Practical Ideas for Model Security

Private, fair models, explanations, model debugging, and common sense

Patrick Hall

March 8, 2019

Like many others I've known for sometime that machine learning models could pose security risks. However I lacked the vocabulary to reason through this very complex set of problems until just recently when Nicolas Papernot and Ben Lorica summarized and outlined some of the major issues practitioners should be worried about. When I went to dig into the literature I found lots of papers ... about adaptive deep learning and encryption. While adaptive deep learning on encrypted data is perhaps a goal to which to aspire in the future, I see several potential vulnerabilities and accompanying safe-guards that apply to the more common linear and tree-based models trained on static data that are used commonly today. Before we get started, it's important to say I am no security expert, but I have been following the areas of interpretability, explanations, fairness, model debugging and privacy in machine learning very closely and it's many of these techniques that can be applied to reduce the attack surfaces of predictive modeling systems. (I think some unification is needed across explanations, fairness, interpretability, model debugging, privacy and security, but I will jump off that bridge another day . . . or preferably . . . let someone else.) In hopes of furthering discussion by *actual* security experts and practitioners in the applied machine learning community, this post will put forward some not unreasonable scenario for attack vectors on a typical machine learning system at a typical organization and propose initial safe guards and solutions.

1 Insider Data Poisoning Attack

Data poisoning refers systematically changing the data a model is trained on in an effort to change the models outputs. At many companies an employee, contractor, or anyone else with access to training data could alter that training data so that whatever the commercial application of the model is, they could benefit from the model prediction. For example, by altering labels so that the model learns to award large loans or discounts to people like them. It's also possible that a malicious insider could use data poisoning to train a model to intentionally discriminate against a group of people.

Potential Solution(s):

- Score your models on your employees and contractors and look for anomalously beneficial outcomes.
- Residual analysis: Look for strange, prominent patterns in the residuals of predictions on employees or contractors.
- Private or fair models: Techniques exist such as learning fair representations (LFR) and private aggregation of teacher ensembles (PATE) that do not learn as much about specific individual traits to make predictions, and these models may be less susceptible to data poisoning attacks.
- Disparate impact analysis: Many banks already undertake disparate impact analysis for fair lending purposes to determine if a model is treating different types of people in an unfair manner. Many other organizations do not seem to care. Disparate impact analysis could potentially discover intentional discrimination. There are several great open source tools to conduct disparate impact analysis, such as aequitas, themis, and AIF360.

2 Insider Watermark Attack

Watermarking is a term borrowed from the deep learning security literature that refers to putting special pixels into an image to trigger a desired outcome from a model. It's possible to do the same with customer data. An employee, contractor, or anyone else with access to the production code that makes predictions using the trained machine learning model could change that code to recognize a strange or unlikely combination of input variable values that would trigger a desired prediction outcome. For instance a malicious insider could easily insert a payload into the production scoring code that recognizes the combination of age of 0 and years at an address of 99 to trigger some kind of outcome from the model that benefits themselves or their associates.

Potential Solution(s):

- Version control of production model code (so that changes to the production code are tracked and auditable).
- Data integrity constraints: Many databases don't allow for strange or unrealistic combinations of input variables and this could potentially thwart watermarking attacks. Applying data integrity constraints on live, incoming data could have the same benefits.
- Outlier detection in training data and in new data: Autoencoders are a common fraud detection model that can catch input data that is strange or unlike other input data, but in complex ways. Autoencoders could potentially catch any watermarks used to test the triggering mechanism in training data and could be run on new data in real-time to catch strange input data before it enters the model.

3 Inversion by Surrogate Model

Inversion basically refers to getting information out of a model (as opposed to putting information into the model). If an attacker can receive many predictions from a model API or other endpoint (website, app etc.), they can train a surrogate model, i.e. simulation, of the unknown model that they are seeking to attack. An attacker could conceivably train a surrogate or simulation model between the inputs they used to generate the received predictions and the received predictions themselves. Depending on the number of predictions they can receive, the surrogate model could become quite an accurate simulation of your model. Once the surrogate model is trained, then the attacker has a sandbox from which to plan impersonation or adversarial example attacks or the potential ability to start reconstructing aspects of the potentially sensitive training data. These types of surrogate models can also be trained using external data sources that can be somehow matched to predictions, as ProPublica famously did with the proprietary COMPAS recidivism model.

Solution(s):

- As a whitehat hacking exercise, train your own surrogate models between inputs and predictions of your production model and see ...
  - The accuracy bounds of different types of surrogate models. Try to understand the extent to which a surrogate model can really represent your production model.
  - What types of data trends can be learned from your surrogate model, like linear trends represented by linear model coefficients or the distribution of outcomes w.r.t. demographics by analyzing the number of individuals assigned to certain surrogate decision tree nodes.
  - What kind of rules can be learned from a surrogate decision tree, e.g. how to impersonate an individual that would receive a beneficial prediction.
- Restrict high numbers of rapid predictions from similar IP addresses; consider artificially slowing down prediction latency.
- Require additional authentication (e.g. 2FA) to receive a prediction.

4 Adversarial Attack for Evasion or Reward

A motivated attacker could conceivably learn, say by surrogate model inversion or by social engineering, how to game a model to receive a desired prediction outcome or avoid an undesirable prediction outcome. Carrying out an attack by engineering an input for such a purpose is referred to as an adversarial example attack. Adversarial example attacks could be used to get an outcome that an attacker wants, say a large loan or product discount, or an adversarial example attack could be used to avoid an undesirable prediction outcome, say a long prison sentence based on a high criminal risk score. Some people might call using adversarial examples to avoid an undesirable outcome from a model prediction "evasion".

