')} = max(0, x_i^{(1)}) - mean(max(0, x_i^{(1)})), \tag{8}$$

$$x_i^{(2')} = x_i^{(2)} x_i^{(2)} - mean(x_i^{(2)} x_i^{(2)}), \tag{9}$$

$$f_{true}(x_1, x_2, x_3) = \frac{1}{1 + \exp - \left(x_i^{(1')} + x_i^{(2')} \sum_{d=3}^{D} x_i^{(d)} \right)}. \tag{10}$$

When we initialize the simulation (see section 3.3.3) with a *linear* mapping, 50% of the borrowers are, on average, defined as *bad*. We normalization the variable 1 and 2 by their respective average to keep a roughly similar probability of *bad* credits in the *non-linear* set-up.

3.3.3 Simulation initialization process

In each simulation, the data is generated in three steps. We first randomly draw the borrowers' variables from the distribution ξ. Next, we apply the function $f_{true}(x_i)$ to these variables to get, for each borrower, a probability of being a *bad* credit. Finally, we randomly draw a value u_i, which is drawn from a $uniform(0, 1)$ for each borrower i. Each borrower is defined as a *bad* credit risks if and only if $f_{true}(x_i) \leq u_i$

3.4 Repeated Game

So far, we have defined the borrowers both by, (a) the random distributions from which they are drawn, and (b) the rule governing their strategic behavior. We also defined the lender's strategic behavior. We now turn our attention towards the repeated game played between these two types of agents in our simulations.

In each round t, the lender first updates his decision function $\phi(\cdot|\theta, \tau)$ using a set of N observations $\{x_i^{(t)}, y_i\}$ where $x_i^{(t)}$ is a vector of variables and y_i is a binary variable equal to 1 if the borrower was found to represent a *bad* credit risk for the lender. The game starts with the lender who solves equation (2) and (3) for the unperturbed input set $\{x_i^{(0)}, y_i\}$. Next, the borrowers observe $\phi(\cdot|\theta, \tau)$ and apply the perturbation scheme to strategically adapt their input to a perturbed vector $x_i^{(1)} = Pert(x_i^{(0)}, \hat{\theta}_t, \hat{\tau}_t)$.[14]

[14] In the *all* perturbation scheme defined in section 3.2.2, $x_i^{(1)}$ is equal to m_i for all borrowers. However, in the *improve* perturbation scheme, only some borrowers perturb their input to m_i, for the others, $x_i^{(1)} = x_i^{(0)}$ holds.

```
**Data**: $N$ historical pairs $[x_i; y_i]$
**Result**: $\hat{\theta}$, $\hat{\tau}$
initialization;
$\hat{\theta}_0 = \arg\min_{\theta} - \sum_{i=1}^{N} y_i \log(f(x_i^{(0)}|\theta)) + (1-y_i)\log(1-f(x_i^{(0)}|\theta))$ ;
$\hat{\tau}_0 = \arg\min_{\tau} \sum_{i=1}^{N} \mathbb{1}_{f(x_i^{(0)}|\hat{\theta}_0)<\hat{\tau}_0} G - \mathbb{1}_{f(x_i^{(0)}|\hat{\theta}_0)<\hat{\tau}_0} L$ ;
$t = 1$;
$\Delta\pi = \epsilon + 1$;
**while** $\Delta\pi > \epsilon$ **do**
    $x_i^{(t)} = Pert(x_i^{(t-1)}, \hat{\theta}_t, \hat{\tau}_t)$;
    $\hat{\theta}_t = \arg\min_{\theta} - \sum_{i=1}^{N} y_i \log(f(x_i^{(t)}|\theta)) + (1-y_i)\log(1-f((x_i^{(t)}|\theta))$ ;
    $\hat{\tau}_t = \arg\min_{\tau} \sum_{i=1}^{N} \mathbb{1}_{f(x_i^{(t)}|\hat{\theta}_t)<\tau} G - \mathbb{1}_{f(x_i^{(t)}|\hat{\theta}_t)<\tau} L$ ;
    $\pi_{t-1} = \sum_{i=1}^{N} y_i \log(f(x_i^{(t-1)}|\theta)) + (1-y_i)\log(1-f(x_i^{(t-1)}|\theta))$;
    $\pi_t = \sum_{i=1}^{N} y_i \log(f(x_i^{(t)}|\theta)) + (1-y_i)\log(1-f(x_i^{(t)}|\theta))$;
    $\Delta\pi = \pi_{t-1} - \pi_t$ ;
    $t + 1$;
**end**
$\hat{\theta} = \hat{\theta}_t$;
$\hat{\tau} = \hat{\tau}_t$;
```

Algorithm 1: The above's algorithm present formally the repeated game played between the lender and the borrowers.

In the second round, the lender observes $\{x_i^{(1)}, y_i\}$ and defines a new decision function $\phi(\cdot|\theta,\tau)$. All borrowers then observe this decision function and adapt their respective inputs $x_i^{(1)}$ to $x_i^{(2)}$ by applying the perturbation scheme.

The game is repeated until convergence. To define said convergence, we compute the lender's profit in round t with the original variables perturbed t and $t-1$ times—that is, $x_i^{(t-1)}$ and $x_i^{(t)}$. If the difference between these two profits is zero, the lender has no incentive to update his decision function. Since the borrowers were the last to update their perturbations, they have no incentive to redo it either: the game has converged.

Algorithm (1) formally describes this process. Notice that since the borrowers move last in each round, the lender's profit at the end of the first round represents the profit of a naive lender who does not anticipate the strategic adaptation of his borrowers.

3.5 Numerical stability

We will now present the parameters we use in the simulations as well as the bootstrapping procedure we apply to obtain numerical stability.

In our numerical experiments, we create a total of $14,000$ borrowers.[15] When the lender minimizes equation (2) to find the optimal parameters in θ he has access to $12,000$ observations. $10,000$ samples are used as a training set, while the remaining $2,000$ observations are used as a validation set. The same $12,000$ simulated borrowers are also used by the lender when he solves equation (3) to find

[15] We chose this sample size through trial and error. We found that larger samples only marginally increased the precision of the lender's precision while it significantly increased computational cost. One simulation, with 20 game rounds run 20 times for numerical stability, take approximately 72 hours on a machine with 16 CPU.

his optimal threshold of τ. The remaining $2,000$ borrowers are used to compute the out of sample performance of the lender, measured as his profit. This profit is therefore estimated on this subsample, which was not used to optimize the function $\phi(x_i|\tau, \theta)$.

We run every experiment 20 times. When discussing a numerical result from these experiments, we show the mean values computed across simulations.

4 Lending club data

In this section, we briefly describe the US-based peer to peer lending company *Lending Club*.[16]. Then we describe the data provided by this firm, which we use to answer our fourth research question (see section 1).

Lending Club connects borrowers with individual lenders through an online platform. The prospective borrowers have to provide significant amount information to the firm, including a credit score like the FICO score and some personal information such as employment history or number of credit cards owned. The Lending Club uses this data, first to filter out bad loans and then to assign a grade to each accepted loan. This grade, together with the loan's term and some macroeconomic factor, is used to determine the interest rate through some publicly available formula. The Lending Club claims to have "*[...] internally developed an algorithm which analyzes the performance of borrower members and takes into account FICO score, credit attributes, and other application data [...]*"[17] to estimate the grade of each borrower.

The loan financing is not done by the Lending Club itself but by other clients of the company. These lenders can use the platform to see the loan applications, including a significant fraction[18] of the information provided by the borrower in their application, as well as the grade estimated by the Lending Club's algorithm. Each lender can then decide to fund part or all of any individual loan application.

The Lending Club has made available a large quantity of historical data. These include all loans funded by the platform from 2007 to 2018. Table 1 shows the number of loans in the dataset year per year. In total, the data contain 2,260,668 individual loans, and more than 98% of them were issued between 2012 and 2018. For each loan, the Lending Club provided the information available to the lender on the website—that is, some but not all of the borrowers' information used to define the loan grade, the amount demanded by the borrower, the loan term, and the grade defined by the proprietary algorithm. In addition, the data contain information about each loan performance—that is, the interest paid to date, the principal repaid to date, an indicator of default, and an indicator that the loan was charged off. This last variable indicated that the loan was unlikely to get repaid and that the Lending Club planned to sell the loan to other companies for a fraction of its face value. If the Lending Club already did sell the loan, they provided the amount raised, as well as the fraction of this resale amount taken as a fee by the Lending Club.

Unfortunately, the Lending Club only released partial data about the loan application they rejected. In consequence, we focus our analysis on the grading of accepted loans. The firm's algorithm automatically assigns one grade, from A (best) to G (worst). Figures 1a and 1b show the fraction of new loans assigned to each grade year per year.[19] In these figures, we can see that the grade

[16]https://www.lendingclub.com/

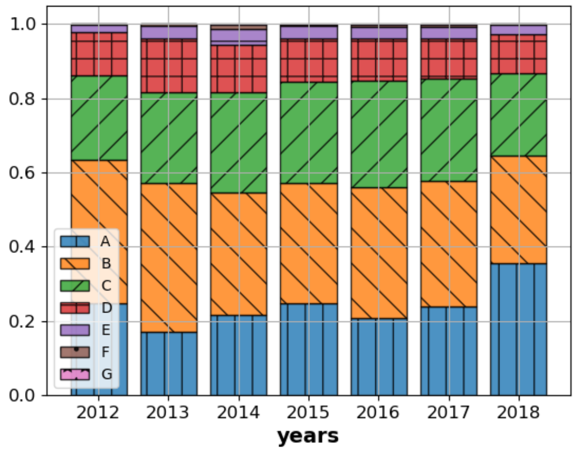
[17]https://www.lendingclub.com/foliofn/rateDetail.action

[18]**TBD Simon: I will put the real number there, but I need to log in LC, and for that, I need Hui to confirm my machine again, sorry.**

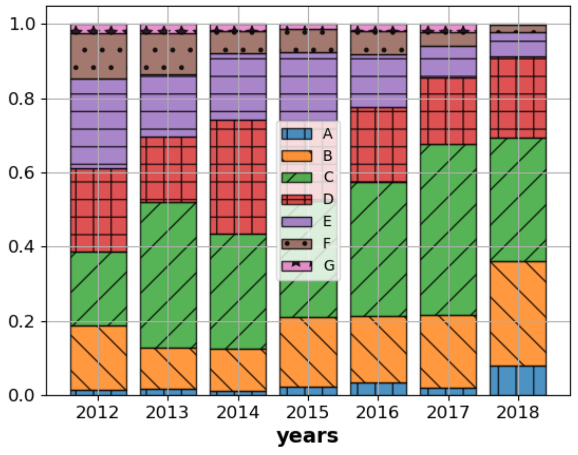
[19]We only display values for 2012, and above as the years 2007 to 2011 together contain only 21,721 individual loans, which translate into 1.88% of the total sample size.

	3 year loans	5 year loans	all loans	cumulative %
2007	603	0	603	0.03
2008	2393	0	2393	0.13
2009	5281	0	5281	0.37
2010	9156	3381	12537	0.92
2011	14101	7620	21721	1.88
2012	43470	9897	53367	4.24
2013	100422	34392	134814	10.21
2014	162570	73059	235629	20.63
2015	283173	137922	421095	39.26
2016	323495	110912	434407	58.47
2017	320419	123160	443579	78.09
2018	344671	150571	495242	100.00

Table 1: The above's table display the number of new loans issued by the Lending Club from 2007 to 2018.



