Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Data Poisoning Attacks against Autoregressive Models

Scott Alfeld,* **Xiaojin Zhu,*** and **Paul Barford***†
*Department of Computer Sciences
University of Wisconsin – Madison
Madison WI 53706, USA
†comScore, Inc.
11950 Democracy Drive, Suite 600
Reston, VA 20190, USA.
{salfeld, jerryzhu, pb}@cs.wisc.edu

Abstract

Forecasting models play a key role in money-making ventures in many different markets. Such models are often trained on data from various sources, some of which may be untrustworthy. An actor in a given market may be incentivised to drive predictions in a certain direction to their own benefit. Prior analyses of intelligent adversaries in a machine-learning context have focused on regression and classification. In this paper we address the non-iid setting of time series forecasting. We consider a forecaster, Bob, using a fixed, known model and a recursive forecasting method. An adversary, Alice, aims to pull Bob's forecasts toward her desired target series, and may exercise limited influence on the initial values fed into Bob's model. We consider the class of linear autoregressive models, and a flexible framework of encoding Alice's desires and constraints. We describe a method of calculating Alice's optimal attack that is computationally tractable, and empirically demonstrate its effectiveness compared to random and greedy baselines on synthetic and real-world time series data. We conclude by discussing defensive strategies in the face of Alice-like adversaries.

Introduction

Forecasting is important in a variety of markets including commodities, energy, new products and others. For example in commodity markets, forecasting volatility is the basis for the pricing of options contracts. In general, high volatility forecasts result in higher prices for options. Better forecasts result in better returns.

A standard assumption in forecasting is that data used in models is reported honestly *i.e.,* in a way that does not seek to influence the outcome. However, this may not be the case. Indeed, one can easily imagine an actor with a profit motive and access to inputs that will attempt to influence outcomes to her own advantage.

In this paper we consider linear, autoregressive forecasting models. The forecaster Bob uses an order-d linear autoregressive model: $x_t = \alpha + \sum_{i=1}^{d} \theta_i (x_{t-i})$, and forecasts h future values with a recursive strategy. We illustrate the concecpts described herein with a running "One Week" example where Bob uses an order 2 AR process to forecast $h = 3$ days into the future. Specifically, Bob uses the values of Monday and Tuesday to forecast Wednesday, Thursday, and Friday with model $\boldsymbol{\theta} = [0.5, -0.6]^\top$. Thus, for the One Week example, if Monday's value is 1 and Tuesday's is 2, Bob forecasts Wednesday as $0.5 \times 2 - 0.6 \times 1 = 0.4$, he then forecasts Thursday as $0.5 \times 0.4 - 0.6 \times 2 = -1$, and Friday as $0.5 \times (-1) - 0.6 \times 0.4 = -.74$.

With a linear AR model and recursive forecasting strategy, Bob is especially vulnerable to intelligent adversaries – once Alice has observed enough sequential forecasts, she can infer the underlying model exactly given minimal additional information. If Alice observes Bob forecast $0.4, -1, -.74$, and she knows that Tuesday's value is 2, then she only need know $d = 2$ to deduce Bob's model.

Our specific contributions are as follows. (i) We present a general, mathematical framework for defining Alice's optimal attack. (ii) We demonstrate how real world settings may be encoded into our framework such that the resulting optimization problem is solvable with standard convex optimization methods. (iii) We empirically demonstrate the effectiveness of the optimal attack, compared to random and greedy baselines using both synthetic and real world data.

Mathematical Formulation

We use boldface lowercase letters to denote column vectors, and capital letters to denote matrices. We denote element-wise inequalities with $\succeq, \preceq$, and equality by definition as $\triangleq$. Elements of a vector $\boldsymbol{v}$ are v_i. Bob's first prediction is for time 0, and $x_i,\ i < 0$ denote past values. For ease of notation, we ignore the offset term α to obtain the AR(d) model, standard in time series analysis literature (Box, Jenkins, and Reinsel 2011). We note, however, that the methods describe here directly extend to including α.

Alice's objective is to move Bob's forecasts as close as possible to her desired target $\boldsymbol{t} \in \mathbb{R}^h$. Alice's actions are to augment the initial values, $\boldsymbol{x}$, fed into Bob's model, by selecting a vector $\boldsymbol{\delta} = [\delta_{-d}, \ldots, \delta_{-1}]^\top$ and augmenting

$x_{-i} \leftarrow x_{-i} + \delta_{-i}$. She may have restrictions and a cost function over the space from which she selects $\boldsymbol{\delta}$. We denote Bob's predictions, after Alice has poisoned the initial values, as $\hat{\boldsymbol{t}}_{\boldsymbol{\delta}}$.

Consider again the One Week example. Alice will augment Monday and Tuesday's by selecting a vector $[\delta_{\text{Monday}}, \delta_{\text{Tuesday}}]^\top$ so as to bring the values for the rest of the week closer to her target $[t_{\text{Wednesday}}, t_{\text{Thursday}}, t_{\text{Friday}}]^\top$.

We consider a powerful attacker. Namely, Alice observes all $x_{-d}, \dots x_{-1}$ before selecting her attack. This assumption is applicable in settings where Alice is "cooking the books" and may misreport past values. Another canonical real-world setting is where Alice is constrained to determine her attack sequentially. That is, she must select δ_{-d} on day $-d$, before observing $x_{-d+1}, \dots, x_{-1}$. We leave this setting as future work, and note that Alice is weaker in the later case – her best online attack is bounded by her optimal attack in the setting presented herein.

