GARFD: Gradient-based Autoregressive Forecaster Defense

Mackenzie Stein and Scott Alfeld
{mstein19, salfeld}@amherst.edu
Department of Computer Science
Amherst College

Abstract

Machine learning and data analysis methods are vulnerable to data manipulation attacks. In this work we assist the learner by acting at training-time to prepare against deployment-time attacks. We present GARFD — Gradient-based AutoRegressive Forecaster Defense. We utilize matrix differential calculus to extend prior results of adversarial regularization to predictors that are linear in the input features but not necessarily linear in the model parameters. This generalization allows our regularization method to be applied to linear autoregressive forecasters. We demonstrate the effectiveness of GARFD empirically on real-world and synthetic data, and find that typically a very large decrease (e.g., 70%) in vulnerability is achievable for a comparatively low (10%) increase in loss.

1 Introduction

In this paper we present GARFD — Gradient-based AutoRegressive Forecaster Defense. Adversarial regularization involves incorporating knowledge of an attacker's capability into the training process to learn a more robust model. Prior work of Lyu et al. [6] present a general approach for using approximate gradient information of the attacker-induced loss when training. In our work, we compute the derivative of the vulnerability exactly for predictors that are linear in the input variables (features). This allows our method to be applied to autoregressive forecasters, to be made mathematically explicit below. It also serves as a generalization of regularization for p-norm constrained attacks (c.f., the work of Xu et al. [8]) to matrix (rather than vector) norms.

We make a standard set of assumptions (c.f., Vorobeychik and Kantarcioglu, 2018 [7]) and for clarity we make the setting of this paper explicit. We frame our discussion in terms of two actors: Bob the defender and Alice the adversary. The learner/defender Bob receives a clean (unperturbed by any attacker) training set. Bob then trains a model $\boldsymbol{\theta}$. Bob deploys the model to predict a test instance which has been perturbed by the attacker Alice. In the parlance of adversarial learning, we act at training time learn a model more robust against decision-time attacks. We assume that the attacker Alice has full knowledge of Bob's learned model, the test instance, and bounded ability (to be made formal below) to perturb the test instance. We assume Bob knows Alice's constraints (what perturbations she can make), but does not know the test instance (during training time), nor does he know Alice's target/intention.

2 Mathematical Preliminaries

In Autoregressive forecasting, a learner considers a time series $\boldsymbol{v}$ and uses past values of the time series to predict future values of it. In Linear Autoregressive Forecasting Bob uses a linear model denoted $\boldsymbol{\theta}$ and a vector of data denoted $\boldsymbol{x}$ to forecast the future values. The data is indexed by time, with each evenly spaced period having a corresponding value. For clarity of exposition, we consider the units to be in terms of days; this allows us to use familiar terms such as "yesterday" and "tomorrow." We then use prior days' data to generate predictions for future days.

We denote the *order* – the number of days used to generate each prediction – as d and the *horizon* – the number of days into the future predicted – as h. There are many ways to use a linear model to forecast $h > 1$ days into the future (see Bontempi et al. [2] for a more detailed discussion). In this work we focus on the *recursive forecasting* method, but note that the methods are readily applicable to other forms of forecasting as well. For each prediction, we use the values of the previous d time periods in our computation, recursively generating predictions for h days.

We denote previous days' data as x_i and future days' forecasts as $\hat{x}_i$, with i indicating the distance from the present. That is to say x_1 is yesterday, x_2 is two days ago, etc., while $\hat{x}_1$ is tomorrow, $\hat{x}_2$ is two days into the future, etc. [1] In this case, we have the weights in our model correspond with a given number of days into the past: θ_1 corresponds with 1 day prior, θ_2 with 2 days prior, and so on.