(a) 3-year loans



(b) 5-year loans

Figure 1: In the above's figures, we display the percentage of new loans assigned to each grade year by year.

composition is not stable through time. More importantly, the fraction of loans with good grades seems to be increasing with time. Increased quality of the borrowers could cause this fact, but it could also be caused by a change in behavior of the borrowers who strategically adapt to the Lending Club classification algorithm. To distinguish between these two hypotheses, we need to see if the general quality of high-grade loans is diminishing. Unfortunately, the quality cannot be measured with certainty before the end of the loan's term. Since the loans are repaid either in 3 or in 5 years, we cannot measure the quality of the loan in the last three years of our data set. Therefore, the trend displayed in figures 1a and 1b can only be presented as anecdotal evidence that the borrowers may have strategically adapted to the lending club's algorithm.

Table 2 displays, performance indicators for each grade—that is, the percentage of loans flagged as default, the percentage of charged-off loans, the mean return on investment, the volatility of return on investment, and the mean return divided by the volatility of return.[20] These performance measures

[20]For each loan, we only have information about the total amount lent to the borrowers and the total dollar amount paid back by the borrower. To compute the return, we make the simplifying assumption that all payments to the

	default %	charged off %	$\bar{r}$	σ_r	$\bar{r}/\sigma_r$
grade					
A	0.001	3.43	0.0203	0.044	22.98
B	0.002	6.66	0.0259	0.063	16.28
C	0.003	9.70	0.0273	0.078	13.20
D	0.005	12.40	0.0287	0.094	10.96
E	0.006	13.42	0.0309	0.097	10.67
F	0.010	16.03	0.0315	0.101	10.26
G	0.000	12.13	0.0337	0.092	11.20

Table 2: The above's table display, for each grade, the percentage of loans which defaulted, the percentage of loans which were charged off. Similarly, for each grade, we display the average return on investment $\bar{r}$, the volatility of return σ_r, and the ratio for the two.

have been computed on the subsample of *completed loan*—that is, loans that were either flagged by the lending club as *fully paid* or *defaulted* and loans older than their term. In this table, we see that the percentage of defaulted and charged-off loans increased as the quality of the grade decrease. The only exception is graded G. the extremely small number of loans can explain this anomaly with this grade, i.e. only 0.5% of the total sample. The mean return is lower for higher grades, but the standard deviation of return is also significantly lower. The ratio of mean return over the volatility of return varies significantly from high grades to low grades.

Together, these descriptive statistics show that the proprietary algorithm of the Lending Club does classify loans into meaningful categories as risk indicators are higher for lower grades.

TODO: Expand this section when LC analysis is complete, and it becomes clearer what we need to introduce. Also, we may want to discuss moving the stat about changing grade into the intro as a motivation of our approach.

5 Results

In section 3, we presented the various theoretical set-ups for our simulations. We laid out two sophistication degrees for the lender, two underlying distributions for the borrowers' variables, two functions mapping these variables to a probability of *bad* risk, and two perturbation schemes. In the following subsections, we use these simulations and some analytical solutions to answer our first three research questions defined in section 1.

5.1 Robust algorithm

We start by answering question i). In the following subsection, we investigate whether or not the repeated game presented in algorithm (2) converges to a stable solution in all the eight simulation setups defined in section 3.

We define a stable solution as one where neither the lender nor his borrowers, would wish to change their respective strategies. At the end of each round, the borrowers are the last to update their perturbations. Therefore, if at the start of the next round, the lender has no incentive to update

borrower are made when the loan is issued, and all payments made by the borrower occurs at the end of the loan's term. Therefore, all Lending Club returns computed in this paper underestimate the true return for a prospective lender.

his strategy, the game has converged to an equilibrium. The lender has no incentive to update his strategy when his profit with the decision function $\phi(x_i^{(t)}|\tau^{(t)}, \theta^{(t)})$ is equal to his profit with the corresponding decision function $\phi(x_i^{(t)}|\tau^{(t-1)}, \theta^{(t-1)})$. In other words, if the lender does not improve his profit by updating his decision function, the game has converged.

We first investigate the cases where the true mapping of the unperturbed input to the *bad* probability is linear. The underlying distribution of the input x_i is homogeneous and follows equation (5). The lender uses a linear logistic classifier to estimate the probability of *bad*.

In figures 2a and 2b, we display the profits of the lender throughout the first five rounds of the repeated game. The x-axis shows the decision threshold τ in percentage points—that is, $x = 30$ means that the lender will give a loan to every borrower with an estimated probability of *bad* below 30%. On the y-axis, we show the average profit per borrower demanding a loan defined as

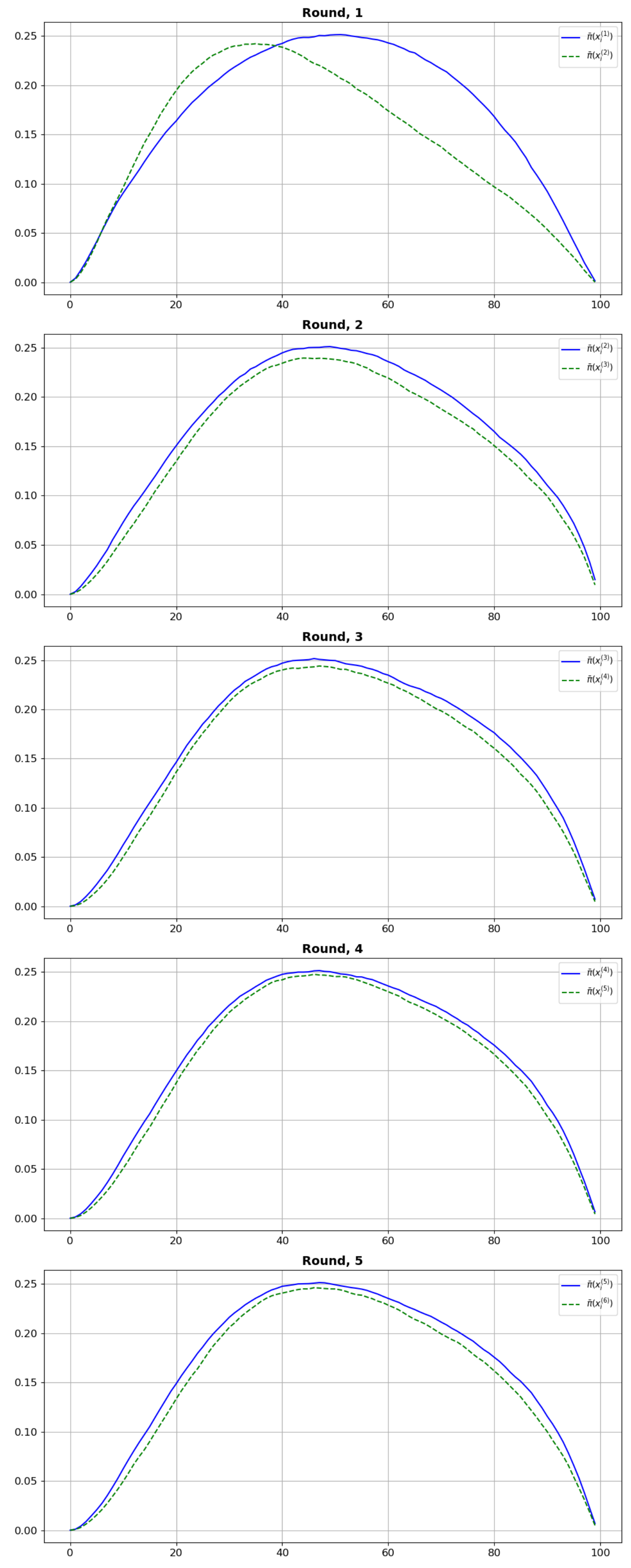
$$\bar{\pi}(x_i|\theta, \tau) = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}_{f(x_i|\theta)<\tau} G - \mathbb{1}_{f(x_i|\theta)<\tau} L. \tag{11}$$

Each subplot shows a different round t of the repeated game. For each t, the two lines represent the profit curves of the lender when the borrowers' data was perturbed $t-1$ times and t times, respectively.

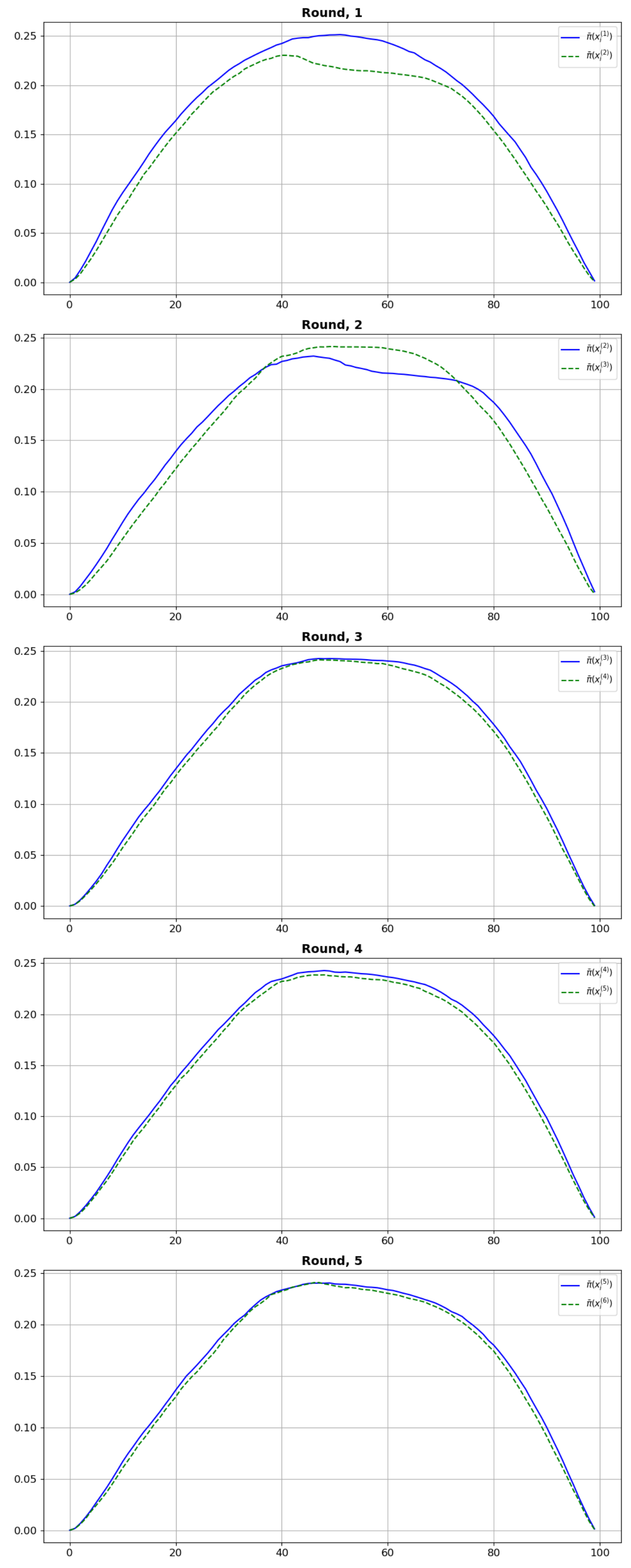
We start the discussion of the results for the cases when the borrowers follow the *all* perturbation scheme, i.e. figure 2a. When we look at the evolution of the two curves throughout the rounds of the game, we see convergence. Indeed, the lender's profit curve is only marginally modified by updating his decision function in round 5. In this context, this is not the case in the first round. Furthermore, we can see that in the last round displayed, and both profits curves are equivalent to the one in the first round with unperturbed data. Therefore, in this set-up, the repeated game allows the lender to fully undo the borrowers' perturbations.

Note that in the first figure—that is, the first round of the repeated game, we see that the lender could obtain a similar profit by recalibrating his logistic model's parameters θ or simply by re-estimating the decision threshold τ as defined in equation (3). This is in line with the intuition conveyed by the closed-form in section 5.2.