Solution(s):

- Use sensitivity analysis and surrogate decision trees to understand what variable values (or combinations) can cause large swings in predictions and screen for these values or combinations of values when scoring new data.
- Activation analysis ...
- Benchmark models ...

5 Impersonation

A motivated attacker can learn, say again by surrogate model inversion or by social engineering, what type of input or individual receives a desired prediction outcome and then impersonate this input or individual to receive the desired prediction outcome.

Solution(s):

- Screen for duplicates, or very similar inputs, potentially in a reduced-dimensional space using autoencoders, multi-dimensional scaling (MDS), etc. while scoring new data.
- Require additional authentication (e.g. 2FA) to receive a prediction.
- Activation analysis ...

6 General Concerns

Several common machine learning usage patterns also have more general security concerns.

Blackboxes and Unnecessary Complexity: Although recent developments in interpretable models and model explanations have provided the opportunity to use accurate and also transparent nonlinear classifiers and regressors, many machine workflows are still centered around more traditional blackbox models. Such blackbox models are only one type of often unnecessary complexity in a typical commercial machine learning workflow. Other sources for undue complexity could be exotic feature engineering or large numbers of package dependencies. Such complexity can be problematic for at least two reasons.

- A dedicated, motivated attacker can, over time, learn more about your overly complex blackbox modeling system than you or your team knows about your own model. (Especially in today's overheated and turnover-prone data "science" market.) To do so, they can use many newly available model-agnostic explanation techniques and old-school sensitivity analysis, among many other more common hacking tools. This knowledge imbalance can potentially be exploited to conduct the attacks described in sections 1 – 5 or for other yet unknown types of attacks.
- Machine learning in the research and development environment is highly dependent on a diverse ecosystem of open source software packages. Some of these packages have many, many contributors and users. Some are highly specific and only meaningful to a small number of researchers or practitioners. It's

well understood that many packages are maintained by brilliant statisticians and machine learning researchers whose primary focus is mathematics or algorithms, not software engineering, and certainly not security. It's not uncommon for a machine learning pipeline to be dependent on dozens or even hundreds of external packages, anyone of which could be hacked to conceal an attack payload.

Distributed Systems and Models: For better or for worse, we are in the age of big data, and many organizations are now using distributed data processing and machine learning systems. Distributed computing can provide a broad attack surface for a malicious internal or external actor in the context of machine learning. Data could be poisoned on only one or a few worker nodes of a large distributed data storage or processing system. A backdoor for watermarking could be coded into just one model of a large ensemble. Instead of debugging one simple dataset or model, now practitioners must examine data or models distributed across large computing clusters.

6.1 General Solutions

Several older and newer general best practices can be deployed to lessen security vulnerabilities and to generally increase fairness, accountability, transparency, and trust in machine learning predictive systems.

Model Debugging for Security: The newer field of model debugging is focused on discovering errors in machine learning model mechanisms and predictions and remediating those errors. Debugging tools such a surrogate models, residual analysis, and sensitivity analysis can be used in whitehat exercises to understand your own vulnerabilities or for forensic exercises to find any potential attacks that may have occurred or be occurring.

Model Documentation and Explanation Techniques: Model documentation is a risk-mitigation strategy that has been used for decades in banking. It allows knowledge about complex modeling systems to be preserved and transferred as teams of model owners change over time. Model documentation has been traditionally applied to highly-transparent linear models, but with the advent of powerful, accurate explanatory tools such as tree SHAP and derivative-based local feature contributions for neural networks, now even pre-existing blackbox model workflows can be atleast somewhat explained, debugged, and documented. Documentation should obviously now include any known, remediated, or anticipated security vulnerabilites.

Model Monitoring and Management Explicitly for Security: Most serious practitioners understand most models are trained on static snapshots of reality represented by training data and their prediction accuracy degrades in real-time as present realities drift away from the information captured in the training data. Today most model monitoring is aimed at discovering this drift in input variable distributions that will eventually lead to accuracy decay. Model monitoring should now likely be designed to monitor for the attacks described in sections 1 – 5 and any other potential threats your whitehat model debugging exercises uncover. (While not directly related to security, my opinion is models should also be evaluated for disparate impact in real-time as well.) Along with model documentation all modeling artifacts, source code, and associated metadata need to be managed, versioned, and audited for security like the valuable commercial assets they are.

Interpretable, Fair, or Private Models: The techniques now exist, e.g. monotonic GBMs (M-GBM), scalable Bayesian rule lists (SBRL), eXplainable Neural Networks (XNN), that allow users to have both accuracy and interpretability. These accurate and interpretable models are easier to document and debug. Newer types of fair and private models, e.g. LFR, PATE, can also be trained to essentially care less about outward visible, demographic characteristics that can be easily observed, engineered into an adversarial example attack, or impersonated. When considering creating a new machine learning workflow in the future consider basing it around lower risk, private or fair models.

I care deeply about the science and practice of machine learning, and I'm now concerned a terrible machine learning hack, combined with credible concerns about privacy and discrimination, could increase burgeoning public skepticism about machine learning and AI. We should all remember there have been AI winters in the past, even the recent past. Security vulnerabilities, privacy violations, and algorithmic discrimination could all potentially combine to lead to decreased funding for machine learning research, draconian over-regulation of the field, and the next AI winter. Let's fix these important problems sooner rather than later.