Accountability Through Robust Classification

Anonymous Authors[1]

Abstract

Algorithms are increasingly common components of high-impact decision-making across verticals. A growing body of literature on adversarial examples in laboratory settings suggests that standard machine learning models are not robust, implying that real-world systems are also susceptible to manipulation or misclassification. These findings pose questions about the accountability of machine learning models used in practice. This paper explores the effect of small perturbations on social data that lead machine learning models to deviate from their expected behavior. In particular, we use loan grade classification to explore how classification models can be susceptible to small changes in loan application data. Finally, we show that a robust optimization algorithm can build models that are resistant to misclassification on small perturbations.

1. Introduction

In this paper, we explore the robustness of algorithmic decision-making to small perturbations in input data. In the past decade, algorithmic decision-making systems have been increasingly used in high-stakes settings, for determining insurance packages [1], setting bail [2], and advising policing strategies [3]. These systems often use standard machine learning techniques to detect patterns in data, and provide recommendations based on these patterns. Robustness guarantees against small perturbations in inputs can help enhance model accountability by ensuring classifiers run as expected on different types of small perturbations on inputs.

In the settings we discuss in this paper, incorrect classifications could lead to substantial consequences. This is in contrast to applications like spam detection, where robust machine learning techniques are used to improve user experience. For example, on the recidivism prediction suite COMPAS, an increase in risk score can increase the likelihood people are jailed awaiting trial [4], while on the loan platform LendingClub, a difference in loan grade can lead to a 2-5% change in interest rate [5].

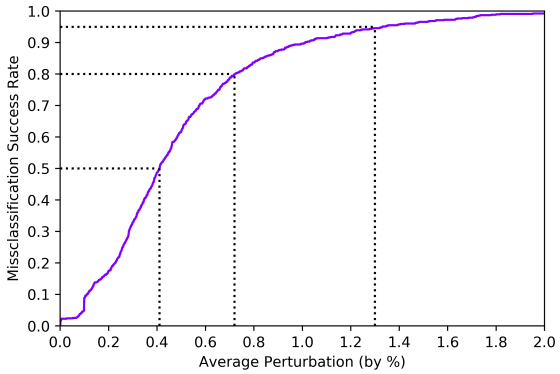

Figure 1. Misclassification of perturbed loan profiles as a function of average perturbation (by percent). Note that a 1.3% average change per feature leads to misclassification on 95% of inputs.

The potential for misclassification on these types of models poses two key questions:

1. *Do small perturbations in input data lead to misclassification?*
2. *Can robust classification help models perform better on perturbed examples?*

We use a dataset of loan applicant information and loan grades from LendingClub [5] as a case study for demonstrating how real-world algorithmic systems can be made more robust. We find that that small perturbations on our high-accuracy loan grade prediction model are successful, with average perturbations of 1.3% per feature leading to misclassification on 95% of inputs (Figure 1). To overcome this issue, we build a robust classifier on adversarially-trained models across perturbation types. We find that our robust classifier achieves an increase in accuracy of 20% against the benchmark when the model is evaluated on the minmax objective. These findings suggest that robust classification can help algorithms better handle inputs not originally accounted for during training.

[1].

Preliminary work. Under review. Do not distribute.

2. Methods

In this section, we introduce the dataset, perturbations, and the robust optimziation algorithm used throughout the paper.

2.1. LendingClub Data

We use loan application data from a set of 421,095 applications submitted to LendingClub in 2015 [5]. LendingClub is an online peer-to-peer lending platform that algorithmically assigns loan categories based on application data. LendingClub and other online loan services base their loan categories on several user-inputted features in order to achieve more accurate credit assessment than traditional loan services—including banks, which mainly use FICO credit scores. Each applicant is assigned a grade from A to G by the platform, where A assigns the lowest interest rates (between 5-8%) and G assigns the highest rates (between (25-29%). LendingClub also assigns applicants to one of five subgrades per category (for example, A is divided between A1-A5). We predict the probability of each applicant belonging to each of the seven grade categories in order to enforce a larger degree of perturbation by our adversarial methods.