The second set of figures shown in figure 2b displays the same profit curves through the repeated game when the borrowers follow the *improve* perturbation scheme. We can see that the repeated game converges to a robust equilibrium—that is, neither the lender nor the borrowers wish to adapt their strategy. However, some information is irrevocably lost for the lender. This loss in information can be seen in two ways. First, by the fact that the original best mean profit was above 0.25 and the equilibrium one is below. Second, by the flatness of the curves close to the decision threshold. This flatness shows that some aggregation of information through the perturbation scheme has significantly reduced the lender's capacity to distinguish between borrowers whose probability of *bad* is close to τ.

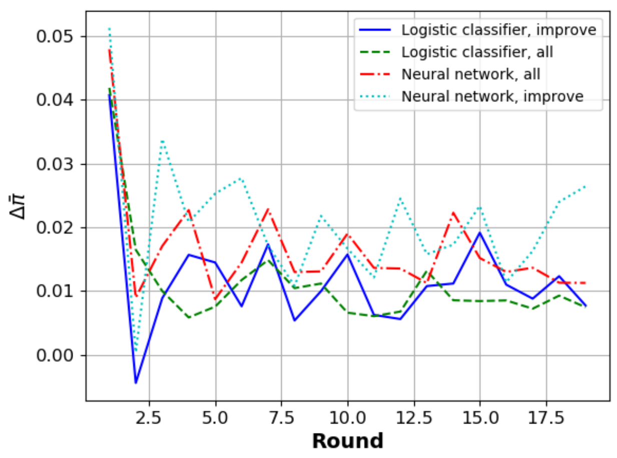


(a) *All* perturbation scheme

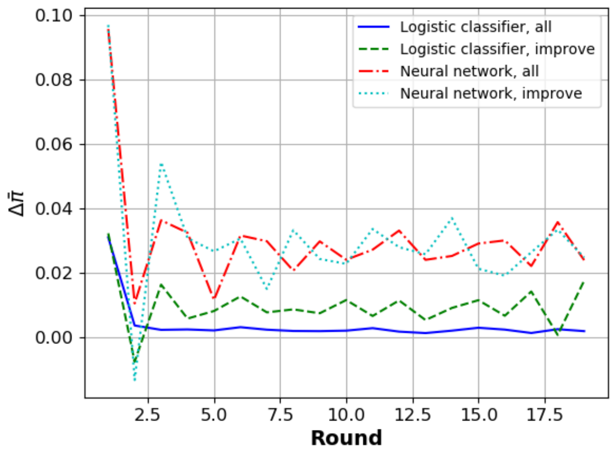


(b) *Improve* perturbation scheme

Figure 2: In the aboves' figures, we show one figure for each of the first five rounds of the repeated game. The figure at round t shows the profit curves for the lender with the borrowers' data perturbed t time and $t+1$ times, i.e $\bar{\pi}(x_i^{(t)})$ and $\bar{\pi}(x_i^{(t+1)})$. The x-axis shows the decision threshold in percentage points. The y-axis displays the average profit per borrower demanding a loan as defined in equation (11).



(a) Linear mapping of the borrowers' variables to their probability of *bad*.



(b) Non-linear mapping of the borrowers' variables to their probability of *bad*

Figure 3: In the two figures show the convergence of the repeated game measured as profit gained by the lender if he adapted his decision profits at the end of a round, i.e. $\Delta\bar{\pi}$. Each line shows this convergence for a different level of sophistication of the lender (logistic classifier or neural network) and for a different perturbation scheme (*all*, and *improve*).

Next, we study the convergence of the repeated game across multiple simulation's set-ups by computing the lender's gain when he updates his decision function's parameters θ and τ at the end of the round.

$$\Delta\bar{\pi} = \bar{\pi}(x_i^{(t+1)}|\theta^{(t+1)}, \tau^{(t+1)}) - \bar{\pi}(x_i^{(t+1)}|\theta^{(t)}, \tau^{(t)}) \tag{12}$$

Figures 3a and 3b show $\Delta\bar{\pi}$ for each round of the repeated game. Figure 3a shows the results in the simulations where the true mapping function $f_{true}(\cdot)$ is linear, whereas figure 3b shows the same results when the true mapping is non-linear. In both figures, the four lines show $\Delta\bar{\pi}$ when the borrowers follow the *all* or *improve* perturbation scheme, and when the lender uses a neural network or a linear logistic classifier,respectively.

In all combinations, we see that the lender's gain when he updates his decision function drops significantly after the first few rounds. $\Delta\bar{\pi}$ also converge to stable numbers—that is, a small variation in $\Delta\bar{\pi}$ from one round of the repeated game to the next. However, this gain does not converge towards zero in all cases.

Finally, we observe that the gain to update the decision function's parameters τ and θ is bigger when the lender uses a neural network to estimate $f_{true}(\cdot)$ than when he uses a linear logistic classifier. This effect appears to be more pronounced when $f_{true}(\cdot)$ is itself non-linear. This result suggests that the neural networks' sensitivity to adversarial attacks, as defined in section 2 allows strategic borrowers too better fool the lender's decision function when it relies on a neural network. We elaborate on this hypothesis in section 5.3.

5.2 The Predicting power of variables

In this section, we answer our second research question—that is, how does the adaptive behavior of the borrowers impact their variables' predictive power? And, under which conditions can one variable lose all its predicting power as a consequence of the borrowers' rational adaptation?

We start this analysis with some closed-form solutions for a simplified version of the repeated game. In particular, we investigate the simple case when the lender's model $f(x_i|\theta)$ is an ordinary least square regression. The lender does not assign to the borrower a probability of *bad*, but rather an inverse credit-scored defined in $\mathbb{R}$. The borrower consider a low credit score desirable and finds his optimal perturbation by solving equation 4.

In the case where x_i is a scalar, the function $f(x_i|\theta)$ becomes:

$$f(x_i|\theta) = \alpha + \beta x_i, \tag{13}$$

$$\alpha = \bar{y} - \beta \bar{x}, \tag{14}$$

$$\beta = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}, \tag{15}$$

where $\bar{\cdot}$ denotes the sample mean. Let $c_i = m_i - x_i$ be a measure of the perturbation size. We modify equation (4) to write the borrowers' problem as

$$c_i = \arg\min_{c_i} \left(\alpha + \beta(x_i + c_i) + \lambda_i c_i^2 \Lambda^{-1}\right). \tag{16}$$

Since the input is one-dimensional, both Λ and c_i are scalars. Solving the first-order condition for c_i yields,

$$c_i = \frac{1}{2\lambda_i} \beta \Lambda. \tag{17}$$

Net, we define α_2 and β_2 as the updated coefficient estimated on the perturbed sample $m_i = x_i + c_i$ $\forall$ i:

$$\beta_2 = \frac{\sum_{i=1}^{n} (x_i + x_i - (\bar{x} + \bar{c}))(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i + c_i - (\bar{x} + \bar{c}))^2}, \tag{18}$$

$$\alpha_2 = \bar{y} - \beta(\bar{x} - \bar{c}). \tag{19}$$

In the special case where $\lambda_i = \lambda$ $\forall$ i it is obvious that $\bar{c} = c_i$ and $\beta_2 = \beta$. To convey the intuition, we look now at some specific parameters' values. If $\beta > 0$, $\lambda > 0$, then it follows that $\alpha_2 > \alpha$. Furthermore $\frac{\partial}{\partial \Lambda}(\alpha_2 - \alpha) > 0$. Which means that robustness in this case is achived by adjusting the constant term α to counter the perturbation. This adjustment is bigger when we the cost of perturbation is low.

A few critical ideas can be extracted from this simple case. First, we see that the optimal perturbation depends on β. This means that there is a trivial solution if and only if β does not change when the lender changes his model. It follows that the value of β in equilibrium cannot be trivially deduced when the borrowers follow the *improve* perturbation scheme. Indeed, under this scheme, $c_i = 0$ if the perturbation does not lead to an improved classification.

Second, we have seen that in this simple set-up, robustness is achieved through an update of the constant term α. As every borrower perturb his variable systemically, the lender can update his decision algorithm by merely moving the decision threshold τ to correct for this perturbation. It follows that, in this specific case, the predictive power of the borrowers' variable is unaffected by the strategic perturbations.

In the appendix 7.1, we provide an extension of this closed-form analysis to two dimensions. We show that under the *all* perturbation scheme, even when the two variables of the borrowers have different cost of perturbations, robustness is still achieved by updating only the constant term α. In other words, in this simplified set-up, the strategic adaptation of the borrowers does not impact the predictive power of individual variables.

5.2.1 Linear logisitc classifier, homogoneous perturbation

Next, we look at the linear logistic classifier. With a one-dimensional input, the model of the lenders is defined as,

$$f(x_i) = \frac{1}{1+\exp\left[-(\alpha+\beta x_i)\right]}. \tag{20}$$

Compared to the previous subsection, this set-up is closer to the problem we wish to tackle as the lender's decision function is reduced to a probability of default instead of some continuous score.

To the extent of our knowledge, this model cannot be solved in closed form and therefore needs to be fitted through some optimization procedure. A common approach is to maximize the likelihood defined as

$$L_{\alpha,\beta} = \prod_{i=1}^{n} f\left(x_i\right)^{y_i}\left(1-f\left(x_i\right)\right)^{(1-y_i)}, \tag{21}$$

where y_i is a binary variable as in the simulations presented in section 3. As in the case of the ordinary least squares, we define $c_i = m_i - x_i$ as the amount of perturbation. We can then write the borrower's problem as

$$c_i = \arg\min_{c_i}\left(\frac{1}{1+\exp\left[-(\alpha+\beta(x_i+c_i))\right]} + \lambda_i c_i^2 \Lambda^{-1}\right). \tag{22}$$

The first order condition of equation (22) yields

$$\beta\frac{1}{2+\exp\left[-(\alpha+\beta(x_i+c_i))\right]+\exp\left[(\alpha+\beta(x_i+c_i))\right]} + 2\lambda_i c_i \Lambda^{-1} = 0. \tag{23}$$

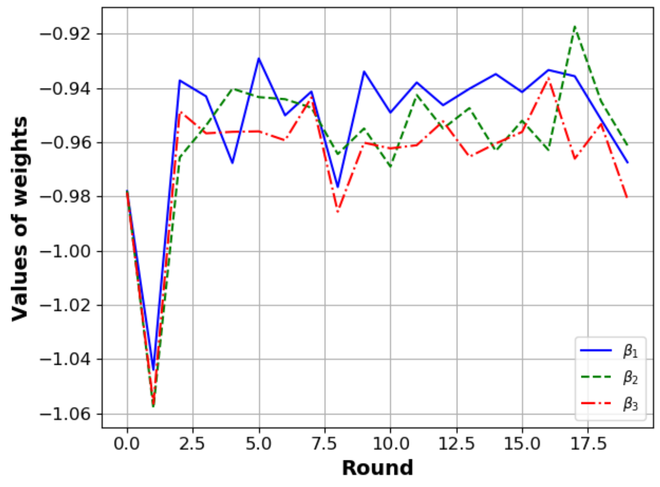
The robust parameters α_r and β_r are obtained through the maximization of the likelihood under the perturbed input m_i:

$$L_{\alpha_r,\beta_r} = \prod_{i=1}^{n} f\left(x_i+c_i\right)^{y_i}\left(1-f\left(x_i+c_i\right)\right)^{(1-y_i)}. \tag{24}$$

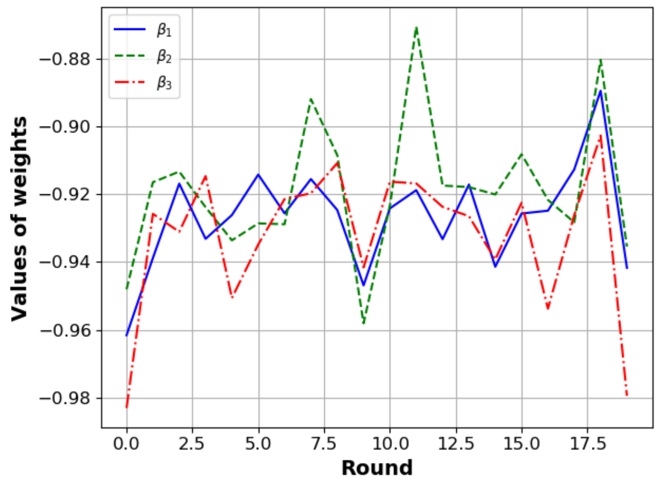
An equilibrium in the repeated game is obtained if, for a given set of α, β, c_i, both equations (23) and (24) are satisfied.