To formally state Alice's optimal attack, we first introduce the following notation. Let the $(h \times 1)$ vector $\boldsymbol{x}_k$ be the last h values of the time series ending at time k: $\boldsymbol{x}_k = [x_{k-(h-1)}, \dots, x_k]^\top$. When $k < h - d$, we zero-pad the front of $\boldsymbol{x}_k$. We assume $h \geq d$ without loss of generality. If $d > h$, we may artificially increase the forecast horizon to d, and encode the fact that Alice does not care about the inflated portion of the horizon into her loss function. We then define the $h \times h$ one-step matrix S and the $h \times d$ matrix Z to zero-pad $\boldsymbol{\delta}$:

$$S \triangleq \begin{bmatrix} \mathbf{0}_h & I_{h-1 \times h-1} \\ & \mathbf{0}^\top_{(h-d-1)\times 1} \quad \overleftarrow{\boldsymbol{\theta}}^\top \end{bmatrix}, \; Z \triangleq \begin{bmatrix} 0_{(h-d)\times d} \\ I_{d\times d} \end{bmatrix} \tag{1}$$

where $\overleftarrow{\boldsymbol{\theta}}^\top \triangleq [\theta_d, \dots, \theta_1]$. We note that $\boldsymbol{x}_k = S\boldsymbol{x}_{k-1}$. In particular, $\boldsymbol{x}_{h-1} = S^h \boldsymbol{x}_{-1}$. Therefore the augmented forecast is $\hat{\boldsymbol{t}}_{\boldsymbol{\delta}} = S^h (\boldsymbol{x}_{-1} + Z\boldsymbol{\delta})$.

We may now formally state Alice's optimal attack:

$$\boldsymbol{\delta}^* \triangleq \arg\min_{\boldsymbol{\delta}} ||\hat{\boldsymbol{t}}_{\boldsymbol{\delta}} - \boldsymbol{t}||_\ell + ||\boldsymbol{\delta}||_e \tag{2}$$

$$= ||S^h (\boldsymbol{x}_{-1} + Z\boldsymbol{\delta}) - \boldsymbol{t}||_\ell + ||\boldsymbol{\delta}||_e \tag{3}$$

$$\text{s.t. } ||\boldsymbol{\delta}||_c \preceq \beta \tag{4}$$

where $||\cdot||_l$ defines Alice's loss function, $||\cdot||_e$ defines her effort function, and $||\cdot||_c$ (along with her budget β) defines the feasible region of her attacks.

Described above is called an *attractive* attack (attract the prediction to a specific target $\boldsymbol{t}$). Approaches similar to those described here can be used to determine the optimal *repulsive* attack, where Alice tries to drive Bob's forecast as far as possible from his original forecast (or from a reference point specified by Alice). For reasons of brevity we do not discuss repulsive attacks here.

Attack Achievability

We note that Alice's target $\boldsymbol{t}$ may not be *achievable* given Bob's autoregressive model. In the One Week example, the target $\boldsymbol{t} = [1, 10, 10]$ is unachievable given Bob's model; $0.5 \times 10 - 0.6 \times 1 = 4.4 \neq 10$. For a fixed horizon h, we let $\mathcal{A}^{(\boldsymbol{\theta})}$ denote the set of all acheivable targets:

$$\mathcal{A}^{(\boldsymbol{\theta})} \triangleq \{\boldsymbol{t} \in \mathbb{R}^h \mid \exists\, \boldsymbol{x} = x_{-d}, \dots, x_{h-1} \text{ where } x_i = \sum_{j=1}^{d} \theta_j x_{i-j} \text{ and } t_i = x_i \text{ for } i = 0, \dots, h-1\} \tag{5}$$

We denote the closest achievable point to Alice's target $\boldsymbol{t}$ as:

$$\boldsymbol{t}^* \triangleq \arg\min_{\hat{\boldsymbol{t}} \in \mathcal{A}^{(\boldsymbol{\theta})}} ||\hat{\boldsymbol{t}} - \boldsymbol{t}||_\ell \tag{6}$$

We call $||\boldsymbol{t}^* - \boldsymbol{t}||_\ell$ Alice's *disappointment*, similar to the notion of approximation error in learning theory (Mohri, Rostamizadeh, and Talwalkar 2012).

We now describe several real-world attack scenarios. We formulate each with our framework via instantiations of $||\cdot||_\ell$, $||\cdot||_e$ and $||\cdot||_c$ for which the resulting optimization problem is convex and solvable with standard methods.

A Quadratic Loss Function for Alice

We first examine the case that Alice has hard constraints, and all available attacks have equal effort ($||\cdot||_e \triangleq 0$). We consider the Mahalanobis norm for Alice's loss: $||\boldsymbol{v}||_\ell \triangleq ||\boldsymbol{v}||^2_W = \boldsymbol{v}^\top W \boldsymbol{v}$, where W is a symmetric, positive semi-definite matrix defining Alice's priorities over different upcoming timesteps. If, for example, $W = \sqrt{\text{diag}(\boldsymbol{w})}$, where $\boldsymbol{w} \succeq 0$, then w_i is the relative weight of how much Alice cares about $\hat{t}_i - t_i$. In the One Week example, $W = \text{diag}([1, 1, \sqrt{2}]^\top)$ would indicate that Alice cares twice as much about Friday's value as she does about the other days. Note that if $d > h$, we may inflate the horizon as described previously, and append $d - h$ all-zero rows and columns to W to encode that Alice does not care about the inflated portion[1].

