A Game Theoretic Approach for Adversarial Machine Learning: Big Data Meets Cyber Security

Bowei Xi
Department of Statistics
Purdue University
xbw@purdue.edu

Joint Work with Wutao Wei (Purdue),
Murat Kantarcioglu (UT Dallas), Yan Zhou (UT Dallas)

Malicious Attacks on the Internet of Things (IoT)

Ultrasonic audio attacks, completely inaudible to people, can control speech recognition systems including Siri, Google Now, and Alexa. Inaudible commands can even manipulate the navigation system in an Audi automobile.

Visual attacks can cause traffic signs to be mis-classified.

Adversarial samples in cyber security may come from a very different distribution; adversarial images in computer vision are created by adding minor perturbations.

Adversarial Machine Learning (ML)

ML techniques are used to detect cyber security incidents.

Adversaries actively transform their objects to avoid detection.

They defeat traditional ML techniques that assume same properties for current and future datasets.

Need new ML techniques for adversarial environment.

Artificial Intelligence (AI) with Adversarial ML

AI needs adversarial ML capacities:

Game theoretic framework to model the interaction between attackers and defender (e.g. a learning system)

Adversarial supervised learning, unsupervised learning, and active learning algorithms

Break transferability of adversarial samples with randomness

Adversarial Stackelberg Game: Leader vs. Follower

Players take sequential actions and maximize their own utilities.

Defender being the follower is a m-leader-one-follower game.

Defender being the leader is a one-leader-m-follower game.

We study the players' equilibrium behavior for the two games – defender being the leader vs. defender being the follower. Wei, W., Xi, B., Kantarcioglu, M., Adversarial Clustering: A Grid Based Clustering Algorithm against Active Adversaries, submitted, arXiv:1804.04780

Adversarial Stackelberg Game: Leader vs. Follower

– Game when defender is the follower.

1. Given the joint attacks from the m adversaries, $T = (t_1, \cdots, t_m)$, solve for the defender's optimal strategy.

$$h_T = \text{argmax}_{\{h \in H\}} \quad \{D(t_1, \cdots, t_m, h)\}$$

2.With the solution above as the defender's optimal strategy h_T against joint attacks $T = (t_1, \cdots, t_m)$, solve for the optimal joint attacks T^e.

$$T^e = (t_1^e, ..., t_m^e) = \text{argmax}_{\{t_i \in S_i, \forall i\}} \sum_{i=1}^{m} A_i(t_i, h_T)$$

$(h^{T^e}, t_1^e, \cdots, t_m^e)$ is an equilibrium strategy for all players in the game.

– Game when Defender is the leader.

1. Given a leader's strategy h fixed, assume the m adversaries' (i.e., the followers') strategies are the attacks $T = (t_1, \cdots, t_m)$. For the i-th adversary, further assume all other adversaries' strategies are fixed, i.e., fixed t_j, $\forall j \neq i$. Solve the following optimization for t_i^h:

$$t_i^h = \text{argmax}_{\{t_i \in S_i\}} \quad \{A_i(t_i, h)\}$$

2. With the solution from above, $T^h = (t_1^h, \cdots, t_m^h)$ is the m adversaries' joint optimal attacks for a given defender strategy h, the defender solves another optimization problem.

$$h^e = \text{argmax}_{\{h \in H\}} \quad \{D(t_1^h, \cdots, t_m^h, h)\}$$

$(h^e, t_1^{h^e}, \cdots, t_m^{h^e})$ is an equilibrium strategy for all players in the game.