This example illustrates the necessity of numerical simulations to answer our research questions i), ii) and iii). Even in the simplest case, with only one dimension and a model in which the variables are still aggregated linearly, the non-linearity introduced by the logistic decision function is sufficient to make the problem non-trivial.

To analyze the effect of perturbations on individual variables0 predicting power, we restrict our analysis to simulations' setups with linear data and a non-sophisticated lender—that is, a lender who uses linear logistic classifier. This latter restriction allows us to easily display the evolution of all the models' parameters throughout subsequent game rounds. Indeed, with x_i being a vector of dimensions 3, the linear logistic classifier only has four parameters —that is, three β, and one constant α, whereas while a neural network with three layers of 64, 32, and 16 neurons respectively



(a) *All* perturbation scheme



(b) *Improve* perturbation

Figure 4: The figure above show the values of the linear logistic classifier's β-coefficients for each the round of the repeated game.

has $2,881$ parameters. Similarly, restricting our analysis to the simulations where the true mapping is non-linear allows us to isolate the effects of perturbations and perturbations costs.

We start by looking at the *sigma-homogenous* case, where the individual variables in the vector x_i are drawn from similar normal distributions as defined in equation (5). Figures 4a and 4b show the values of the three β-coefficients for each round of the game. Figure 4a shows these numbers when the borrowers follow the *all* perturbation scheme, while figure 4b shows the same coefficients' values when the borrowers follow the *improve* perturbation scheme.

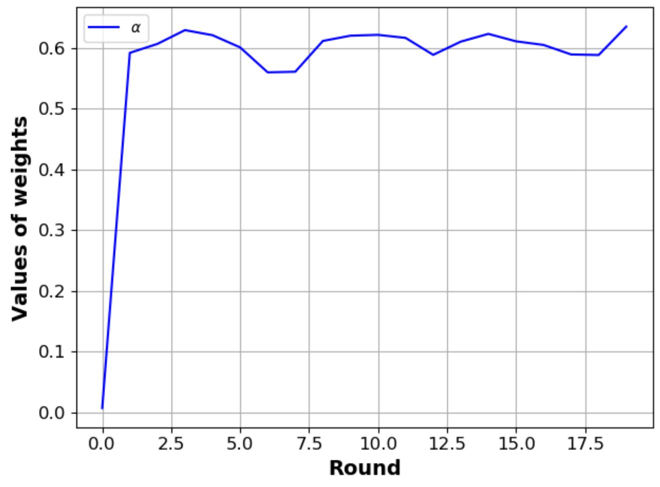
In both figures 4a and 4b, we see that the models' parameters slightly changes at the beginning of the game and then appear to stay at a relatively stable value once the game has converged. This convergence appears to be more stable under the *all* perturbation scheme than under the *improve* one. The β seem to lose a small amount of predictive power due to the borrowers' perturbations as the absolute values of the β-coefficient is smaller at the end of the repeated game than at its start. Figures 5a and 5b show the value of the constant term α throughout the rounds of the games. Under both the *all* and the *improve* perturbation scheme, the constant term α is moved from its original value of zero to a significantly higher value as the game converges to an equilibrium. This result is in line with the intuition conveyed by our closed-form analysis when the lender uses an ordinary least squares model.

5.2.2 Linear logistic classifier, heterogeneous perturbation

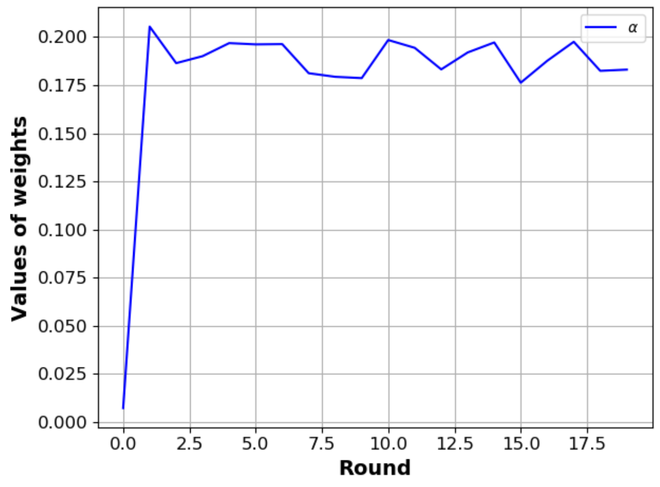
We continue our analysis by looking at simulations under the σ*-heterogeneous* mapping. Under this mapping, the underlying distribution of the variables in x_i is defined in equation (6). The first dimension of the vectors x_i is drawn from a normal$\sim \mathcal{N}(0,10)$ whereas the other two dimensions are drawn from normals$\sim \mathcal{N}(0,1)$.

Equation (4) dictates that the perturbation cost of a single variable is proportional to the inverse covariance matrix of the original inputs. Therefore, in this simulation, the perturbation cost of the first variable is significantly lower than the other two.

Figures 6a and 6b show the mean absolute perturbation defined as $|x_i^{(d)} - m_i^{(d)}|$ for each round of



(a) *All* perturbation scheme



(b) *Improve* perturbation

Figure 5: The figures above show the values of the linear logistic classifier's coefficient α through the rounds of the repeated game.

the repeated game.[21] The significantly higher mean absolute perturbation of the first variable shows that the lower cost of perturbation did lead to stronger perturbations by the borrowers. This effect appears both under the *all* and *improve* perturbation scheme.

Figures 7a and 7b show the corresponding β for each round of the repeated game. Similarly, figures 8a and 8b show the corresponding α. Somewhat surprisingly, we observe that the importance of the first variable does not decrease more than the other two variables, even though the perturbations are significantly higher for this variable. The reason we observe such behavior is the following: Remember that the cost of perturbing a variable is modified here by changing the variable standard deviation but keeping the true function mapping the vector x_i unchanged. In a classification problem, all things being equal, the variable with the highest standard deviation has the strongest predictive power.[22]

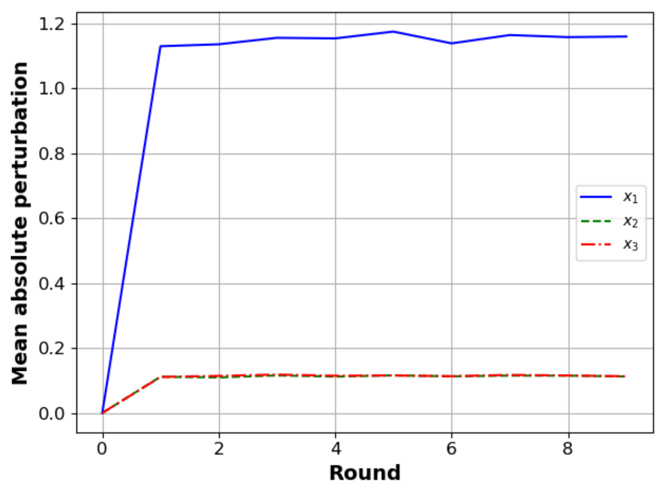
In this simulation's setup, the increased standard deviation increases the average absolute perturbation of the first variable's and its predictive power. These two effects cancel each other out, and the variable's predictive power is unaffected.

To confirm this theory, we run another set of simulations where the standard deviation is kept constant for all three variables, but the cost of perturbation is modified to correspond to the previous simulation. In other words, all three variables are drawn from $\mathcal{N}(0,1)$ but the covariance matrix used in equation (4) to find optimal perturbations corresponds to the one displayed in equation (6). This allows us to isolate the effect of cheaper perturbation costs from the effect of high a standard deviation.

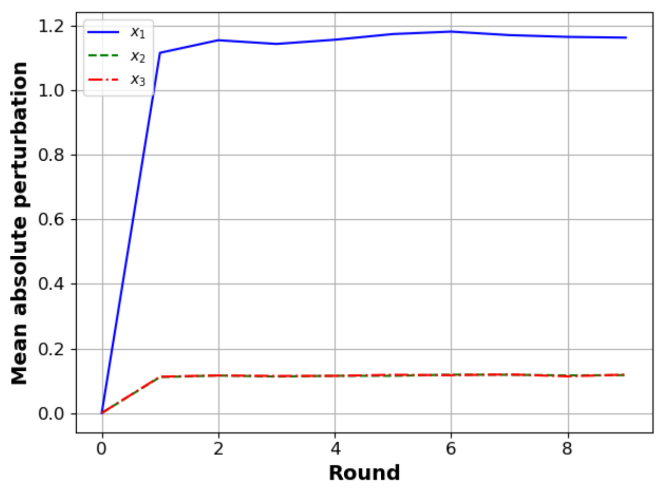
In figures 9a and 9b we show, as in figures 6a and 6b, the mean absolute perturbation per variable

[21]Note that we analyze each dimension of the vector x_i. The upper script d in $x_i^{(d)}$ indicates the variables dimension and not the round of the repeated game.

[22] We illustrate this idea through a trivial numerical example with a logistic classifier with a single input as defined in equation (20). Let's assume $\alpha = 0$ and $\beta = 1$. We now assume two discrete distributions for the scalar input x_i–that is low-σ and high-σ. Under the low-σ distribution, x_i can be equal to 1, or -1 with equal probabilities. Under the high-σ distribution, x_i can be equal to 2, or -2 with equal probability. It is obvious to see that both these distributions have similar mean but high-σ as a higher standard deviation. Under the low-σ distribution, the logit function $f(x_i)$ can yield two values, 0.73 or 0.27. Under the high-σ distribution, the corresponding predicted probabilities become 0.88 and 0.12. This simple case illustrates how in classification problems when keeping everything else equal, increasing the standard deviation of a variable increases its predictive power.

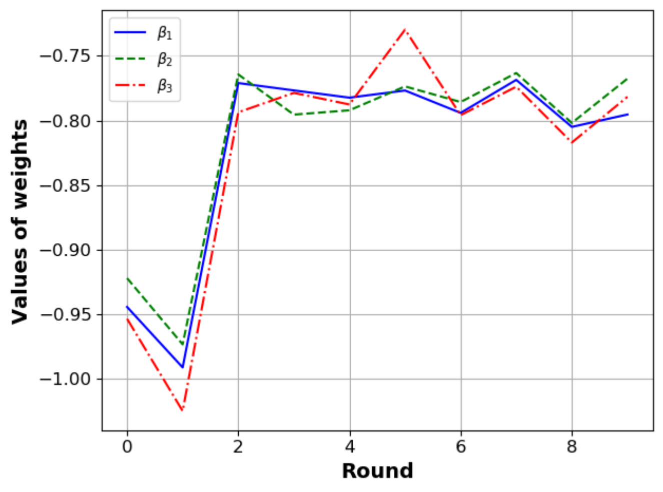


(a) *All* perturbation scheme

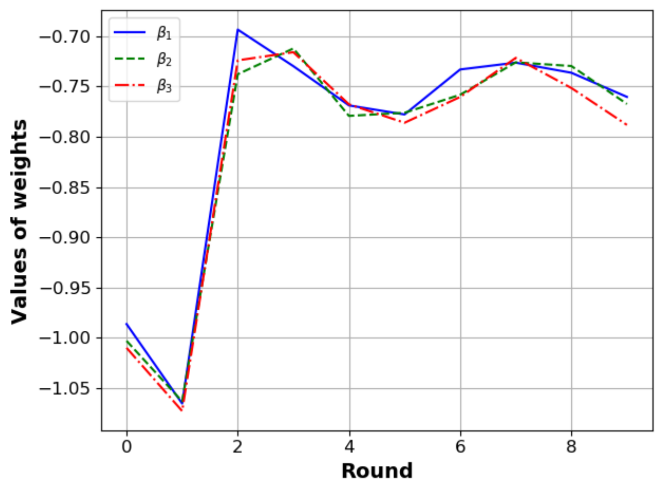


(b) *Improve* perturbation

Figure 6: These figures show the mean perturbation of each input dimension d measured as $|x_i^{(d)} - m_i^{(d)}|$. The distribution of the variables in x_i is heterogenous. The first variable x_0 has a standard deviation of 10, while the others have a standard deviation of 1.