We consider this norm for two reasons. First, it is flexible enough to capture many real-world scenarios, as weighted mean squared error is a common evaluation for forecasting performance. Second is its mathematical convenience. Under the settings with $||\cdot||_\ell = ||\cdot||^2_W$, $||\cdot||_e = 0$, we may write Alice's goal as a standard quadratic minimization problem. She seeks to minimize:

$$F(\boldsymbol{\delta}) \triangleq ||S^h (\boldsymbol{x}_{-1} + Z\boldsymbol{\delta}) - \boldsymbol{t}||^2_W \tag{7}$$

$$= \frac{1}{2}\boldsymbol{\delta}^\top Q \boldsymbol{\delta} + \boldsymbol{c}^\top \boldsymbol{\delta} \tag{8}$$

where:

$$Q \triangleq \left(S^h Z\right)^\top W \left(S^h Z\right) \tag{9}$$

$$\boldsymbol{c} \triangleq Z^\top \left(S^h\right)^\top W^\top S^h \boldsymbol{x}_{-1} - Z^\top \left(S^h\right)^\top W^\top \boldsymbol{t} \tag{10}$$

[1] To accommodate an offset term α we append an all-0 row S, followed by appending the column $[0, \dots, 0, \alpha, 1]^\top$. We then append a 1 to $\boldsymbol{t}$, an all-0 row to Z, and an all-0 row and column to W.

This can be seen by a simple calculation:

$$\begin{aligned} F(\boldsymbol{\delta}) =& \left(S^h\left(\boldsymbol{x}_{-1}+Z\boldsymbol{\delta}\right)-\boldsymbol{t}\right)^\top W\left(S^h\left(\boldsymbol{x}_{-1}+Z\boldsymbol{\delta}\right)-\boldsymbol{t}\right) \\ =& 2\left(S^h\boldsymbol{x}_{-1}\right)^\top W\left(S^hZ\boldsymbol{\delta}\right) \\ &+\left(S^hZ\boldsymbol{\delta}\right)^\top W\left(S^h\boldsymbol{x}_{-1}\right)-2\,\boldsymbol{t}^\top WS^hZ\boldsymbol{\delta} \\ &+\underbrace{\left(S^h\boldsymbol{x}_{-1}\right)^\top W\left(S^h\boldsymbol{x}_{-1}\right)}_{\text{Independent of }\boldsymbol{\delta}} \\ &-\underbrace{2\,\boldsymbol{t}^\top WS^h\boldsymbol{x}_{-1}+\boldsymbol{t}^\top W\boldsymbol{t}}_{\text{Independent of }\boldsymbol{\delta}} \end{aligned} \tag{11}$$

Alice's Constraints

Given no constraints or efforts to her attack, Alice can always drive Bob's forecast to $\boldsymbol{t}^*$ by setting $\delta_i = x_i^* - x_i$ where $\boldsymbol{x}^*$ is the $\boldsymbol{x}$ which achieves $\boldsymbol{t}^*$ in (5). In reality, Alice often has constraints in poisoning the data. We first examine the case of hard constraints, where Alice has strict bounds on what attack vectors are available. We then consider the "soft constraint" case, where Alice must pay a cost based on the magnitude of her attack.

For the hard constraints presented below, we phrase the resulting minimization problem as a quadratic program in standard form:

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} \quad \frac{1}{2}\boldsymbol{\delta}^\top Q\boldsymbol{\delta}+\boldsymbol{c}^\top\boldsymbol{\delta} \tag{12}$$

$$\text{s.t.} \quad G\boldsymbol{\delta}\preceq\boldsymbol{h} \tag{13}$$

Per-Step Budget: Consider the case where Bob is forecasting the number of visits to a web site for use in determining ad placement. For concreteness, assume Bob uses total daily page visits in his model (one time step is one day). Alice is a hacker, controlling a botnet (a collection of computers on the Internet which she can take limited control of). She uses her botnet to create fraudulent traffic, and aims to affect Bob's predictions of web traffic.

Alice acts under two main constraints: (i) on any given day, Alice's botnet may only create up to β false page visits (based on the number of computers she has infected), and (ii) Alice may not remove legitimate traffic. This yields the constraints $0 \le \delta_i \le \beta$, or

$$||\boldsymbol{\delta}||_c = \begin{cases} ||\boldsymbol{\delta}||_\infty & \text{if } \boldsymbol{\delta}\succeq 0 \\ \infty & \text{otherwise} \end{cases} \tag{14}$$

in (4). We encode such constraints into (13) via

$$G=\begin{bmatrix} I_d \\ -I_d \end{bmatrix}, \boldsymbol{h}=\begin{bmatrix} \mathbf{1}_{d\times 1}\beta \\ \mathbf{0}_{d\times 1} \end{bmatrix} \tag{15}$$

where I_d is the $d \times d$ identity matrix, and $\mathbf{0}$ is the all-zero vector.