With notation from Alfeld et al. [1] we use an $h \times h$ *one-step* matrix $\boldsymbol{S}$ and $h \times d$ *zero-pad* matrix $\boldsymbol{Z}$:

$$\boldsymbol{S} \stackrel{\text{def}}{=} \left[\begin{array}{c|cc} \boldsymbol{0}_h & \multicolumn{2}{c}{\boldsymbol{I}_{(h-1)\times(h-1)}} \\ \hline & \boldsymbol{0}^\top_{(h-d-1)\times 1} & \overleftarrow{\boldsymbol{\theta}}^\top \end{array}\right], \boldsymbol{Z} \stackrel{\text{def}}{=} \begin{bmatrix} \boldsymbol{0}_{(h-d)\times d} \\ \boldsymbol{I}_{d\times d} \end{bmatrix}$$

where $\overleftarrow{\boldsymbol{\theta}} = [\theta_d, \ldots, \theta_1]^\top$. This yields $\hat{\boldsymbol{x}} = \boldsymbol{S}^h \boldsymbol{Z}\boldsymbol{x}$ where $\boldsymbol{x} = [x_d, \ldots, x_1]^\top$ and the $h \times 1$ vector of forecasts $\hat{\boldsymbol{x}} = [\hat{x}_1, \ldots, \hat{x}_h]^\top$. Note that the above formulation requires that we have $h > d$. If $h \leq d$, this is easily accounted for by forecasting $d + 1$ days and ignoring some forecasts.

2.1 Attacking Linear Predictors

We utilize the setting presented by Alfeld et al. [1], which we repeat here in the interest of being self contained. Given an input sequence $\boldsymbol{x}$, Alice selects a perturbation vector $\boldsymbol{\alpha}$ from her available set of perturbations $\mathcal{A}$. The sum $\boldsymbol{x} + \boldsymbol{\alpha}$ is then given to Bob, who makes a prediction $f(\boldsymbol{x} + \boldsymbol{\alpha})$ and suffers due to the inaccuracy caused by Alice's manipulation. In performing a decision-time attack, the attacker may attempt to pull predictions toward her target ("attractive" attacks) or drive them as far as possible from some value ("repulsive" attacks). Because we seek to defend against the worst-case attacker, we model Alice as explicitly attempting to maximize Bob's loss (a repulsive attack):

$$\boldsymbol{\alpha}^{\text{rep}}(\mathcal{A}, \boldsymbol{x}, ||\cdot||_B) \stackrel{\text{def}}{=} \arg\max_{\boldsymbol{\alpha}\in\mathcal{A}} ||f(\boldsymbol{x} + \boldsymbol{\alpha}) - f(\boldsymbol{x})||_B \tag{1}$$

where $||\cdot||_B$ denotes Bob's loss. We utilize the following theorem:

Theorem 1. *(Alfeld et al. [1])*

Let $\boldsymbol{\alpha}^{rep}$ be the optimal repulsive attack against a linear predictor Bob with prediction function $f(\boldsymbol{x}) = \boldsymbol{M}\boldsymbol{x}$ and (squared Mahalanobis) loss $||f(\boldsymbol{\alpha})||_B = ||f(\boldsymbol{\alpha})||^2_{\boldsymbol{W}} = f(\boldsymbol{\alpha})^\top \boldsymbol{W} f(\boldsymbol{\alpha})$, where $\boldsymbol{W} = \boldsymbol{V}^\top\boldsymbol{V}$ is a positive definite matrix, under the general ball constraints $\mathcal{A} = \{\boldsymbol{\alpha} : ||\boldsymbol{\alpha}||_{\boldsymbol{C}} \leq c\}$ for the attacker, where $\boldsymbol{C} = \boldsymbol{G}^\top\boldsymbol{G}$ is a positive definite matrix and $c \in \mathbb{R}_+$ is a constant. Then the repulsive attack as defined in (1) is equal to $c\boldsymbol{G}^{-1}\boldsymbol{s}_1$ where $\boldsymbol{s}_1$ is the right-singular vector corresponding to the largest singular value of $\boldsymbol{V}\boldsymbol{M}\boldsymbol{G}^{-1}$. Further, Bob's loss induced by $\boldsymbol{\alpha}^{rep}$ is the squared spectral norm of $c\boldsymbol{V}\boldsymbol{M}\boldsymbol{G}^{-1}$.

We define $\texttt{vulnerability}(\boldsymbol{\theta}, \boldsymbol{G}^{-1})$ as Bob's loss when using model $\boldsymbol{\theta}$ induced by an attacker with constraints defined by $\boldsymbol{G}^{-1}$. We note that, in the AR (recursive) forecasting setting, $\boldsymbol{M} = \boldsymbol{S}^h\boldsymbol{Z}$, but the methods presented here are applicable to general linear predictors[2]. The linearity (in $\boldsymbol{x}$) of such predictors is the underlying reason that $\boldsymbol{x}$ does not appear in the optimal attack or loss induced by it. Note, however, that our AR forecaster (when $h > 1$) is non-linear in $\boldsymbol{\theta}$, making Bob's vulnerability a non-convex function of $\boldsymbol{\theta}$.