Adversarial Stackelberg Game: Leader vs. Follower

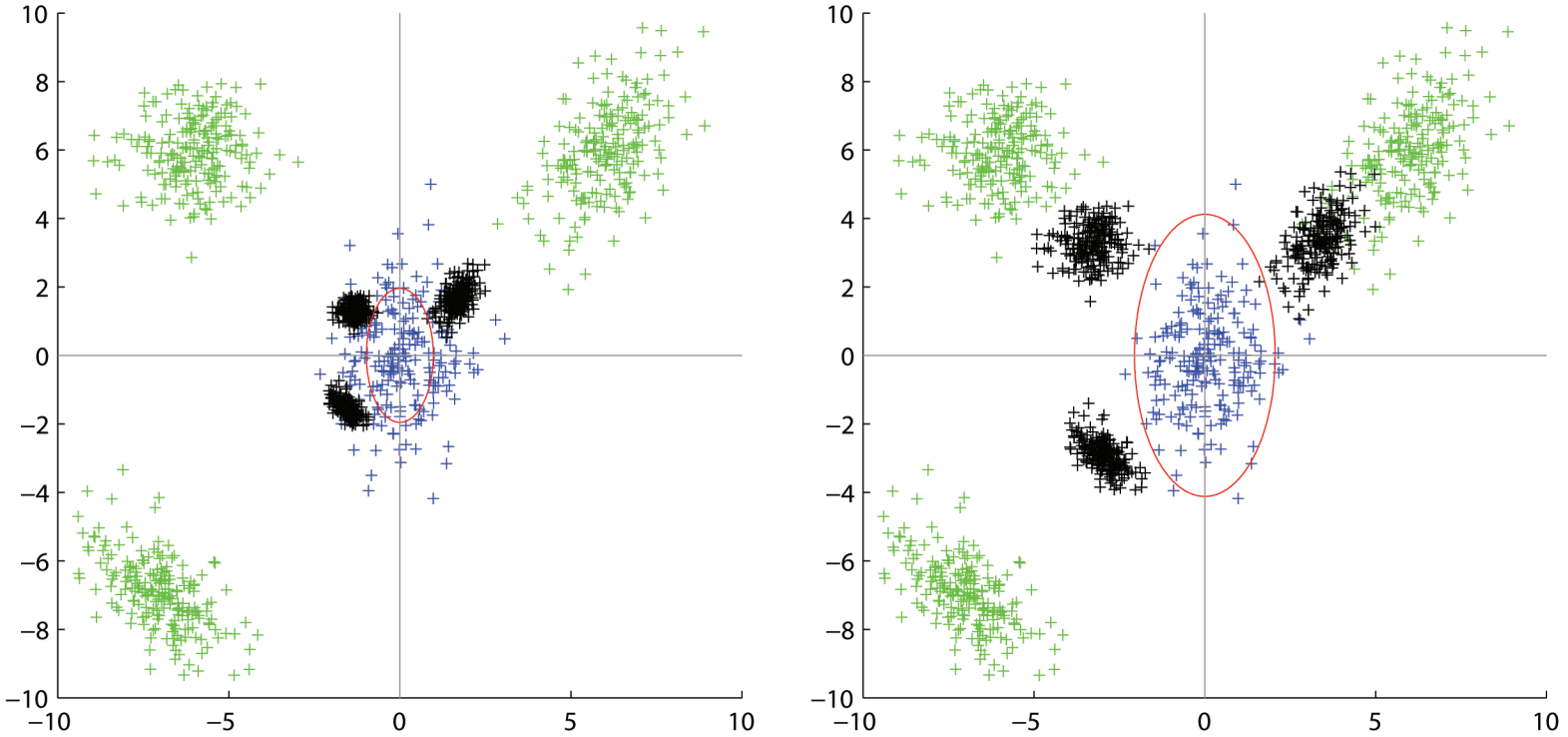
Attackers' strategies are to move their objects toward the center of the normal population by a factor t ($0 \leq t \leq 1$).

Defender's strategy is to draw a $\alpha-$ level defensive wall, comparable to a confidence region for multivariate Gaussian distribution.

Attackers' payoffs are the respective expected values of the utilities generated by the adversarial samples that avoid detection; defender's payoff is -1 times misclassification cost.

Defensive wall identifies high confidence normal region.

Equilibrium Behavior: Leader vs. Follower



Left is defender being the leader in the game – a smaller defensive wall and strong attacks; right is defender being the follower in the game – a larger defensive wall and mild attacks.

A Grid Adversarial Clustering Algorithm

Adversaries fill in the gap between previously well separated normal and abnormal clusters with small amount of attack objects.

Previous work largely focused on adversarial classification. It needs a reasonably large amount of carefully labeled data instances at high cost, time and human expertise.

Meanwhile, a large number of unlabeled instances can also be used to understand the adversaries' behavior.

A Grid Adversarial Clustering Algorithm

Our algorithm, ADClust, identifies normal and abnormal sub-clusters within a large mixed cluster along with the unlabeled overlapping regions, and outliers as potential anomalies.