Total Budget: In futures markets, actors bet on the future price of a commodity. With some details ignored, we illustrate the mechanics of futures markets with the following example. Charlotte and David meet on e.g., September 18 to discuss the price of natural gas. They settle on a price x (dollars per 100 cubic feet (Ccf)), and a quantity (q Ccf). One week later, on September 25, the market price of natural gas will be y dollars per Ccf (however, y is unknown on the 3rd). They both sign a contract saying that on September 25, Charlotte will pay David $qy - qx$ dollars, where David pays Charlotte if $qy - qx$ is negative.

In this example, Charlotte benefits from the future price y being low, and David benefits from y being high – they both have an incentive to accurately forecast gas prices. Furthermore, they both have an incentive for the other party's forecasts to be wrong. We consider the case where one market participant, Alice, aims to alter the forecasts of others so as to improve the deals she makes with them. Alice may take the roll of either Charlotte or David in the natural gas example. She corrupts the past values (e.g., amounts of fuel stored or apparent demand) fed into the other party's forecasting model.

If Alice is bound by a per-step budget, she may find her optimal attack as she did in the botnet example above. We consider the case where Alice has some store of β of the commodity, and may use it to artificially inflate (but not deflate) past values. This induces the constraint $\sum_i \delta_i \le \beta$ and the non-negativity constraints $\boldsymbol{\delta} \succeq \mathbf{0}$. We encode such constraints into (13) via

$$G=\begin{bmatrix} \mathbf{1}_{d\times 1}^\top \\ -I_d \end{bmatrix}, \boldsymbol{h}=\begin{bmatrix} \boldsymbol{\beta} \\ \mathbf{0}_{d\times 1} \end{bmatrix} \tag{16}$$

where $\mathbf{1}$ is the all-ones vector.

Soft Constraints: We now consider the following example, where Alice is a participant on the wholesale electricity market. Alice runs a paper mill, and Bob forecasts electricity usage. Alice can artificially raise (by turning on heaters[2]) or lower (by reducing paper production) her daily electricity usage, but must pay to do so (either for the electricity, or from the lost revenue). For simplicity we assume lowering usage by 1KWh costs as much as raising by 1KWh, but note that Alice's cost is non-linear. While in reality Alice's cost function may be quite complicated, we may approximate it as quadratic.

We generalize this to the non-linear effort function $||\cdot||_e \triangleq \lambda||\boldsymbol{\delta}||_{\boldsymbol{H}}^2 = \boldsymbol{\delta}^\top H\boldsymbol{\delta}$, where λ is a fixed scalar determining the weight of Alices cost, and H is a positive semi-definite matrix defining Alice's costs (similar to how W defines the weights on forecasts). This yields the unconstrained minimization problem:

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}} \frac{1}{2}\boldsymbol{\delta}^\top Q\boldsymbol{\delta}+\boldsymbol{c}^\top\boldsymbol{\delta}+\lambda\boldsymbol{\delta}^\top H\boldsymbol{\delta} \tag{17}$$

with optimal solution

$$\boldsymbol{\delta} = -\left(Q+2\lambda H\right)^{-1}\boldsymbol{c} \tag{18}$$

$$= -\left(\left(\left(S^hZ\right)^\top W\left(S^hZ\right)\right)+2\lambda H\right)^{-1} \left(Z^\top\left(S^h\right)^\top W^\top S^h\boldsymbol{x}_{-1}-Z^\top\left(S^h\right)^\top W^\top\boldsymbol{t}\right) \tag{19}$$

[2] http://www.nytimes.com/2012/09/24/technology/data-centers-in-rural-washington-state-gobble-power.html

Experiments

We investigate the capabilities of an attacker through empirical experiments first on synthetic, and then on real world data.

We compare Alice's optimal attack against two baselines: Random and Greedy. Random selects 100 attacks $\boldsymbol{\delta}_1, \ldots, \boldsymbol{\delta}_{100}$ and evaluates F on each, selecting the minimum. We note that as the number of attacks Random takes increases, its performance approaches optimal. Greedy first minimizes F as a function of only δ_{-1}. That is, Greedy fixes $x_{-d}, \ldots, x_{-2}$ and attacks only x_{-1}. We note that this is a mathematically simpler task, as the objective function is now univariate. After finding the optimal δ_{-1}, Greedy continues, attacking only x_{-2} while fixing $x_{-d}, \ldots, x_{-3}, x_{-1} + \delta_{-1}$, and so on.

Greedy selects this particular ordering due to the following observation. The value δ_{-d} directly affects only the forecast for time 0 (it indirectly affects the forecasts for future times). The value δ_{-1}, however, directly affects forecasts for time 0 through $d-1$. Greedy begins by poisoning values with the most direct effect, proceeding to those with less direct effect.

In practice, when learning a time series model, it is common to restrict the set of models considered to only those which are *weakly stationary*. A model is weakly stationary if its first moment and covariance do not vary in time, which for AR(d) processes is equivalent to the roots of the polynomial $z^d - \sum_{i=1}^{d} \theta_i z^{d-i}$ all lying within the unit circle on the complex plane. All models used in experiments on synthetic data, below, are weakly stationary. We use such models due to their prevalence in practice, but note that the methods do not assume stationarity, and work for general models.