(a) *All* perturbation scheme



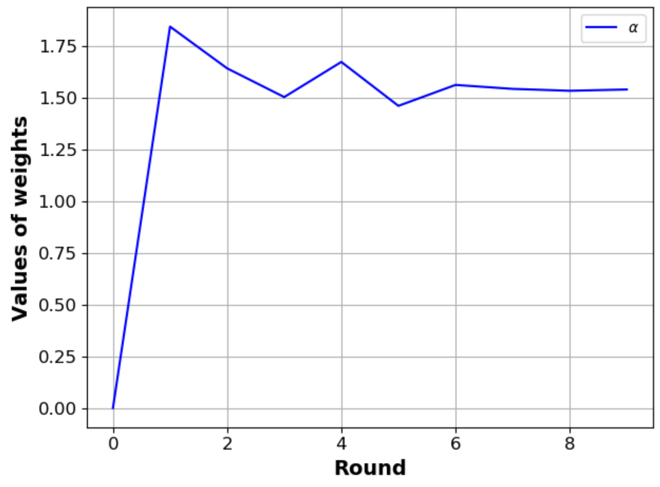
(b) *Improve* perturbation

Figure 7: These figures show the values of the linear logistic classifier's coefficients β for each round of the repeated game. The variable are drawn from the same distributions as in figures 6a and 6b.

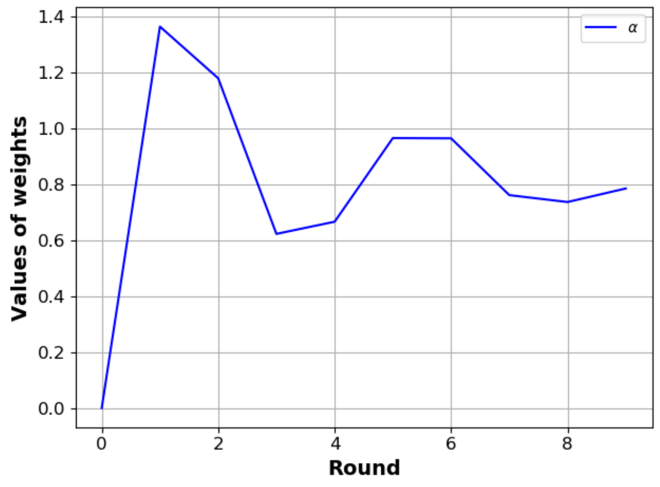
. As was the cases in these first figures, we can see that the mean perturbation is significantly higher for the first variables under both the *all* and *improve* perturbation scheme.

In figures 10a and 11b, we display the logistic classifier's parameters β for each round of the repeated game. Unlike the results displayed in figures 7a and 8b, when the game converged, the absolute value of β_1 is significantly lower than the ones of β_2 and β_3. This result is significantly stronger under the *improve* than under the *all* perturbation scheme.

These simulations delivered the following observation: (a) The importance of a predicting power can be diminished by a relatively lower cost of perturbation. (b) If this relatively lower cost is obtained by a higher standard deviation, then the effect is canceled out by a relatively stronger predictive power. This stronger predictive power is obtained because, in a classification problem, everything being equal, a variable with a higher standard deviation is a better predictor than one



(a) *All* perturbation scheme



(b) *Improve* perturbation

Figure 8: These figures show the values of the linear logistic classifier's coefficient α for each round of the repeated game. The variable are drawn from the same distributions as in figures 6a and 6b.

with a lower standard deviation. (c) While a significant lower cost of perturbation does diminish the predictive power of a variable, it does not drag it down to zero. Under the *all* perturbation scheme, in the last simulation, the cost of perturbing the first variable by one unit was 0.0001, while the cost of perturbing the other variables was 1.[23] This huge difference in perturbation costs translates into a relatively small difference in optimal β. β_1 converges to -0.7, whereas β_2 and β_3 converge to -0.85, and β_1/β_2 yields 0.88. The same ratio obtained by dividing the low perturbation cost by the high perturbation cost yields 0.0001. This significant difference suggests that only a perturbation cost close to zero, or a relatively low original predictive power would yield the perturbation process to completely remove the variable predictive power.

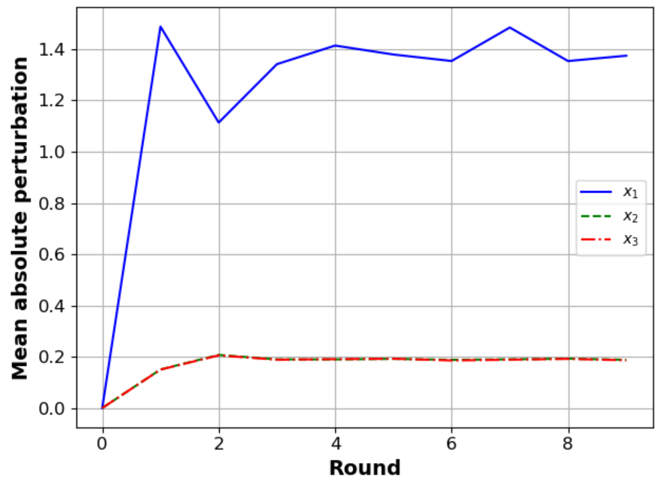
This last point (c) is especially relevant given the recent interest in new types of data to construct a credit score. For example, Óskarsdóttir et al. (2019), for instance, showed that the phone call history of borrowers could be used to help predict default probabilities, while Netzer et al. (2019) showed how natural language processing could allow a lender to use the texts written by the borrowers in loan screening applications. While it is clear that phone call habits can be strategically changed by potential borrowers, it is also clear that any such change would come with a high utility cost for the borrower. Our analysis suggests that while some robustness method should be applied, the phone call history of the borrowers is unlikely to lose all predictive power once the borrower adapts to the decision rules' introduction. On the other hand, it could be argued that a borrower can strategically change his essay written to obtain a loan at a cost close to zero. Therefore, such a data source may lose all its predictive power once the borrowers adapted to the new decision function of the lender.

5.3 Under- and overspecification

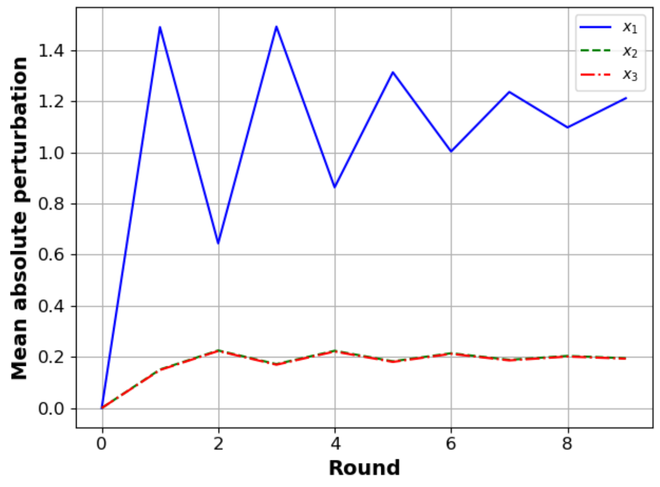
In this subsection, we answer our third research question—that is can underspecified models, such as a linear logistic classifier, outperform overspecified models like a deep neural network?

We define overspecified models as the ones which can capture functional forms more complex than the one they are trying to estimate. Symmetrically, we define a model as underspecified if it

[23]The cost is directly proportional to the inverse covariance matrix. Since the variable are uncorrelated, this translates into $1/\sigma^2$ per unit of perturbation.

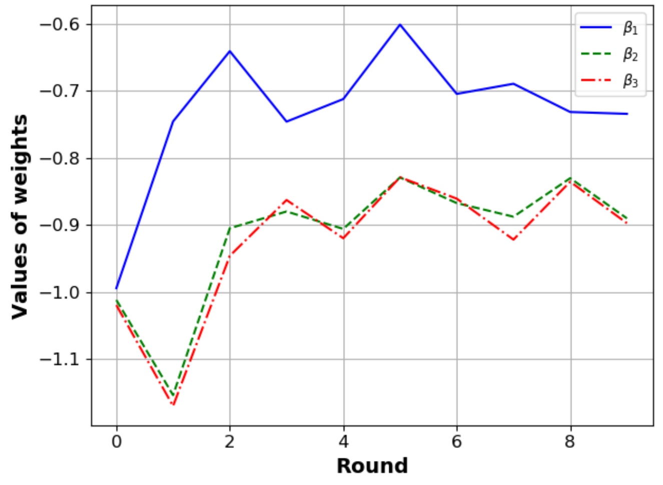


(a) *All* perturbation scheme

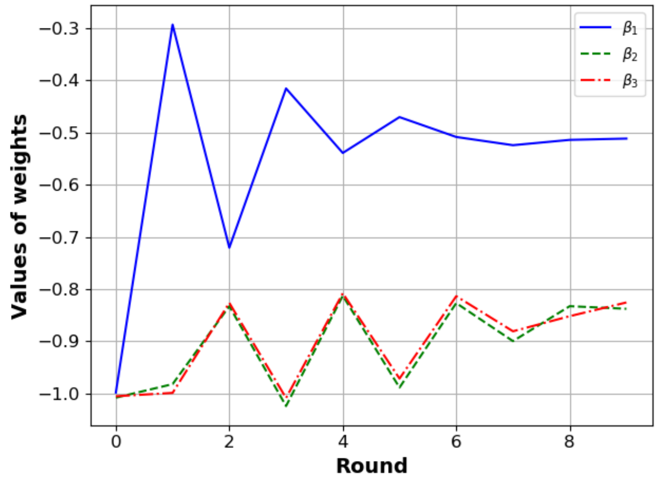


(b) *Improve* perturbation

Figure 9: These figures displays the mean perturbation input per input measured as $|x_i^{(d)} - m_i^{(d)}|$. The distribution of the variables in x_i is heterogenous. All three variables have similar standard deviation but the covariance matrix used in equation (4) corresponds to the one displayed in equation (6).



(a) *All* perturbation scheme



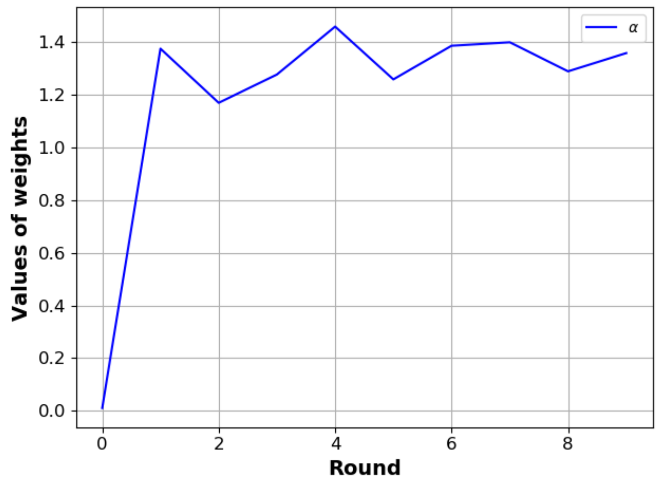
(b) *Improve* perturbation

Figure 10: These figures shows the values of the logistic classifier's β coefficients accross the rounds of the repeated game. The simulations parameters are the same as those used to produce figures 9a and 9b.