Naturally, we do not know the exact model that LendingClub platform uses to make its decisions since the platform considers its model proprietary data. To overcome this, we train a neural network as a proxy for the classification system built by LendingClub. Specifically, we train a neural network to predict loan grade from 44 continuous features including FICO score, debt-to-income ratio, and loan-to-income ratio, and 6 discrete features, including state and filing status. Our network uses 2 dense ReLU layers of size 100 and 60, with a dropout of 0.2 and a final softmax layer to define predictions for 7 possible loan grades. This architecture is based off Andersen's architecture for approaching this classification problem [6]. We achieve an accuracy of 94.46%.

2.2. Pertubation Types

We evaluate a number of adversarial methods on the heterogeneous feature space of our model, targetting the 44 continuous features. Adversarial attacks perturb an input up to a designated noise budget, under the intuition that with a sufficiently small budget any perturbation in an input is "meaningless" in terms of actually changing the ground truth classification of the input.

Fast Gradient Sign Method (FGSM) perturbs an input in the direction of the gradient of the loss function of the model with respect to the input [7]. The size of the perturbation of each feature is a constant parameter ϵ. Given a loss function $L(\theta, x, y)$ with parameters θ, input x, and label y, FGSM produces the perturbation

$$x \quad = x + \epsilon \operatorname{sign}(\nabla_x L(\theta, x, y)).$$

Projected Gradient Descent (PGD) is a similar process to FGSM, but descends the actual gradient, as opposed to the sign [8]. Additionally, the gradient is recomputed at every step. The maximum perturbation for each iteration is clipped by a parameter ϵ. PGD iteratively produces perturbed examples x_i (up to the maximum step x_N), following the procedure

$$\begin{aligned} x_0 &= x, \\ x_i &= \operatorname{clip}_\epsilon(x_{i\ 1} + \epsilon(\nabla_{x^*_{i-1}} L(\theta, x_{i\ 1}, y))). \end{aligned}$$

Jacobian-based Saliency Mapping Attack (JSMA) isolates and perturbs the features with highest impact on the loss of a target label [9]. JSMA greedily increases the probability of the target label while decreasing the probabilities of all other labels. With $F_j(x)$ as the output of a neural network for class j, a saliency map is defined as

$$S(x,t)[i] = \begin{cases} 0, \text{ if } \frac{\partial F_t(x)}{\partial x_i} < 0 \text{ or } \sum_{j \neq t} \frac{\partial F_j(x)}{\partial x_i} > 0 \\ (\frac{\partial F_t(x)}{\partial x_i}) \left| \sum_{j \neq t} \frac{\partial F_j(x)}{\partial x_i} \right|, \text{ otherwise} \end{cases}$$

JSMA perturbs the features with the highest impact on salience by a constant perturbation ϵ.

Max Salience Attack (MSA) is an attack we propose. This attack mimics self-report manipulation, where a loan applicant might change a small number of inputs to try to affect loan grade. Specifically, we perturbed each feature individually by a set percent increase or decrease, and found which of these features produced the highest rate of successful misclassifications. We then perturbed one and two of these most salient features (1-MSA, 2-MSA) by a fixed percent change to model worst-case perturbations by an agent with access to changing only a small number of features with no knowledge of neural network gradient.

Finally, we standardize each feature in the training and perturbation procedures before applying the perturbation. This is because, unlike the dataset perturbations are typically applied to (e.g. images), the features relevant to loan grade prediction are not drawn from the same supports.

2.3. Robust Optimization Algorithm

To make our loan grade prediction classifier robust against these perturbations, we train our model with the robust optimization algorithm proposed by Chen et al. [10]. The key idea of the algorithm is that, given a model (e.g. a neural network) and a set of perturbations, the algorithm can identify which perturbations have high classification

Algorithm 1 Oracle Efficient Improper Robust Optimization [10]

Input: Objectives $L = \{L_1, \ldots, L_m\}$, Apx Bayesian Oracle M, parameters T, η
for $t = 1$ **to** T **do**
  $w_t[i] \propto \exp \eta \left(\sum_{\tau=1}^{t} L_i(h_\tau) \right)$
  $h_t = M(w_t)$
end for
Output: the uniform distribution over $\{h_1, \ldots, h_T\}$

losses, and focus more on those perturbation types during the training.

More formally, given a distribution $D = (D_1, \ldots, D_m)$ over loss functions $L = \{L_1, \ldots, L_m\}$, we would like to minimize the loss in the worst-case over L, namely $\tau = \min_{h \in H} \max_{i \in [m]} L_i(h)$. We call this the bottleneck loss throughout the rest of this paper.