Experiments were conducted using the optimization package cvxopt v1.1.7 (Dahl and Vandenberghe 2006), figures were made with Matplotlib v1.4.3 (Hunter 2007). Due to space limitations we describe the results of experiments only in the hard-constraint settings.

Synthetic Data

We empirically investigate the following question: How effective is Alice's attack as a function of her target and Bob's model? We begin by creating four different models $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(4)}$, and evaluating an attacker's effectiveness for various targets. Specifically, let $d = 5$ and draw $\boldsymbol{\theta}^{(i)}$ from a unit spherical Gaussian $\mathcal{N}(\mathbf{0}, I)$, using rejection sampling to ensure the models are weakly stationary [3]. For each model, we let $h = 7$ and select $n = 1,000$ target vectors by sampling each value *iid* from $\mathcal{N}(0, 1)$. We then generate initial values $x_{-d}, \ldots, x_{-1}$ by beginning with Gaussian white noise and drawing values from the random process $x_t = \sum_{i=1}^{d} \theta_i x_{t-i} + \omega_i$ where $\omega_i \overset{\text{iid}}{\sim} N(0, 1)$[4]. We simulate the per-step budget setting and constrain the attacker by: $-\beta \leq \delta_i \leq \beta = 1/3$. We let $W = I$, yielding $||\cdot||_W^2 \triangleq ||\cdot||_2^2$.

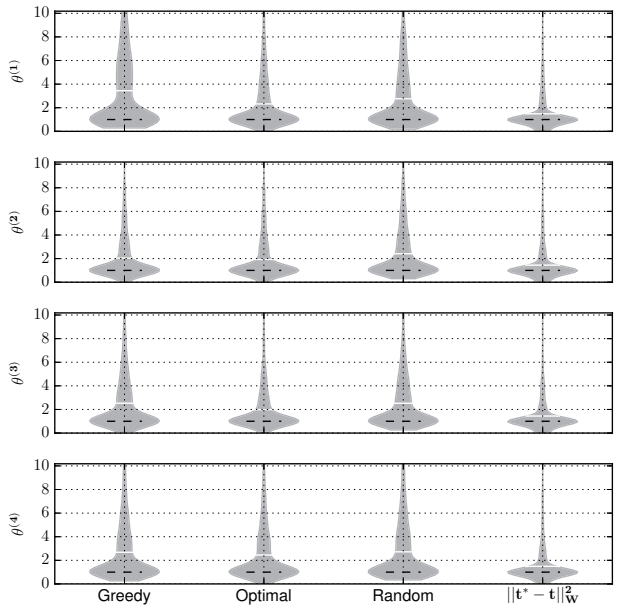

Figure 1: 1,000 Attacks on Four Models. Each row corresponds to one model, for which we consider 1,000 attack target vectors. In the first three columns, we show the distribution of difference between the attack's effectiveness and Alice's disappointment ($||\hat{\boldsymbol{t}} - \boldsymbol{t}^*||_W^2$)) (lower is better) for each attacker. White (solid) lines denote the mean value , and black (dashed) lines show the median. The right most column shows the value of the best, achievable target.

For each attacker, we report the empirical distribution of their loss compared to the best achievable target: $||\hat{\boldsymbol{t}}_{\boldsymbol{\delta}} - \boldsymbol{t}||_W^2 - ||\boldsymbol{t}^* - \boldsymbol{t}||_W^2$. In addition, we report the distribution of $||\boldsymbol{t}^* - \boldsymbol{t}||_W^2$. Figure (1) shows the results in violin plots.

For display purposes, we have cut off the y-axis at the 95$^{\text{th}}$ percentile – the additional remaining values create long tails. We first notice that across the four models, $||\boldsymbol{t}^* - \boldsymbol{t}||_W^2$ shows a consistent distribution. This is as expected, as *a priori* no model should lead to more achievable targets than any other. Interestingly, the shape of the other distributions (in particular Greedy and Optimal) vary from model to model. We then note that the performance between greedy and random is mixed – Greedy tends to win under $\boldsymbol{\theta}^{(2)}$, Random under $\boldsymbol{\theta}^{(1)}$, and it is unclear under the other models. While optimal attack always outperforms the baselines (by nature of being optimal), the amount by which it does so varies by model. In fact, it is unclear that Alice obtains significant gains by using the optimal attack over the baselines.

To further investigate this, we perform an additional experiment, this time varying over many different models. We perform the following process for $i = 1, \ldots, 1000$: (1) Draw $\boldsymbol{\theta}^{(i)}$ from $\mathcal{N}(\mathbf{0}, I)$. (2) Generate $x_{-d}, \ldots, x_{-1}$ as before. (3) Draw $\boldsymbol{t}$ from $\mathcal{N}(\mathbf{0}, I)$. (4) Evaluate the different attack meth-

[3] Due to the many random vectors and matrices created from them, numerical precision is a concern. To insure accuracy, we bound the condition number of matrices in (3), removing the occasional numerically unstable result.

[4] We use the values after a 500 step burn-in period.