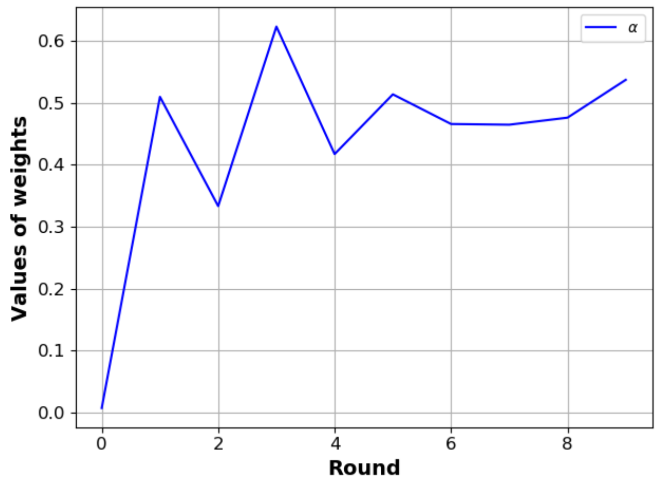
can only capture a fraction of some true function. For example, if we define some target function $f_{true}(x, y|\beta_1, \beta_2) = \beta_1 x + \beta_2 y^2$. Then, model $f_a(x, y|\beta_1, \beta_2) = \beta_1 x + \beta_2 y$ is underspecified with regard to the true function $f_{true}(\cdot)$. Symmetrically, another model defined as $f_b(x, y|\beta_1, \beta_2, \beta_3, \beta_4) = \beta_1 x + \beta_2 y + \beta_1 x^2 + \beta_2 x^3$ is overspecified.

Neural networks are known to be universal approximators—that is, they can, given enough data, they can approximate any function (see, e.g., Hornik et al., 1989; Hornik, 1991). Therefore, a neural network can be said to be overspecified for any functional form $f_{true}(\cdot)$.

The linear logistic classifier used by what we defined in section 3 as an *unsophisticated* lender can be underspecified. In our previous analysis, we focussed on the cases where the true function mapping the borrowers' variables to their respective probability of *bad* was linear. In such a set-up, the linear-logistic classifier is correctly specified—that is, neither over- nor underspecified. With equation (10),



(a) *All* perturbation scheme



(b) *Improve* perturbation

Figure 11: These figures shows the values of the logistic classifier's α coefficients accross the rounds of the repeated game. The simulations parameters are the same as those used to produce figures 9a and 9b.

we also define a non-linear mapping for which the linear logistic classifier is underspecified.

As a benchmark, we start by investigating the static scenario when the borrowers do not adapt to the lender's decision function. In table 3, we report the average profit of the lender when the borrowers do not perturb their inputs. We see that when $f_{true}(\cdot)$ is linear, the correctly specified model slightly outperforms the overspecified neural network.[24] When the true mapping is non-linear, we see that the neural network significantly outperforms the underspecified linear logistic classifier with an average excess profit of 0.34.

None of these two results are surprising: the overspecified model slightly underperformed the correctly specified one because of numerical noise, while the underspecified model is significantly outperformed by the overspecified one. In other words, when the input distribution is static, it is only slightly better to use the correct model than using a neural network. However, it is significantly better to use a neural network than an underspecified model as long as the model's introduction does not impact the inputs' distribution.

	Logistic Classifier	Neural Network
Linear	0.255	0.253
Non-Linear	0.201	0.235

Table 3: This table shows the average profit of the lender as defined in equation (11), when the borrowers do not adapt to the lender's decision function.

Next, we analyze over- and underspecification when the borrowers do adapt to the algorithm through the repeated game, which is summarized in algorithm (2).

In figures 12a and 12b, we show the average profit of both a sophisticated and an unsophisticated lender estimated at the end of each round. Hence, the profit in these figures has been computed

[24] The small difference in average profit (0.02) when the mapping is linear can be explained as numerical noise. The lender using a linear logistic classifier has to correctly estimate the four parameters of his model. The same lender using a neural network with three layers containing 64, 32, and 16 neurons has to estimate 2,881 parameters. This massive difference in the complexity of the model's calibration significantly increases the likelihood of small numerical errors.

when the borrowers were the last to update their strategies. In both figures, the function $f_{true}(\cdot)$ which maps the borrowers' variables to their respective probability of *bad* is linear. Therefore, the neural network is overspecified while the logistic classifier is correctly specified.

We see that both under the *all* and under the *improve* perturbation scheme, the correctly specified model produces a slightly higher average profit than the overspecified models. We also note that in all cases, the lender's profits at the start of the repeated game is significantly lower than his profit at the end of the game. This robustness of the lender's algorithm is achieved by *playing* the first few rounds of the repeated game.

In figures 13a and 13b, we display the same profit lines as in figures 12a and 12b, but the function $f_{true}(\cdot)$ is now non-linear, as defined in equation 10. In these two figures, the linear logistic model is now underspecified. Under the *all* perturbation scheme, we see that the neural network outperforms the linear logistic classifier in most of the game's rounds. However, this outperformance is noisy and smaller than in the static case presented in table 3. Under the *improve* perturbation scheme, the neural network and the linear logistic classifier create similar profits for the lender. These last results show that under the *improve* perturbation scheme, the advantage of the neural network over the underspecified model is entirely undone by the borrowers' perturbations.

Finally, we see that with a non-linear mapping $f_{true}(\cdot)$, and under both perturbation schemes, the profit in the first round of the game obtained with the neural network is significantly lower than the one obtained with the linear logistic classifier. The said profits represent the ones from a naive lender who does not anticipate any of the borrowers' perturbations.

This last result shows that an unanticipated endogenous perturbation following the introduction of a new decision function is likely to cost significantly more to a *sophisticated* lender using a neural network than an *unsophisticated* one using a linear logistic classifier.

Taken together, the results presented in this subsection show that vanilla neural networks[25] are not well suited for applications when strategic endogenous perturbations of the network's inputs are likely to occur. We have shown that, even when the target function that is non-linear, the perturbation process reduces the model's performance to that of a simple linear logistic classifier.

We suggest that this weakness of neural networks can be traced back to these models sensitivity to adversarial attacks (see, section 2). We present now evidence to confirm this hypothesis by computing the average change in the perturbation of the borrowers for each round t—that is ,

$$\Delta Pert_t = \frac{1}{N}\frac{1}{3}\sum_{d=1}^{3}\sum_{i=1}^{N}\left|x_i^{((d),(t))} - x_i^{((d),(t-1))}\right|. \tag{25}$$

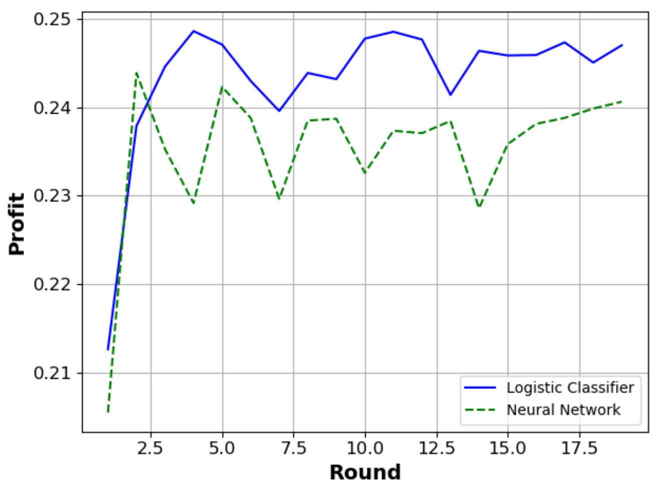
This measure estimates, on average, how much an individual borrower adapts his perturbation strategy from one round to the next.

Figures 14a and 14b shows this measure under both perturbation schemes when the true mapping $f_{true}(\cdot)$ is linear. In contrast, figures 15a and 15b show the same measure when the true mapping is non-linear. In all four cases, the mean change in perturbation is higher between the first and second rounds of the game, when the borrowers' input moved from no perturbation to some perturbation. More importantly, we see that the average change in perturbation $\Delta Pert_t$ is significantly higher when the lender uses a neural network than when he uses a linear logistic classifier.

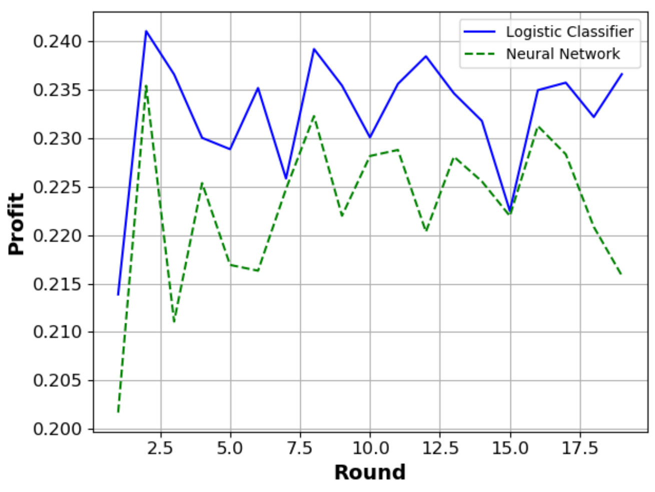
These findings show that, when the lender uses a neural network, the borrowers always adapt their perturbation strategy to the latest calibration of the neural network. They do so even when the profit of the lender is stabilized throughout the game round. Hence, the borrowers do adapt to take

[25]Vanilla neural networks here are defined as non-robust against adversarial attacks, such as a feed-forward network (Goodfellow et al., 2014)

advantage of the networks' sensitivity to specific attacks instead of adapting because the functional form estimated by the network has changed.

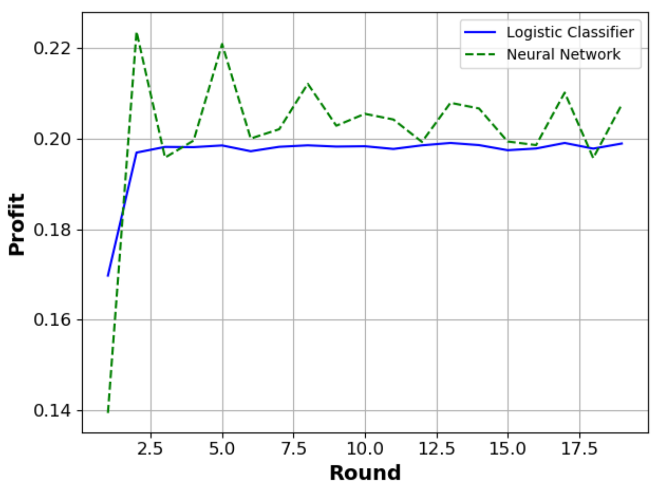


(a) *All* perturbation scheme

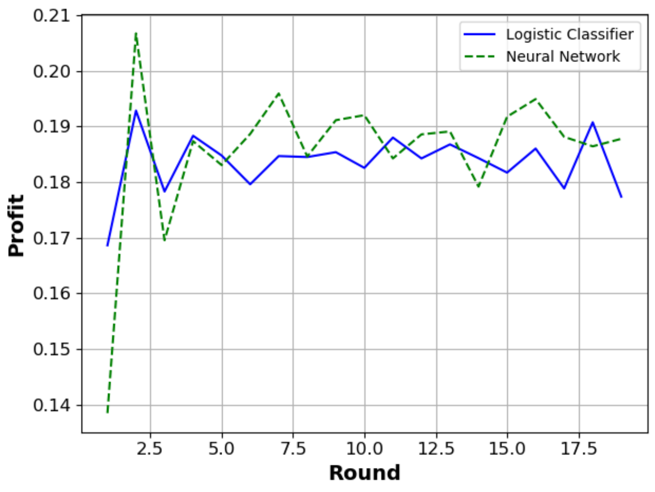


(b) *Improve* perturbation

Figure 12: In these figures, we show the profits of sophisticated (neural network) and unsophisticated (logistic classifier) lenders throughout the repeated game's rounds. The lenders' profit is computed at the end of each round as defined in algorithm 1, hence the borrowers where the last agents to adapt their strategies. The true function mapping the borrowers' variable to their respective probability of *bad* was **linear**. It follows that the neural network is overspecified while the logistic classifier is correctly specified.