[1]Note that x_0 does not exist so as to avoid ambiguity. Instead we act between yesterday and tomorrow.

[2] We focus on Bob's loss being quadratic and on Alice being constrained by a quadratic norm. As discussed in Alfeld et al. [1], their defense methods are also applicable to other settings (namely ℓ_1 and ℓ_∞ norms). The underlying problem of finding $\boldsymbol{\alpha}^{\text{rep}}$ is reduced to computing an induced matrix norm for each norm. Our methods are equally applicable to these settings with minor variations to the derivation of derivatives. Similarly, our focus is on autoregressive forecasting, but our methods are applicable to any linear predictor $f(\boldsymbol{x}) = \boldsymbol{M}\boldsymbol{x}$.

3 GARFD: Proposed Method

By utilizing matrix differential calculus, we compute derivatives of Bob's vulnerability, enabling gradient-based optimization methods. The key idea is to account for the model's vulnerability as we train, and balance its accuracy and robustness. When Bob learns his model from training set $\boldsymbol{X}$, he seeks to minimize the objective function, $J_{\boldsymbol{X}}(\boldsymbol{\theta})$. Traditionally, the defender focuses on learning a model that is accurate to the data (which here we assume to be uncorrupted), and after the fact will implement a defense strategy to account for manipulated data at decision-time. This means the objective function is simply the loss on the training set (perhaps with a regularization term in addition). In our new strategy, our defense is part of learning the model, and thus Bob's objective function must balance accuracy and robustness:

$$J_{\boldsymbol{X}}(\boldsymbol{\theta}) = \rho * \texttt{loss}(\boldsymbol{X};\boldsymbol{\theta}) + (1-\rho) * \frac{\texttt{vulnerability}(\boldsymbol{\theta})}{h} \tag{2}$$

Note that the factor $1/h$ is due to the vulnerability and loss being of different scale. Where loss is normalized by the number of points in $\boldsymbol{X}$, the vulnerability as defined by Theorem 1 scales with h. So as to directly compare the two, we scale the vulnerability down by the hoirzon h.

We use the weight ρ indicate how much Bob values one measure over the other. At $\rho = 1.0$ Bob performs traditional learning with no fear of an attacker. At $\rho = 0.0$ Bob ignores the loss of the model and seeks only to minimize his vulnerability. We note that this is accomplished by setting $\boldsymbol{\theta} = \boldsymbol{0}$, an action rarely advisable in practice. For $\rho \in (0, 1.0)$, Bob balances his model's performance against its vulnerability to attack.

Let $\boldsymbol{Q}_1 = (c\boldsymbol{V}\boldsymbol{S}^h\boldsymbol{Z}\boldsymbol{G}^{-1})$ and $\boldsymbol{Q}_2 = \boldsymbol{Q}_1^\top\boldsymbol{Q}_1$ with terms defined as in Theorem (1). We compute the derivative of $\lambda_0(\boldsymbol{Q}_2)$ with respect to $\boldsymbol{\theta}$. We note that differentiating the training loss with respect to $\boldsymbol{\theta}$ is standard practice and solutions are known for common loss functions. We have:

$$D_{\boldsymbol{\theta}}\lambda_0(\boldsymbol{Q}_2) = \left(\boldsymbol{u}_0^\top \otimes \boldsymbol{u}_0^\top\right)(D_{\boldsymbol{\theta}}\boldsymbol{Q}_2) \tag{3}$$

$$(D_{\boldsymbol{\theta}}\boldsymbol{Q}_2) = (\mathcal{K}_{dd} + \mathcal{I}_{d^2})\left(\mathcal{I}_d \otimes \boldsymbol{Q}_1^\top\right) D_{\boldsymbol{\theta}}\boldsymbol{Q}_1 \tag{4}$$