A classifier with a well defined classification boundary is comparable to a point estimate, not accurate due to very few labeled objects.

Unlabeled overlapping areas identified by ADClust are comparable to confidence regions.

We identify the high confidence normal regions in mixed clusters through defensive walls.

A Grid Adversarial Clustering Algorithm

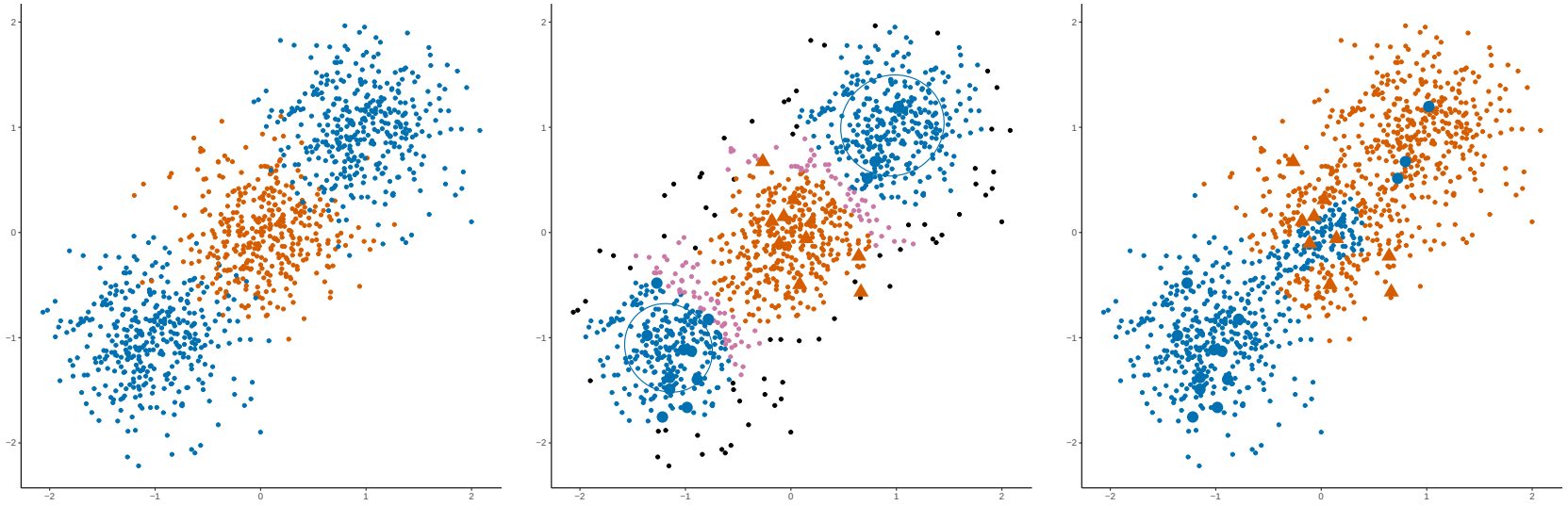
In the first pass, our algorithm (ADClust) groups the data points into normal sub-clusters, abnormal sub-clusters, unlabeled sub-clusters and unlabeled outliers. In a second pass, we do not use labels, and simply group the data points into large unlabeled clusters and identify unlabeled outliers.

Next step is to match the normal, abnormal, unlabeled smaller sub-clusters from the first pass with the unlabeled larger clusters from the second pass.

The last step is to play a conservative strategy, drawing $\alpha-$level defensive walls inside normal sub-clusters to identify high confidence normal areas.

A Grid Adversarial Clustering Algorithm

Compare with a semi-supervised learning algorithm, S4VM. $\alpha = 0.6$.



Left: actual clusters with blue for normal and orange for abnormal; Middle: our ADClust with purple for unlabeled; Right: S4VM. Solid circles (normal) and solid triangles (abnormal) are known correctly labeled objects.

A Grid Adversarial Clustering Algorithm

KDD Cup 1999 Network Intrusion Data: Around 40% are network intrusion instances. Average over 100 runs. In one run, 100 instances are randomly sampled with labels. 99.6% become unlabeled.

KDD data is highly mixed, yet we achieve on average nearly 90% pure normal rate inside the defensive walls.