Moreover, we assume that we are given an α-approximate Bayesian oracle M. That is, $M(D)$ can find a hypothesis h that α-approximates the minimum expected risk

$$\mathbb{E}_{L \sim D}[L(h)] \leq \alpha \min_{h \in H} \mathbb{E}_{L \sim D}[L(h)]$$

In our setting, L corresponds to the loss of the model over different perturbations. Furthermore, we assume that the neural network trained over the perturbations is an α-approximate Bayesian oracle.

The algorithm for optimizing this objective can be thought of as a no-regret dynamics on a zero-sum game. Concretely, the algorithm runs a multiplicative weight update (MWU) over L (Algorithm 1). Intuitively, the algorithm first assigns uniform importance weights to all perturbation types. Then in each iteration, it puts more weights to perturbations with higher loss.

Under this framework, Chen et al. [10] show that when $\eta = \sqrt{\log(m)/2T}$, the ensemble hypothesis $h = \frac{1}{T} \sum_{t=1}^{T} h_t$ satisfies

$$\max_{i \in [m]} L_i(h) \leq \alpha \min_{h \in H} \max_{i \in [m]} L_i(h) + \sqrt{\frac{2 \log(m)}{T}}$$

In other words, the model can be robust against a distribution of perturbations, with the same α multiplicative factor loss, and an additive error that goes to zero as iteration increases.

3. Evaluation of Perturbations

We implemented perturbation methods on our loan category prediction model, varying the perturbation size and evaluating the success of changing the predicted label. We evaluated the extent to which perturbed train examples were misclassified as the perturbation budget increased.

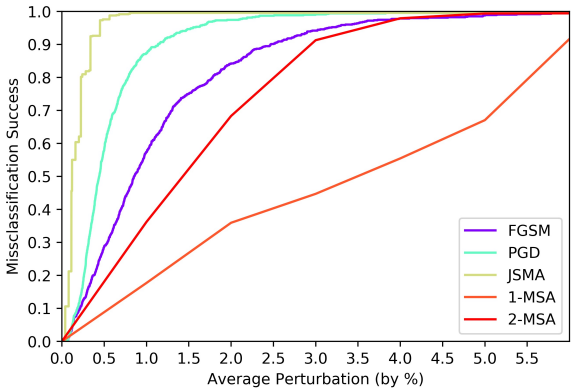

Figure 2. Perturbations against a pre-trained classification model. The horizontal axis is the average percentage change in the perturbed features. The vertical axis is the rate of originally correctly classified samples that were misclassified upon small perturbations.

We found that small perturbations were effective at perturbing data by small amounts to change loan categories (Figure 2). In particular, JSMA demonstrated an ability to target specific features that significantly affected the neural network's loan category prediction.

FGSM, which is based on perturbing all features by a constant amount, was more inefficient with budgeting perturbation size compared to other methods. PGD likely achieved higher perturbation efficiency by not perturbing features with low salience as much.

1-MSA, which only changed loan-to-income ratio for applicants, performed quite well, despite its constraints. MSA did not utilize gradients of the neural networks, which put it at a distinct disadvantage compared to other networks. However, MSA demonstrates that even a small change in a single, preselected variable can easily impact loan category classification. A loan applicant could easily round off this input (by rounding either income or loan) and change their loan category. Meanwhile, 2-MSA, which only extends upon 1-MSA by perturbing one additional variable, achieves significantly better results. Overall, MSA suggests that traditional, gradient-based methods are not necessary to produce small perturbations for successful misclassifications.

4. Robust Classification

We experimentally show that the robust optimization algorithm introduced in [10] improves the performance of our loan grade prediction model. Specifically, we train a neural network model with the same architecture as in Section 2.1. We train the model over the five perturbations introduced in Section 2.2 as well as the original dataset with no perturbations (referred to as None).

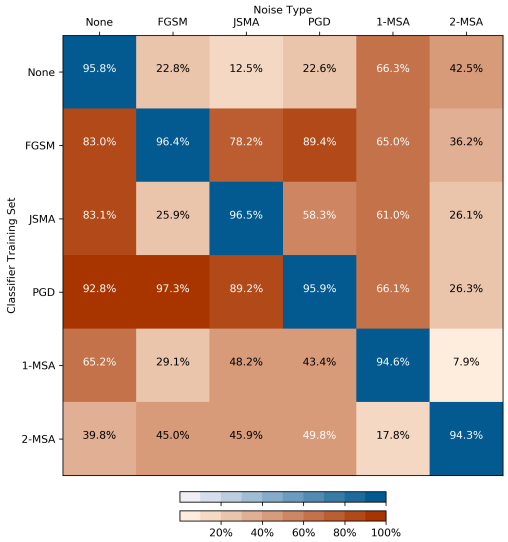

Figure 3. Payoff matrix representing the minmax problem being addressed by the robust optimization algorithm for loan classification.