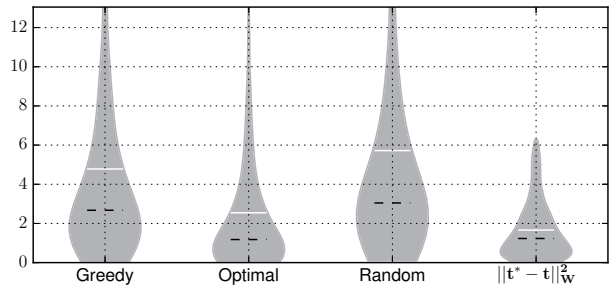

Figure 2: 1,0000 Attacks on 1,000 Models. Each column shows the distribution of $F(\boldsymbol{\delta})$, similar to Figure (1).

ods. (5) Consider the bottom 95% so as to avoid the long tail.

This is similar to the previous experiment, but expanding in breadth instead of depth in terms of models. That is, in this experiment, we are using 1,000 different models for Bob, and testing one attack on each, whereas previously we used 4 models and tested 1,000 attacks against each. Figure (2) shows the results as a violin plot, where we have left a large portion of the tail depicted for illustrative purposes. Here we observe that Greedy does outperform Random in terms of mean (the solid white line) and median (dashed black line), but only slightly. We remind the reader that if Random were to try more attacks, it would approach optimal in the limit. By expanding the breadth of the models in this way, we also see that both Random and Greedy are significantly inferior to the optimal attack.

Real World Example: Futures Markets

We next illustrate the effects of Alice's attack in a more detailed evaluation of real-world data. We consider two attackers, aiming to change the forecast market value of natural gas. The first attacker, Alice High, aims to increase the forecast, whereas the second attacker, Alice Low, aims to lower the forecast.

We obtained[5] historical US natural gas prices for 2014. After centering the series, we learned an order $d = 5$ AR model via Yule-Walker estimation (Box, Jenkins, and Reinsel 2011). We let $h = 10$, and assume the attackers are aiming for two spikes/dips with value ± 0.15, one on the 5$^{\text{th}}$ day and one on the 10$^{\text{th}}$, with relatively low (in absolute value) values elsewhere. We encode this as $\boldsymbol{t} = [0, 0, 0, 0, 0.15, 0, 0, 0, 0, 0.15]^\top$ for Alice High, and its negative for Alice Low. To encode the fact that the attackers care more about the spike/dip days than the others, we let W be the diagonal matrix where $w_{i,i} = 0.1$ if $i \notin \{0, 9\}$, and 1.0 otherwise. We constrain both Alices with a modest total budget of $\beta = 0.1$, and impose non-negativity constraints on $\boldsymbol{\delta}$.

Figure (3) shows the results. The most striking feature is that Alice High's attack is hugely effective for both spikes, and Alice Low's is for the second dip, albeit less so for the first dip. We further note that Alice Low's attack differs considerably more from the corresponding $\boldsymbol{t}^*$ than Alice High's. While Alice's attacks do not closely follow the target series, recall that Alice has heavily down-weighted the predictions for the non-spike/dip times. This setting also demonstrates how the effectiveness of attacks depends largely on the attacker's constraints. Note that Alice Low is unable to achieve the first dip, but the corresponding $\boldsymbol{t}^*$ does. This means that there is an attack leading to a better result, but Alice's constraints prevent her from selecting it. In contrast, Alice High's attack results in a $\hat{\boldsymbol{t}}$ very close to the best achievable forecast for her target.

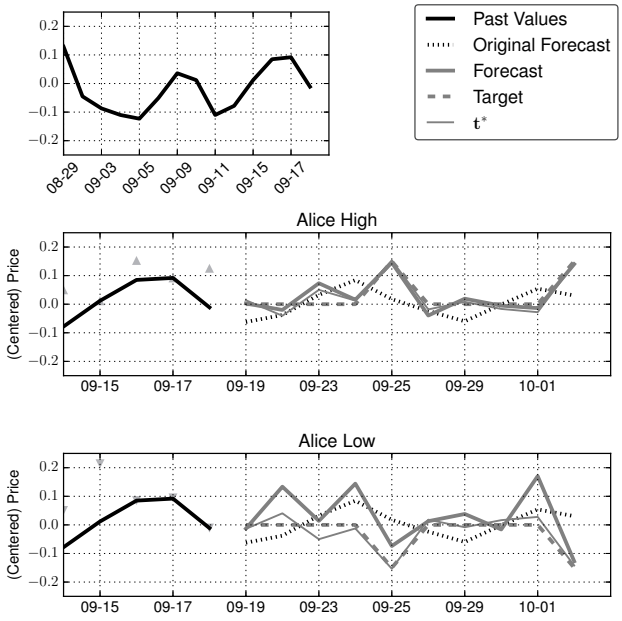

Figure 3: Natural Gas Settle Price. The top plot shows the US natural gas settle price. The bottom shows the results of two attackers. Alice High aims to cause two peaks, and Alice Low aims to cause two dips. Each has a total budget of $\beta = 0.1$, and non-negativity constraints. Downward triangles denote Alice Low's alterations, Upward triangles denote Alice High's.