Adversarial Active Learning

Active learning is another approach when there are very few labeled instances. It uses strategic sampling techniques.

Oracles assign labels to the most influential samples. Active learning requires less training data points to achieve accurate results.

In adversarial settings, malicious oracles selectively return incorrect labels. Also assume there are weak oracles that return noisy labels.

Adversarial Active Learning

1: Data is clustered using the labeled instances as seeds;

2: Oracle behavior profile is computed; Oracles are modeled as a mixture of three components—genuine, weak and malicious;

3: The most accurate group of oracles are used to label a selected sample;

4: The newly labeled sample is added to the labeled dataset. Go back to Step 1.

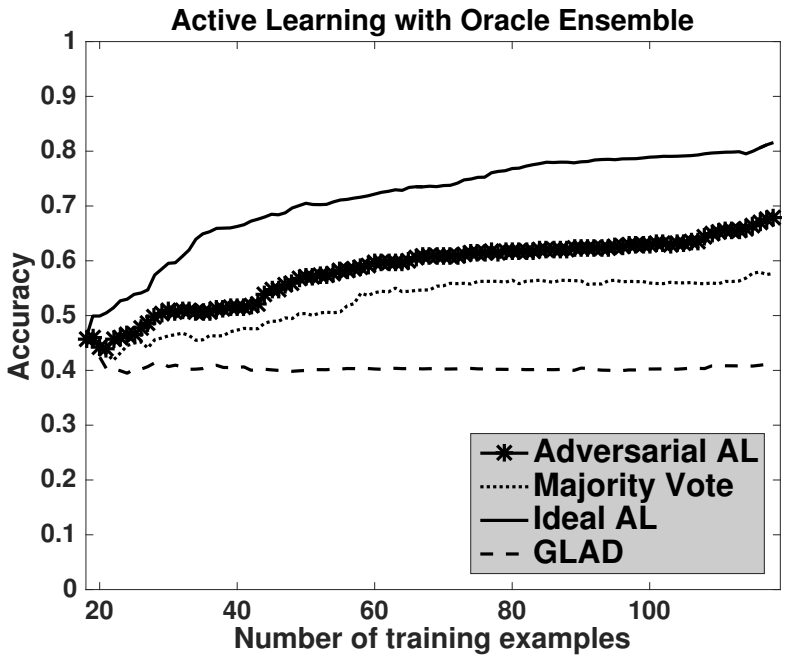
Adversarial Active Learning

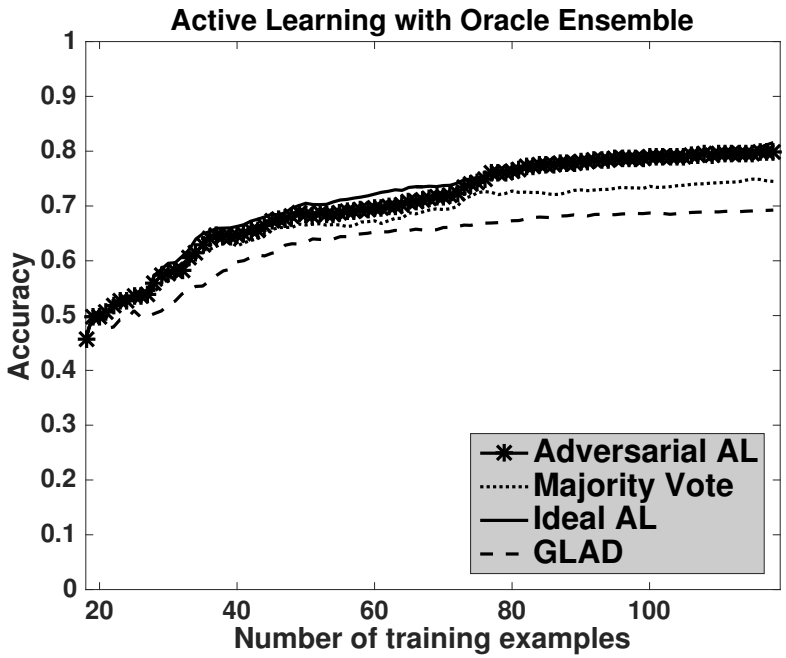
Webspam data is from LibSVM data repository. 350,000 instances, approximately 60% are webspam.