As stated in Section 2.3, the robust optimization framework can be thought of as a minmax game. In particular, it is a two player game where one player has classifiers as strategies and another player has perturbation types as strategies. The payoff matrix of the specific game played in this loan grade prediction optimization is shown in Figure 3. It can easily be seen that if the row player commits to a classifier trained on a specific perturbed dataset, the column player can easily choose a different perturbation type to have misclassifications. This motivates the use of Algorithm 1, which is essentially estimating the optimal mixed strategy over the different classifiers for the row player.

The evaluation of the algorithm is based on the bottleneck loss as introduced in section 2.3. The baseline is a training with fixed uniform distribution, meaning that we do not update the distribution over perturbation types at each iteration of the algorithm. This can be thought of as uniformly randomizing over the row strategies in Figure 3.

The result of the robust optimization is shown in Figure 4. The horizontal axis is the number of iterations (T in Algorithm 1). For the top plot, the vertical axis is the bottleneck loss, and for the bottom plot, it is the accuracy corresponding to that loss. The robust optimization algorithm reduced the bottleneck loss from 1.0 to 0.6 and increased the bottleneck accuracy from 50% to 70%. The loan grade prediction model trained with the robust optimization algorithm is clearly more robust against perturbations.

5. Conclusion

While small perturbations in input data can lead loan grade models to misclassify, building robustness into these classifiers can help ensure that models classify as expected on inputs that are slightly different from those the model was

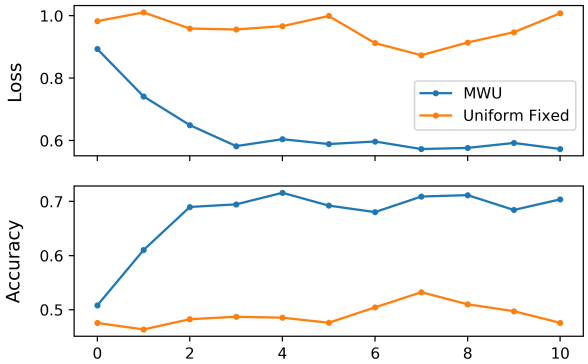

Figure 4. Bottleneck loss (top) and accuracy (bottom) over 10 iterations of the robust optimization algorithm.

trained on. Ensuring that a model will behave as specified through robust training can build algorithmic accountability.

Future work in this space would benefit from more exploration into the feasible set of perturbations that lead to misclassification. By better understanding this feasible set for individual domains, robust classification could help make algorithmic systems build stronger decision boundaries across problem types. Furthermore, an analysis into whether weaker decision boundaries exist for different classes of individuals could help highlight issues of fairness or differential robustness. Finally, exploring more domains and models outside of laboratory conditions would help better inform how robustness can be employed in practice.

References

[1] Kumba Sennaar. How america's top 4 insurance companies are using machine learning -. *TechEmergence*, 2018.

[2] Ellora Israeli. Opinion | when an algorithm helps send you to prison. *Nytimes.com*, 2018.

[3] Mick Dumke and Frank Main. A look inside the watch list chicago police fought to keep secret. *Chicago Sun-Times*, 2018.

[4] Sam Corbett-Davies, Emma Pierson, Avi Feller, and Sharad Goel. A computer program used for bail and sentencing decisions was labeled biased against blacks. it's actually not that clear. *Washington Post*, 2016.

[5] LendingClub Loan Data. *https://www.lendingclub.com/info/download-data.action*, 2018.

[6] James Andersen. Predicting loan grades with a neural network: A machine learning pipeline on aws. *Medium*, 2018.

[7] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[8] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[9] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pages 372–387. IEEE, 2016.

[10] Robert S Chen, Brendan Lucier, Yaron Singer, and Vasilis Syrgkanis. Robust optimization for non-convex objectives. In *Advances in Neural Information Processing Systems*, pages 4708–4717, 2017.