Defense Analysis

A future goal of this line of research is to create forecasting models robust to the attacks of intelligent adversaries. To this end, we now explore how Alice's effectiveness varies with her constraints. This is motivated by the fact that in the real world, Bob may have the ability to further restrict Alice by e.g., imposing harsher (for example, legal) punishments should she be caught or by more diligently inspecting her behavior. We return to the natural gas example, but with a variety of different targets. So as to allow Greedy to participate, we now constrain the attacker via a per-step budget. We do not impose non-negativity constraints. We examine Alice (performing an optimal attack), Greedy, and Random's effectiveness as a function of β.

[5]Data is publicly available from http://www.quandl.com, Quandl Code CME/NGF2015.

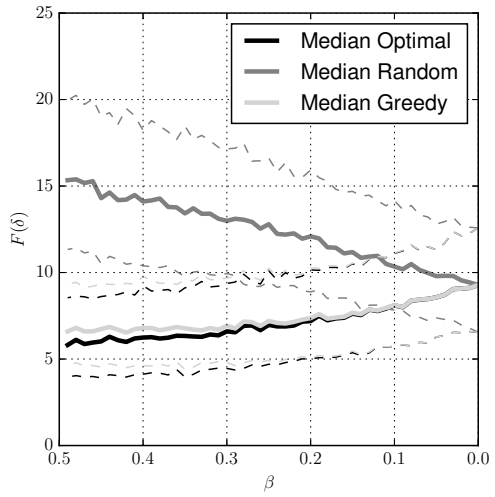

Figure 4: Effectiveness as a Function of Budget. We evaluate the effectiveness of various attackers against the natural gas prices presented in Figure (3). Dashed lines show the 25$^{\text{th}}$ and 75$^{\text{th}}$ percentiles over 1,000 trails (each with a different target).

Figure 4 shows the results. When the attacker is highly unconstrained, Greedy is suboptimal, but still quite close to optimal. We posit that with relaxed constraints, Greedy has more potential for causing harm when selecting early values which cannot be healed by later alterations. As Alice becomes more constrained Greedy approaches the optimal attack. We further note that Random's performance increases, relative to Optimal, as the constrains become more restrictive. This is expected, as Random has a better chance of finding a high quality attack when the sample space is small. Finally, all attackers converge as $\beta \to 0$; when $\beta = 0$, the only available option is $\boldsymbol{\delta} = \mathbf{0}$.

Related Work

The security ramifications of using machine learning models have long been studied (Barreno et al. 2006; 2010; Dalvi et al. 2004; Laskov and Kloft 2009; Tan, Killourhy, and Maxion 2002). Much of the work has focused on the common settings of regression and classification, although recent work has also looked at the security of LDA (Mei and Zhu 2015a). Attacks come in two forms: poisoning the training data of a learner, and attacking an already-learned model.

The former case has been studied in depth for a variety of learning algorithms. Support vector machines, in particular, have received a great deal of attention (Biggio and Laskov 2012; Biggio et al. 2014; Xiao et al. 2014; Xiao, Xiao, and Eckert 2012). A somewhat orthogonal line of research has posed the problem of learning in the presence of adversaries in a game theoretic context (Liu and Chawla 2009; Brückner, Kanzow, and Scheffer 2012; Dalvi et al. 2004; Brückner and Scheffer 2009; 2011). In addition, adversarial learning has been put into the framework of machine teaching (Zhu 2015). Using this, a general framework for data poisoning attacks against learners was recently proposed (Mei and Zhu 2015b), which mirrors our own.

The latter case, of attacking a fixed model, has also received considerable attention. One canonical example is in the realm of spam detection (c.f., (Nelson et al. 2009) and (Lowd and Meek 2005)). Here, an adversary has a spam message she would like to slip past a spam filter. She performs an *evasion attack* (Biggio et al. 2013) by augmenting her original message so as to be classified as benign by the already learned model. The attack motivation in our setting is similar, where the past values are like Alice's original message, and she aims to alter them so as to change Bob's prediction about it (in our case, this prediction is a forecast of the future rather than a binary label).

Using machine learning methods for time series forecasting has also gained growing interest, as such methods often outperform traditional ARIMA models from the statistics literature(Ahmed et al. 2010). See (Bontempi, Taieb, and Le Borgne 2013) for further details and a review of machine learning methods for forecasting.

Conclusions and Discussion

We have presented a general mathematical framework for formulating an adversary's data poisoning attack against a fixed (presumably learned) linear autoregressive model. The framework is general enough to accommodate various targets, costs, and constraints of the attacker. We have provided several examples of real world settings for which the resulting optimization problem is convex and solvable. We then explored the effectiveness of optimal attacks, as well as two baselines, with empirical experiments on synthetic data as well as real world data from US natural gas futures markets. We examined Alice's effectiveness as a function of her target and constraints, as well as of Bob's model.

From the empirical investigations, both baselines are considerably outperformed by the optimal attack. We observed that the attacks (as well as the random and greedy baselines) have a long tail, but often are very effective. The relative effectiveness of the Greedy and Random baselines vary by model, but in general the Greedy baseline outperforms Random, although Random will of course approach optimal as the number of trials increases. As future work we plan to perform further empirical investigation to determine (i) what is Alice's effectiveness as a function of the order d of the model and the forecast horizon h, (ii) how does Alice's effectiveness vary as a function of the *type* of constraint (e.g. per-step vs total budget), and (iii) what aspects of Bob's model effectively hinder Alice's abilities. A primary goal of this line of research is to determine methods by which Bob can learn a model which is both accurate and robust to attacks.