Compare our adversarial active learning technique to 1.) majority-vote; 2.) a crowd-sourcing technique—GLAD; and 3.) active learning technique without malicious and weak oracles.

We use support vector machine (SVM) as the underlying classifier in the active learning process.

Adversarial Active Learning





Left: 5 genuine oracles; Right: 10 genuine oracles.

Total 30 oracles. Rest are 50% weak and 50% malicious oracles.

Results averaged over 10 runs.

Robust results when the majority are malicious and weak oracles.

Adversarial SVM

AD-SVM solves a convex optimization problem where the constraints are tied to adversarial attack models.

Free-range attack: Adversary can move attack objects anywhere in the domain.

$$C_f(x_{.j}^{min} - x_{ij}) \leq \delta_{ij} \leq C_f(x_{.j}^{max} - x_{ij})$$

Targeted attack: Adversary can only move attack instances closer to a targeted value.

$$0 \leq (x_{ij}^t - x_{ij})\delta_{ij} \leq C_\xi \left(1 - C_\delta \frac{|x_{ij}^t - x_{ij}|}{|x_{ij}| + |x_{ij}^t|}\right) (x_{ij}^t - x_{ij})^2$$

Adversarial SVM

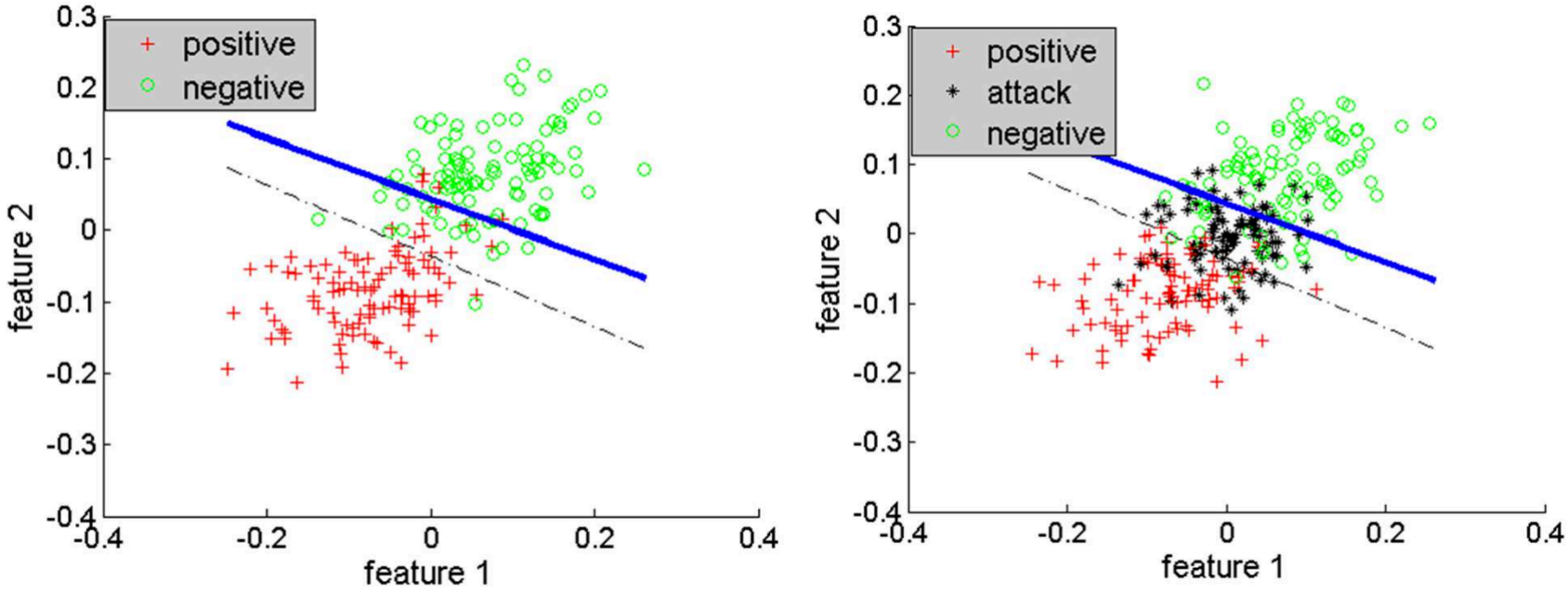
SVM risk minimization model: free-range attack