(a) *All* perturbation scheme

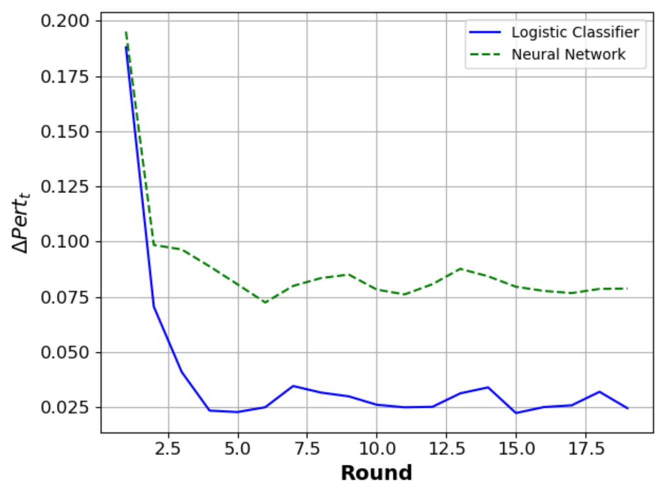


(b) *Improve* perturbation

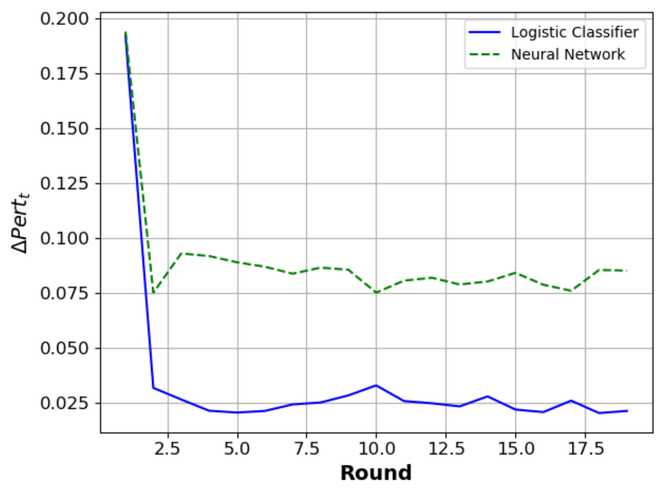
Figure 13: In these figures we show the profits of sophisticated (neural network) and unsophisiticated (logistic classifier) lender throughout the repeated game's rounds. The lender's profit is computed at the end of each round as defined in algorithm 1, hence the borrowers where the last agents to adapt their strategies. The true function mapping the borrowers' variable to their respective probability of *bad* was **non-linear**. It follows that the neural network is overspecified while the logistic classifier is under specified.

5.4 Lending club analysis

TODO, write the analysis here

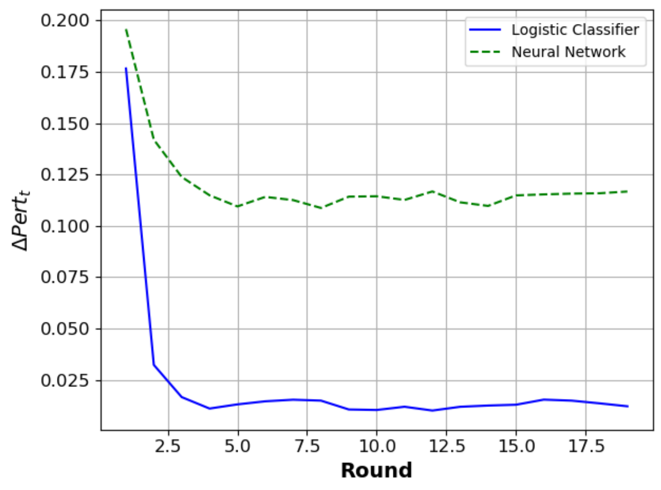


(a) *All* perturbation scheme

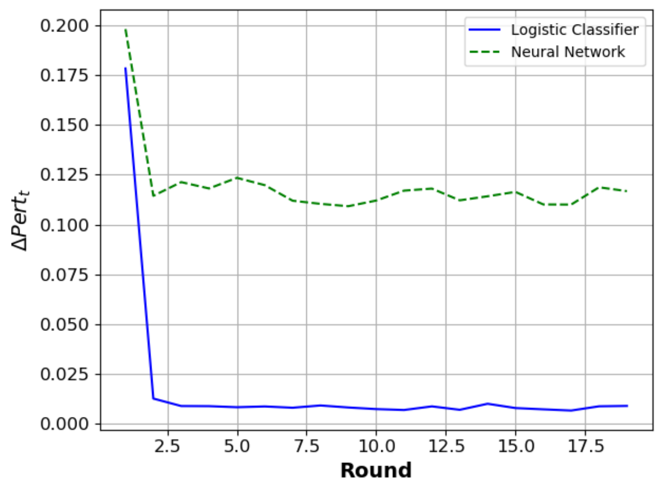


(b) *Improve* perturbation

Figure 14: In these figures we show the average change in the borrowers' perturbation strategy as defined in equation 25. The true mapping function f_{true} in both figures is *linear*.



(a) *All* perturbation scheme



(b) *Improve* perturbation

Figure 15: In these figures we show the average change in the borrowers' perturbation strategy as defined in equation 25. The true mapping function f_{true} in both figures is *non-linear*.

6 Conclusion

7 Appendix

7.1 Ordinary least square, two dimension

Next, we extend our analysis to two dimensions. Let x_1 and x_2 be two independent variables with mean $\bar{x_1}$ and $\bar{x_2}$. The lenders score is given by the following formula:

$$f(x_1, x_2|\theta) = \alpha + \beta_1 x_1 + \beta_2 x_2, \tag{26}$$

where,

$$\beta_1 = \frac{\left(\sum(x_2-\bar{x_2})^2\right)\left(\sum(x_1-\bar{x_1})(y-\bar{y})\right) - \left(\sum(x_1-\bar{x_1})(x_2-\bar{x_2})\right)\left(\sum(x_2-\bar{x_2})(y-\bar{y})\right)}{\left(\sum(x_1-\bar{x_1})^2\right)\left(\sum(x_2-\bar{x_2})^2\right) - \left(\sum(x_1-\bar{x_1})(x_2-\bar{x_2})\right)^2}, \tag{27}$$

$$\beta_2 = \frac{\left(\sum(x_1-\bar{x_1})^2\right)\left(\sum(x_2-\bar{x_2})(y-\bar{y})\right) - \left(\sum(x_1-\bar{x_1})(x_2-\bar{x_2})\right)\left(\sum(x_1-\bar{x_1})(y-\bar{y})\right)}{\left(\sum(x_1-\bar{x_1})^2\right)\left(\sum(x_2-\bar{x_2})^2\right) - \left(\sum(x_1-\bar{x_1})(x_2-\bar{x_2})\right)^2}, \tag{28}$$

and

$$\alpha = \bar{y} - \beta_1\bar{x_1} - \beta_2\bar{x_2}. \tag{29}$$

Let $c_1 = m_1 - x_1$ and $c_2 = m_2 - x_2$ be the measure of cheat size of a given borrower. Then, extending equation 4, we can write the borrowers' optimisation problem as:

$$c_1, c_2 = \arg\min_{c_1,c_2}\left(\alpha + \beta_1(x_1+c_1) + \beta_2(x_2+c_2) + \lambda_i c_1^2\Lambda_1^{-1} + \lambda_i c_2^2\Lambda_2^{-1}\right) \tag{30}$$

Solving the first order condition we get the optimal cheat level in the two variables as:

$$c_1 = \frac{1}{2\lambda_i}\beta_1\Lambda_1 \tag{31}$$

$$c_2 = \frac{1}{2\lambda_i}\beta_2\Lambda_2 \tag{32}$$

c_1 and c_2 is function of their respective cheat cost Λ and regression coefficient β. In the special case where $\lambda_i = \lambda\forall i$ we can generalize the one dimensional result that only α will adapt to the perturbed input. Indeed, in this simplfiy case, c_1 and c_2 are equal for all agents which means that $x_1 - \bar{x_1} = x_1 + c_1 - (\bar{x_1} + \bar{c_1})$ as $c_1 = \bar{c_1}$. Hence equation (27) and (28) are not affected by the perturbation and α becomes:

$$\alpha_2 = \bar{y} - \beta_1(\bar{x_1} + c_1) - \beta_2(\bar{x_2} + c_2). \tag{33}$$

7.2 Gan Formulation of the lender's cost

Adapting the robustness approach developed in Hosseini et al. (2017), we can model the cost of cheating in a semi-non parametric manner. We will model the probability of getting caught for a given $m_i - x_i$ directly through a neural network instead of an inverted matrix. The method is only semi-parametric because we still need to define the cost of cheating relative to the benefits of getting a loan, but it will be a single easy to interpret parameters.

7.2.1 Borrower

In this small subsection, we propose a small economic justification of the one-parameter model presented in the next section.

Let $c(x_i - m_i)$ be the probability of catching agent i falsifying their true input x_i into submitted input m_i. We will model the borrower's dilemma as follows. At any time, being accepted by the lender for a loan increases his utility by some fixed value γ. The borrower is assumed risk-neutral and discounts his future utility with some factor θ. Assuming he does not falsify his input, his expected utility is defined as:

$$u_{x^i} = \sum_{t=1}^{\infty} \gamma \theta^t \phi(x_i) \tag{34}$$

Getting caught cheating will not only cost him γ for not receiving a loan but also any chance of getting a loan in the future. Hence, his utility when cheating is defined as:

$$u_{m^i} = \mathbb{E}\left[\sum_{t=1}^{\tau} \gamma \theta^t \phi(x_i)\right] \tag{35}$$

Where τ is the random stopping time when the borrower is caught from cheating and denied the opportunity to apply for a loan in the future. Assuming $c(x_i - m_i)$ is i.i.d across time for a given $x_i - m_i$, τ follows a geometric distribution with $\mathbb{E}(\tau) = \frac{1-c(x_i-m_i)}{c(x_i-m_i)}$ and $P(\tau = t) = (1 - c(x_i - m_i))^t c(x_i - m_i)$. Which means we can write u_{m^i} as:

$$u_{m^i} = \left[\sum_{t=1}^{\infty} (1 - c(x_i - m_i))^t c(x_i - m_i) \sum_{\bar{t}=1}^{t} \gamma \theta^{\bar{t}} \phi(x_i)\right] \tag{36}$$

For a given algorithm $\phi(\cdot)$ and probability of getting caught $c(\cdot)$ a borrower cheats if $u_{x_i} < u\hat{m}_i$ where $\hat{m}_i$ maximize equation 36

7.2.2 Single parameter version

In this section, I propose a single parameter way of modeling the decision of a borrower deciding to cheat or not. This process is applied to one who's unperturbed input is refused by the lender's algorithm, $\phi(x_i) = 0$. This simpler formulation of the problem is perhaps a less direct link to the traditional economic model but can be defined by a single easy to interpret the parameter.