$$D_{\boldsymbol{\theta}}\boldsymbol{Q}_1 = \left(\left(\boldsymbol{Z}\boldsymbol{G}^{-1}\right)^\top \otimes \mathcal{I}_h\right)(\mathcal{I}_h \otimes (c\boldsymbol{V}))\, D_{\boldsymbol{\theta}}\left(\boldsymbol{S}^h\right) D_{\boldsymbol{\theta}}\left(\boldsymbol{S}^h\right) \quad = D_{\boldsymbol{S}}\left(\boldsymbol{S}^h\right) D_{\boldsymbol{\theta}}\boldsymbol{S} \tag{5}$$

$$D_{\boldsymbol{S}}\left(\boldsymbol{S}^h\right) = \sum_{i=1}^{h}\left(\boldsymbol{S}^\top\right)^{h-i} \otimes \boldsymbol{S}^{i-1} \tag{6}$$

where $\mathcal{K}_{dd}$ denotes the $d^2 \times d^2$ commutation matrix. To obtain (5) we utilize the differential product rule twice (letting $\boldsymbol{A} = c\boldsymbol{V}, \boldsymbol{B} = \boldsymbol{S}^h\boldsymbol{Z}\boldsymbol{G}^{-1}$, followed by $\boldsymbol{A}' = \boldsymbol{S}^h, \boldsymbol{B}' = \boldsymbol{Z}\boldsymbol{G}^{-1}$) and the chain rule. We note that (6) follows from repeated application of the matrix differential product rule. To compute $D_{\boldsymbol{\theta}}\boldsymbol{S}$ we utilize the definition of matrix derivatives:

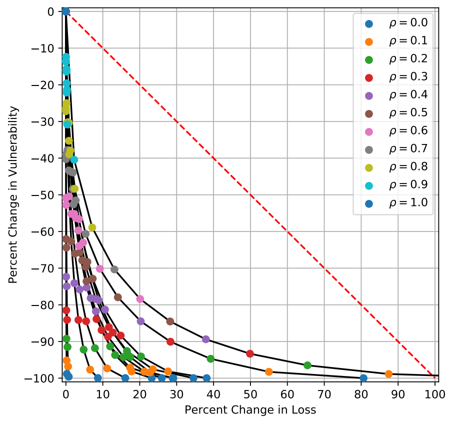
$$D_{\boldsymbol{\theta}}\boldsymbol{S} = \frac{d\texttt{Vec}(\boldsymbol{S})}{d\texttt{Vec}(\boldsymbol{\theta})^\top},\ (D_{\boldsymbol{\theta}}\boldsymbol{S})_{ij} = \frac{d\texttt{Vec}(\boldsymbol{S})_i}{d\texttt{Vec}(\boldsymbol{\theta})_j} \tag{7}$$

4 Experiments

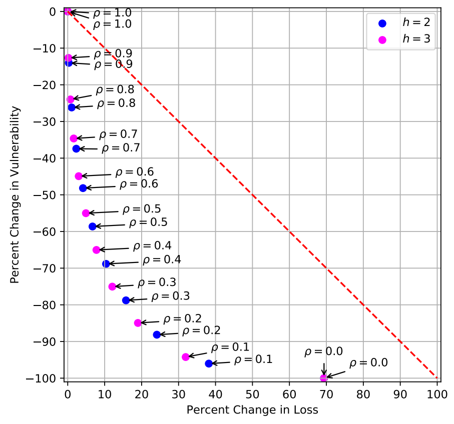
All experiments were run on modern consumer hardware using Google Compute Engine and run-time was not a concern or limiting factor. Figures presented were created with Matplotlib [5].

We begin by considering an attacker motivated by a political agenda around climate change. We use minimum daily temperatures for Melbourne, Australia [4]. For each of the 12 months in 1981, we train 11 autoregressive forecasters with $h = 2, 3$, $d = 2$. The forecasters differ in how much they weight vulnerability vs accuracy: $\rho \in [0, 0.1, \ldots 0.9, 1.0]$. For each learner and each month, we compute the percentage increase in (training) loss and the percentage decrease in vulnerability. Results are shown in Figure (1a).

We further explore the effects of GARFD with synthetic data. We answer the following question in the affirmative: When the choice of model is fully appropriate for the underlying data, is the loss in accuracy worth the gain in robustness to attack? To do so, we generate 1,000 order $d = 2$ models by sampling uniformly from the set of models where the modulus of each root of the characteristic



(a) **12 Months of Daily Temperatures.** Points of the same color correspond to the same learner (the same value of ρ). Points on the same black line were trained using the same month's data.