$$
\begin{array}{ll}
\text{argmin}_{w,b,\xi_i,t_i,u_i,v_i} & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\
s.t. & \xi_i \geq 0 \\
& \xi_i \geq 1 - y_i \cdot (w \cdot x_i + b) + t_i \\
& t_i \geq \sum_j C_f \left(v_{ij}(x_j^{max} - x_{ij}) - u_{ij}(x_j^{min} - x_{ij}) \right) \\
& u_i - v_i = \frac{1}{2}(1 + y_i)w \\
& u_i \succeq 0 \\
& v_i \succeq 0
\end{array}
$$

SVM risk minimization model: targeted attack

$$
\begin{array}{ll}
\text{argmin}_{w,b,\xi_i,t_i,u_i,v_i} & \frac{1}{2}||w||^2 + C\sum_i \xi_i \\
s.t. & \xi_i \geq 0 \\
& \xi_i \geq 1 - y_i \cdot (w \cdot x_i + b) + t_i \\
& t_i \geq \sum_j e_{ij} u_{ij} \\
& (-u_i + v_i) \circ (x_i^t - x_i) = \frac{1}{2}(1 + y_i)w \\
& u_i \succeq 0 \\
& v_i \succeq 0
\end{array}
$$

Adversarial SVM



The black dashed line is the standard SVM classification boundary, and the blue line is the AD-SVM classification boundary. It is a conservative strategy in anticipation of an attack.

DNN Models with a Randomness Factor

Attack a deep neural network (DNN) by adding minor perturbations to an image.

An example of the 3's

Many Proposed Defense and Detection Strategies

Enhance a DNN model by re-training on the adversarial samples – overfit to the adversarial samples & adversary can compute new adversarial samples against the re-trained DNN model.

Detection techniques are also proposed to differentiate adversarial samples from benign ones.

Unfortunately many are quickly shown to fail the latest attacks.

DNN Models with a Randomness Factor

Attacks are designed to break DNN models, such as the Carlini and Wagner's iterative L2 attack.

$$x^* = \text{argmin}_{x' \in D} \ (||x' - x||_2^2 + c \cdot l(x')),$$

Transferability of adversarial samples means that adversarial samples that break one learning model have the ability to break another model even if they belong to different model classes.

We show that creating DNN models with a randomness factor successfully break the transferability of adversarial samples.

DNN Models with a Randomness Factor

Assume the adversarial perturbation is bounded by $\epsilon > 0$. The transferability of an adversarial perturbation depends on the attack intensity.

We train a set of DNN models with stochastic gradient descent from several random initialization points.

To stop transferability, the distance between two decision boundaries must be sufficiently large.

We measure the spread of the weights on 3 datasets: MNIST, CIFAR, Traffic Sign.

DNN Models with a Randomness Factor

Weak Attack Scenario: Adversary has perfect knowledge of one randomly selected DNN to generate adversarial samples. (MNIST)

Baseline DNN and Ensemble-AdTrain (re-train a set of DNNs with adversarial samples) have accuracy 0.00 under attack.

Random-Model-10: randomly select 1 DNN to classify each query request. Accuracy $\mathbf{0}.863 \pm 0.289$

Ensemble-10: majority vote of 10 DNNs. Accuracy $\mathbf{0}.991 \pm 0.002$

Ensemble-AdTrain-Random: apply randomization to the re-trained DNNs. Accuracy $\mathbf{0}.874 \pm 0.291$

Random-Weight-10: randomly select one in a set of DNNs and randomly adding a small noise to the weights of the selected DNN to classify each query request. Accuracy $\mathbf{0}.779 \pm 0.078$