The cheating borrower has three potential states of nature in which he may end up. Ideally, he gets the loan but doesn't get caught where he gains some utility γ. In the worst-case scenario, he gets the loan but gets caught and gets a utility cost ζ. Alternatively, he can simply not get the loan. This last state of nature can be standardized to have no impact on his utility as he will simply not cheat if it does not allow him to get a loan. As such, we can write his expected utility as:

$$\mathbb{E}(u) = \gamma f(m_i|\hat{\theta}) + \zeta c(x_i - m_i) \tag{37}$$

It follows that the borrowers cheat if:

$$\gamma f(m_i|\hat{\theta}) + \zeta c(x_i - m_i) > 0 \tag{38}$$

$$-\frac{\gamma}{\zeta} \frac{f(m_i|\hat{\theta})}{c(x_i - m_i)} > 1 \tag{39}$$

We can finally define $\kappa = -\frac{\gamma}{\zeta}$ as the strictly positive ratio of utility.[26] Between receiving a loan and getting caught. A κ below 1 captures a world in which borrowers' gain in getting a loan is comparatively weaker than the costs of getting caught. While a κ bigger than 1 captures a world in which the cost of getting caught is comparatively weaker than the benefits of getting a loan.

7.2.3 Lender with cheat detection

In this game, the lender has to define two algorithms. The first, $\phi(x_i)$, transforms the borrower's input into a binary variable giving or not a loan to the lender. This is similar to what is defined in section 3 by equation 3 assuming no cheat. Only the original unperturbed data are used at this stage.

The second algorithm, $\xi(\cdot)$, is used to catch a cheater. This is performed at time $t+1$ after having assigned loans according to algorithm $\phi(m_i)$ at time t.

$\xi(m_i, f(m_i|\hat{\theta}), y_i|\Gamma)$ is function of a set of parameters Γ and takes as input the perturbed covariates m_i, the probability of default $f(m_i|\hat{\theta})$ as estimated in the algorithm $\phi(\cdot)$ and the binary variable equal to 1 if the borrowers defaulted. The latter are available because detecting cheater is performed at time $t+1$ where default have been observed.

7.2.4 Repeated game

The repeated game is a very simple extension of what was proposed in previous sections. At every step of the game, the lenders update both his decision algorithm and his cheat detection algorithm instead of simply one. Every other step is similar.

The algorithm requires a cheat algorithm to perturb, and the cheat algorithm requires to perturb data to train. This means that we have a problem to initialize the cheat algorithm, this can, however, easily be worked around by either using a very simplistic $\xi(m_i, f(m_i|\hat{\theta}), y_i|\Gamma) = \alpha$ for some fix α in the first round or follows the perturbation rules defined in the previous section for the first round.

[26] ζ is strictly negative

Algorithm 2 summarize the step of the convergence game with quasi-non-parametric cheat cost.

```
**Data**: $N$ historical pairs $[x_i; y_i]$
**Result**: $\hat{\theta}$, $\hat{\tau}$, $\hat{\Gamma}$
initialization;
$\hat{\theta}_0 = \arg\min_{\theta} - \sum_{i=1}^{N} y_i \log(f(x_i|\theta)) + (1-y_i)\log(1-f(x_i|\theta))$ ;
$\hat{\tau}_0 = \arg\min_{\tau} \sum_{i=1}^{N} \mathbb{1}_{f(x_i|\hat{\theta_0})<\hat{\tau}} G - \mathbb{1}_{f(x_i|\hat{\theta_0})<\hat{\tau}} L$ ;
$m_i^{(0)} = x_i$;
$m_i^{(1)} = x_i + 2\epsilon$;
$t = 1$;
$\xi(m_i, f(m_i|\hat{\theta}), y_i|\hat{\Gamma}) = \alpha$;
**while** $\frac{1}{N}\sum_{i=0}^{N} ||m_i^{(t-1)} - m_i^{(t)}|| < \epsilon$ **do**
    $m_i^{(t)} = Pert(x_i, \hat{\theta}_t, \hat{\tau}_t, \xi(\cdot, \hat{\Gamma}_t))$;
    $\hat{\theta}_t = \arg\min_{\theta} - \sum_{i=1}^{N} y_i \log(f(m_i^{(t)}|\theta)) + (1-y_i)\log(1-f((m_i^{(t)}|\theta))$ ;
    $\hat{\tau}_t = \arg\min_{\tau} \sum_{i=1}^{N} \mathbb{1}_{f(x_i|\hat{\theta}_t)<\tau} G - \mathbb{1}_{f(x_i|\hat{\theta}_t)<\tau} L$ ;
    $\hat{\Gamma}_t = \arg\min_{\Gamma} - \sum_{i=1}^{N} \mathbb{1}_{x_i \neq m_i} \log(\xi(m_i, f(m_i|\hat{\theta}), y_i|\Gamma)) + \mathbb{1}_{x_i = m_i} \log(1-\xi(m_i, f(m_i|\hat{\theta}), y_i|\Gamma))$ ;
    $t+1$;
**end**
$\hat{\theta} = \hat{\theta}_t$;
$\hat{\tau} = \hat{\tau}_t$;
$\hat{\Gamma} = \hat{\Gamma}_t$;
```

Algorithm 2: Repeated Game With Cheat detection

References

Bastani, O., Ioannou, Y., Lampropoulos, L., Vytiniotis, D., Nori, A., and Criminisi, A. (2016). Measuring neural net robustness with constraints. In *Advances in neural information processing systems*, pages 2613–2621.

Bravo, C., Thomas, L. C., and Weber, R. (2015). Improving credit scoring by differentiating defaulter behaviour. *Journal of the operational research society*, 66(5):771–781.

Broecker, T. (1990). Credit-worthiness tests and interbank competition. *Econometrica: Journal of the Econometric Society*, pages 429–452.

Carleton, W. T. and Lerner, E. M. (1969). Statistical credit scoring of municipal bonds. *Journal of Money, Credit and Banking*, 1(4):750–764.

Cisse, M., Adi, Y., Neverova, N., and Keshet, J. (2017). Houdini: Fooling deep structured prediction models. *arXiv preprint arXiv:1707.05373*.

De Meza, D. and Webb, D. C. (1987). Too much investment: a problem of asymmetric information. *The quarterly journal of economics*, 102(2):281–292.

Durand, D. (1941). *Risk elements in consumer installment financing.* National Bureau of Economic Research, New York.

Dziugaite, G. K., Ghahramani, Z., and Roy, D. M. (2016). A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853*.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188.

Fredrikson, M., Jha, S., and Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 1322–1333.

Freixas, X., Hurkens, S., Morrison, A. D., and Vulkan, N. (2007). Interbank competition with costly screening. *The BE Journal of Theoretical Economics*, 7(1).

Gao, J., Wang, B., Lin, Z., Xu, W., and Qi, Y. (2017). Deepcloak: Masking deep neural network models for robustness against adversarial samples. *arXiv preprint arXiv:1702.06763*.

Gao, L., Zhou, C., Gao, H.-B., and Shi, Y.-R. (2006). Credit scoring model based on neural network with particle swarm optimization. In *International Conference on Natural Computation*, pages 76–79. Springer.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Hand, D. J. and Henley, W. E. (1997). Statistical classification methods in consumer credit scoring: a review. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 160(3):523–541.

Hansen, L. and Sargent, T. (2008). *Robustness.* Princeton University Press.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural networks*, 4(2):251–257.

Hornik, K., Stinchcombe, M., and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366.

Hosseini, H., Chen, Y., Kannan, S., Zhang, B., and Poovendran, R. (2017). Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318*.

Huang, C.-L., Chen, M.-C., and Wang, C.-J. (2007). Credit scoring with a data mining approach based on support vector machines. *Expert systems with applications*, 33(4):847–856.

Huang, R., Xu, B., Schuurmans, D., and Szepesvári, C. (2015). Learning with a strong adversary. *arXiv preprint arXiv:1511.03034*.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. (2018). Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE.

Juuti, M., Szyller, S., Marchal, S., and Asokan, N. (2019). Prada: protecting against dnn model stealing attacks. In *2019 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 512–527. IEEE.

Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*.

Kurakin, A., Goodfellow, I., Bengio, S., Dong, Y., Liao, F., Liang, M., Pang, T., Zhu, J., Hu, X., Xie, C., et al. (2018). Adversarial attacks and defences competition. In *The NIPS'17 Competition: Building Intelligent Systems*, pages 195–231. Springer.

Liang, B., Li, H., Su, M., Bian, P., Li, X., and Shi, W. (2017). Deep text classification can be fooled. *arXiv preprint arXiv:1704.08006*.

Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. (2016). Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582.

Netzer, O., Lemaire, A., and Herzenstein, M. (2019). When words sweat: Identifying signals for loan default in the text of loan applications. *Journal of Marketing Research*, 56(6):960–980.

Orgler, Y. E. (1970). A credit scoring model for commercial loans. *Journal of money, Credit and Banking*, 2(4):435–445.

Óskarsdóttir, M., Bravo, C., Sarraute, C., Vanthienen, J., and Baesens, B. (2019). The value of big data for credit scoring: Enhancing financial inclusion using mobile phone data and social network analytics. *Applied Soft Computing*, 74:26–39.

Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., and Swami, A. (2017). Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia conference on computer and communications security*, pages 506–519.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., and Swami, A. (2016). The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE.

Rosenberg, E. and Gleit, A. (1994). Quantitative methods in credit management: a survey. *Operations research*, 42(4):589–613.

Rozsa, A., Gunther, M., and Boult, T. E. (2016). Towards robust deep neural networks with bang. *arXiv preprint arXiv:1612.00138.*

Stiglitz, J. E. and Weiss, A. (1981). Credit rationing in markets with imperfect information. *The American economic review*, 71(3):393–410.

Su, J., Vargas, D. V., and Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation*, 23(5):828–841.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199.*

Takemura, T., Yanai, N., and Fujiwara, T. (2020). Model extraction attacks against recurrent neural networks. *arXiv preprint arXiv:2002.00123.*

Thomas, L. C. (2000). A survey of credit and behavioural scoring: forecasting financial risk of lending to consumers. *International journal of forecasting*, 16(2):149–172.

Tramèr, F., Kurakin, A., Papernot, N., Goodfellow, I., Boneh, D., and McDaniel, P. (2017). Ensemble adversarial training: Attacks and defenses. *arXiv preprint arXiv:1705.07204.*

Tramèr, F., Zhang, F., Juels, A., Reiter, M. K., and Ristenpart, T. (2016). Stealing machine learning models via prediction apis. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 601–618.

Wei, Y., Yildirim, P., Van den Bulte, C., and Dellarocas, C. (2016). Credit scoring with social network data. *Marketing Science*, 35(2):234–258.

West, D. (2000). Neural network credit scoring models. *Computers & Operations Research*, 27(11-12):1131–1152.

Wiginton, J. C. (1980). A note on the comparison of logit and discriminant models of consumer credit behavior. *Journal of Financial and Quantitative Analysis*, 15(3):757–770.

Xu, W., Evans, D., and Qi, Y. (2017). Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155.*

Yang, C., Wu, Q., Li, H., and Chen, Y. (2017). Generative poisoning attack method against neural networks. *arXiv preprint arXiv:1703.01340.*

Zhang, Y., Jia, R., Pei, H., Wang, W., Li, B., and Song, D. (2019). The secret revealer: Generative model-inversion attacks against deep neural networks. *arXiv preprint arXiv:1911.07135.*

Zhao, Y. (2017). Research on the consumer finance system of ant financial service group. *American Journal of Industrial and Business Management*, 7(5):559–565.