We have described several settings where Alice's optimal attack is phrased in a standard quadratic program. In future work, we will explore other settings which still result in solvable systems. One example is when Alice has a hard constraint on the squared 2-norm of her attack ($||\cdot||_c = ||\cdot||_2^2$). This yields a quadratically constrained quadratic pro-

gram (QCQP), and in interest of brevity we did not explore this setting here.

Additional future work includes investigating a more limited attacker and more complicated forecaster. In some settings, Alice must select her values "online". That is, Alice selects δ_i on day $-i$ without knowing future values, and she may not later change δ_i. We also seek to develop similar methods of attack (and ultimately defense) against nonlinear models, as well as direct and multiple-output forecasting strategies.

Acknowledgments

This material is based upon work supported by the DHS grant BAA 11-01 and AFRL grant FA8750- 12-2-0328. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the DHS or AFRL.

Support for this research was provided in part by NSF grant IIS 0953219 and by the University of Wisconsin-Madison Graduate School with funding from the Wisconsin Alumni Research Foundation.

References

Ahmed, N. K.; Atiya, A. F.; Gayar, N. E.; and El-Shishiny, H. 2010. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews* 29(5-6):594–621.

Barreno, M.; Nelson, B.; Sears, R.; Joseph, A. D.; and Tygar, J. D. 2006. Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, 16–25. ACM.

Barreno, M.; Nelson, B.; Joseph, A. D.; and Tygar, J. 2010. The security of machine learning. *Machine Learning* 81(2):121–148.

Biggio, B., and Laskov, P. 2012. Poisoning attacks against support vector machines. In *In International Conference on Machine Learning (ICML*.

Biggio, B.; Corona, I.; Maiorca, D.; Nelson, B.; Šrndić, N.; Laskov, P.; Giacinto, G.; and Roli, F. 2013. Evasion attacks against machine learning at test time. In *Machine Learning and Knowledge Discovery in Databases*. Springer. 387–402.

Biggio, B.; Corona, I.; Nelson, B.; Rubinstein, B. I.; Maiorca, D.; Fumera, G.; Giacinto, G.; and Roli, F. 2014. Security evaluation of support vector machines in adversarial environments. In *Support Vector Machines Applications*. Springer. 105–153.

Bontempi, G.; Taieb, S. B.; and Le Borgne, Y.-A. 2013. Machine learning strategies for time series forecasting. In *Business Intelligence*. Springer. 62–77.

Box, G. E.; Jenkins, G. M.; and Reinsel, G. C. 2011. *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons.

Brückner, M., and Scheffer, T. 2009. Nash equilibria of static prediction games. In *Advances in neural information processing systems*, 171–179.

Brückner, M., and Scheffer, T. 2011. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 547–555. ACM.

Brückner, M.; Kanzow, C.; and Scheffer, T. 2012. Static prediction games for adversarial learning problems. *the Journal of Machine Learning Research* 13(1):2617–2654.

Dahl, J., and Vandenberghe, L. 2006. Cvxopt: A python package for convex optimization. In *Proc. eur. conf. op. res*.

Dalvi, N.; Domingos, P.; Sanghai, S.; Verma, D.; et al. 2004. Adversarial classification. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 99–108. ACM.

Hunter, J. D. 2007. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering* 9(3):90–95.

Laskov, P., and Kloft, M. 2009. A framework for quantitative security analysis of machine learning. In *Proceedings of the 2nd ACM workshop on Security and artificial intelligence*, 1–4. ACM.

Liu, W., and Chawla, S. 2009. A game theoretical model for adversarial learning. In *Data Mining Workshops, 2009. ICDMW'09. IEEE International Conference on*, 25–30. IEEE.

Lowd, D., and Meek, C. 2005. Good word attacks on statistical spam filters. In *CEAS*.

Mei, S., and Zhu, X. 2015a. The security of latent dirichlet allocation. In *18th Intl Conf. on Artificial Intell. and Statistics*.

Mei, S., and Zhu, X. 2015b. Using machine teaching to identify optimal training-set attacks on machine learners. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Mohri, M.; Rostamizadeh, A.; and Talwalkar, A. 2012. *Foundations of machine learning*. MIT press.

Nelson, B.; Barreno, M.; Chi, F. J.; Joseph, A. D.; Rubinstein, B. I.; Saini, U.; Sutton, C.; Tygar, J.; and Xia, K. 2009. Misleading learners: Co-opting your spam filter. In *Machine learning in cyber trust*. Springer. 17–51.

Tan, K. M.; Killourhy, K. S.; and Maxion, R. A. 2002. Undermining an anomaly-based intrusion detection system using common exploits. In *Recent Advances in Intrusion Detection*, 54–73. Springer.

Xiao, H.; Biggiob, B.; Nelson, B.; Xiao, H.; Eckerta, C.; and Rolib, F. 2014. Support vector machines under adversarial label contamination. *Neurocomputing Preprint*.

Xiao, H.; Xiao, H.; and Eckert, C. 2012. Adversarial label flips attack on support vector machines. In *ECAI*, 870–875.

Zhu, X. 2015. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *AAAI Conference on Artificial Intelligence (Senior Member Track)*.