(b) **Forecasting Synthetic Data.** Each point shows the average over 1000 models trained with a particular value of ρ. The data generation process matches the assumptions made in autoregressive forecasting.

Figure 1: **Effects of GARFD.** We applied GARFD to learners for both Melbourne climate and synthetic data. In both cases, we report the reduction in vulnerability (vertical axis) and increase in loss (horizontal axis). Note that along both axes, lower is better. Each point is a model (in the case of synthetic data, the average of 1000 models) trained with GARFD with various weightings (ρ) between robustness and accuracy. A model being below the diagonal (shown as a dashed red line) means that the percentage decrease in vulnerability is greater than the percentage decrease in loss. For nearly every setting of ρ, GARFD helps the model's robustness more than it hurts its accuracy.

polynomial is greater than or equal to 1.1. This ensures that the models are stationary [3] and avoids numerical issues (e.g., when a model is border-line stationary[3]). We focus our attention on stationary models due to their prevalence and applicability in practice. For each model $\boldsymbol{\theta}^{\text{ex}}$, we generate a time series of values. That is, each value $x_i^{\text{truth}} = \theta_1^{\text{ex}} x_{i-1}^{\text{truth}} + \theta_2^{\text{ex}} x_{i-2}^{\text{truth}} + \epsilon_i$ where each ϵ_i is drawn iid from the normal distribution $N(0, 1)$.[4] We define Bob's loss as the mean squared error (MSE) measured between the generated time series and his one-step-ahead predictions, as is standard for AR processes [3]. For clarity of exposition, we let Bob weigh each prediction equally ($\boldsymbol{V} = \mathcal{I}$) and Alice be constrained to select $\boldsymbol{\alpha}$ from an h-dimensional hypersphere ($\boldsymbol{G} = \mathcal{I}$) with $c = 1.0$.

We then consider 11 different learners ($\rho = 0.1, 0.2, \dots, 0.9, 1.0$), which each learn a model (using 10,000 iterations of gradient descent, which was found in preliminary experiments to be enough to converge). We compute the percentage change in loss and vulnerability compared to $\rho = 1.0$ (learning with no notion of vulnerability) for each value of ρ. These values are displayed in Figure 1b. As with the experiments using temperature data, the reduction in vulnerability exceeds the increase in loss, in many cases by a very large margin. For example, with $\rho = 0.5$, the resulting model has less than half the vulernability with an increase in loss of less than 10%.

5 Conclusion

In this work we extend adversarial regularization to learners with a prediction function linear in input but not necessarily in model parameters. This generalizes past results and extends the capability (of incorporating knowledge of the attacker into the learning process) to autoregressive forecasters. Through an empirical investigation on real-world and synthetic data, we found that our methods substantially reduce model vulnerability with comparatively little increase in loss. This work was supported by a post-baccalaureate fellowship awarded by the Office of the Provost and dean of the Faculty at Amherst College.

[3]We note that one of the one-thousand trials was removed due to numerical instability.

[4]In addition, two iid values drawn from $N(0, 1)$ were used to seed the recursion.

References

[1] Scott Alfeld, Xiaojin Zhu, and Paul Barford. Explicit Defense Actions Against Test-Set Attacks. In *Proceedings of the 31st Conference on Artificial Intelligence (AAAI 2017)*, 2017.

[2] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. Machine learning strategies for time series forecasting. In *European business intelligence summer school*, pages 62–77. Springer, 2012.

[3] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

[4] Jason Brownlee. *Introduction to time series forecasting with python: how to prepare data and develop models to predict the future*. Machine Learning Mastery, 2017.

[5] J. D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9 (3), 2007.

[6] Chunchuan Lyu, Kaizhu Huang, and Hai-Ning Liang. A unified gradient regularization family for adversarial examples. In *2015 IEEE International Conference on Data Mining*, pages 301–309. IEEE, 2015.

[7] Yevgeniy Vorobeychik and Murat Kantarcioglu. Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169, 2018.

[8] Huan Xu, Constantine Caramanis, and Shie Mannor. Robustness and regularization of support vector machines. *Journal of Machine Learning Research*, 10(Jul):1485–1510, 2009.