DNN Models with a Randomness Factor

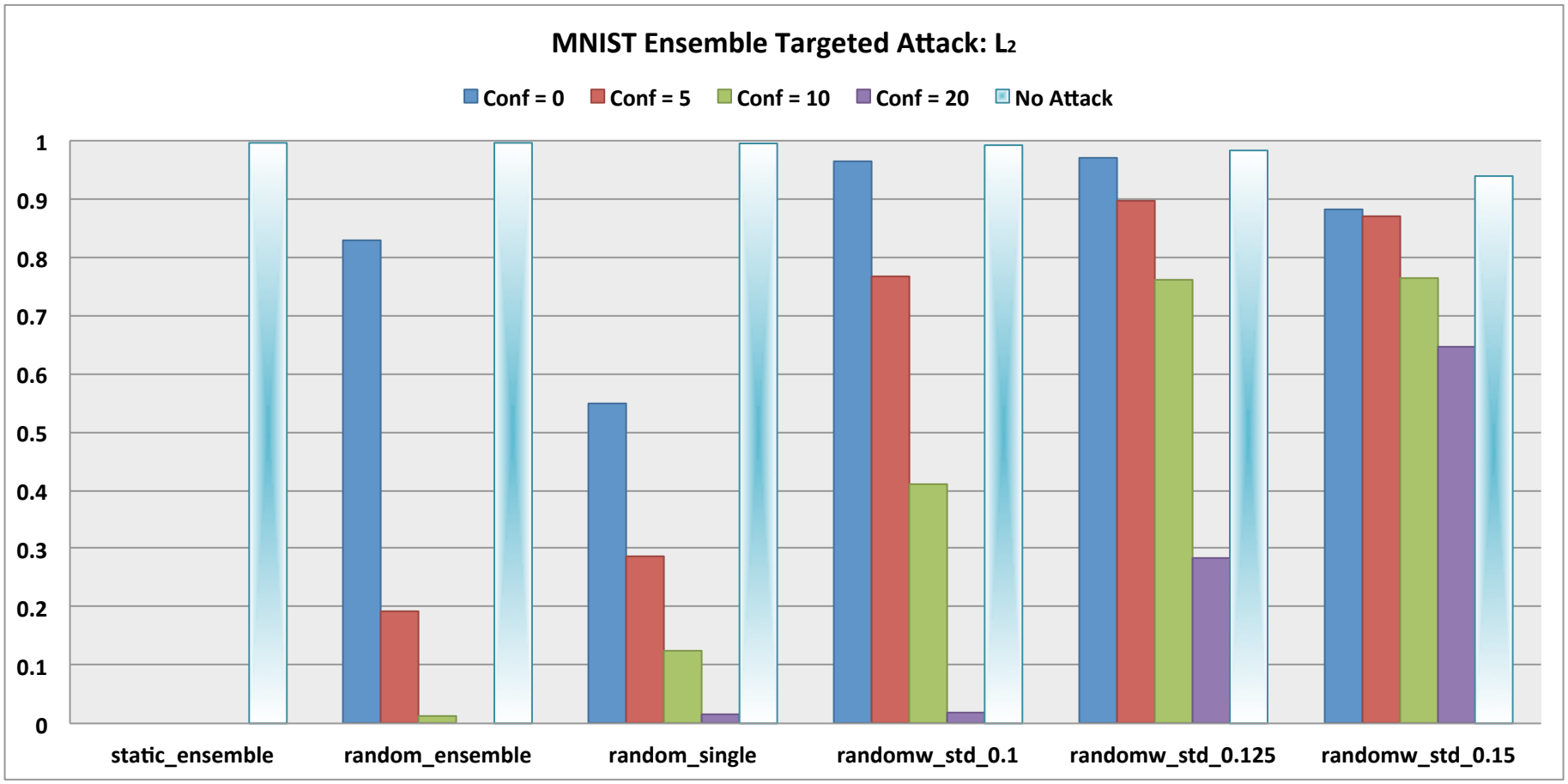
Strong Attack Scenario: Adversary has perfect knowledge of all 10 DNNs to generate adversarial samples. Attacking the ensemble requires greater data perturbation than attacking a single DNN. (MNIST)

Random-Ensemble: Creating a larger pool of 100 trained DNNs. For each new query, randomly selecting 10 DNNs. They may or may not overlap with the 10 leaked DNNs.

Random-Single: For each new query, randomly select 1 DNN from the set of 10 leaked DNNs

Random-std-x: Adding random noises to all leaked DNNs in the ensemble

DNN Models with a Randomness Factor



Discussion

IoT devices must be secured against both traditional cyber attacks and new attacks based on adversarial machine learning.

We need to design robust machine learning techniques in different application domains, where adversarial samples have different properties.

Related publications on *http://www.stat.purdue.edu/∼xbw/*

Thank you!