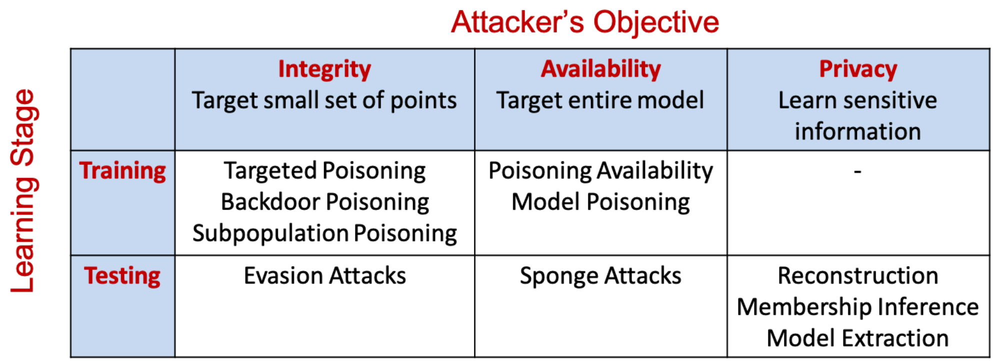
CY 7790, Lecture 5: Evasion Attacks

Giorgio Severi and Alina Oprea

September 27, 2021

The topic of the class today is evasion attacks, which are attacks against machine learning (ML) at testing time. We have discussed the taxonomy of adversarial ML attacks (see Figure 1). Today, we will focus on evasion attacks at testing time, starting with white-box scenarios and then considering some gray-box scenarios. The two papers discussed today are the first in the literature that simultaneously discovered that ML classifiers are not resilient to small manipulations of their inputs at inference time.



Attacker's Objective

Learning Stage

	Integrity Target small set of points	Availability Target entire model	Privacy Learn sensitive information
Training	Targeted Poisoning Backdoor Poisoning Subpopulation Poisoning	Poisoning Availability Model Poisoning	-
Testing	Evasion Attacks	Sponge Attacks	Reconstruction Membership Inference Model Extraction

Figure 1: Taxonomy of adversarial attacks against ML.

1 Biggio et al. Evasion Attacks against Machine Learning at Test Time

Problem Statement. The paper considers the setting of a binary classifier with malicious and benign classes with continuous and differentiable discriminant function $g(x)$. The goal of the attacker is to start with a malicious sample x^0, and find an adversarial example x within a maximum distance d_{max} from x^0 that evades the classifier **with maximum confidence**. Note that the perturbation $||x - x^0||$ does not have to be minimized, but it need to be within a maximum upper bound d_{max}. The adversarial examples belongs thus to a ball centered at x^0 of maximum distance d_{max}.

Threat Model. Two types of adversaries are considered: (1) Perfect Knowledge (PK), in which the adversary knows the feature space, the type of the classifier, and the parameters of trained model; and (2) Limited Knowledge (LK) in which the attacker knows the feature representation and the type of the classifier, but does not know either the learned classifier or its training data.

The adversarial capability involves modifications to the input data and the feature vectors. These modifications could be unlimited (for image classification), or constrained (for PDF malware classification). In the case of PDF malware classification, the features represent counts of various PDF objects, and the adversary can only increment the values of the features.

$$\arg\min_{x} \; F(\mathbf{x}) = \hat{g}(\mathbf{x}) - \frac{\lambda}{n} \sum_{i|y_i^c=-1} k\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$
$$\text{s.t.} \;\; d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max} \;\;,$$

Figure 2: Evasion attack objective. The first term minimizes the discriminant function at x to classify it as benign, while the second term maximizes the density of benign samples around x using a KDE model k.

Methodology. Given a malicious sample x^0, the goal is to find an adversarial example x at distance at most d_{max} that minimizes the discriminant function g. The discriminant function $g(x)$ estimates the probability that x is malicious, and by minimizing it, the adversarial example is classified as benign.

Assume that the attacker only minimizes $g(x)$, for $||x - x^0|| \leq d_{max}$. The main challenge is that the optimization might find a solution in a sparsely populated region of the training set (where the probability density $p(x) \approx 0$). To overcome this problem, the paper adds a second term into the optimization objective that maximizes the density of x in the region of benign examples. This is achieved via Kernel Density Estimate k, by taking an average across a set of n training points the attacker has access to. This optimization formulation is given in Figure 2.

Once the optimization problem is formulated, a gradient descent algorithm is proposed (see Figure 3). Interestingly, the same method to train ML models is also used to find adversarial examples!

```
Algorithm 1 Gradient-descent evasion attack
Input: $\mathbf{x}^0$, the initial attack point; $t$, the step size; $\lambda$, the trade-off parameter; $\epsilon > 0$ a small constant.
Output: $\mathbf{x}^*$, the final attack point.
 1: $m \leftarrow 0$.
 2: repeat
 3:     $m \leftarrow m + 1$
 4:     Set $\nabla F(\mathbf{x}^{m-1})$ to a unit vector aligned with $\nabla g(\mathbf{x}^{m-1}) - \lambda \nabla p(\mathbf{x}^{m-1}|y^c = -1)$.
 5:     $\mathbf{x}^m \leftarrow \mathbf{x}^{m-1} - t\nabla F(\mathbf{x}^{m-1})$
 6:     if $d(\mathbf{x}^m, \mathbf{x}^0) > d_{\max}$ then
 7:         Project $\mathbf{x}^m$ onto the boundary of the feasible region.
 8:     end if
 9: until $F(\mathbf{x}^m) - F(\mathbf{x}^{m-1}) < \epsilon$
10: return: $\mathbf{x}^* = \mathbf{x}^m$
```

Figure 3: Gradient descent algorithm for evasion attack generation.

The paper shows how to compute gradients for several models, including:

1. Linear classifiers: The discriminant function is $g(x) = w \cdot x + b$.
2. Kernel SVM: The discriminant function is $g(x) = \sum \alpha_i y_i k(x, x_i) + b$.
3. Multi-layer perceptrons with single hidden layer and sigmoid activation.

The gradients of the KDE component can also be computed for Gaussian kernels with L_2 and L_1 distance.

Class Discussion.

Extension to multiple layers: The method could be extended to neural networks with multiple hidden layers. The gradients for Algorithm 1 can be computed in a similar way to single-layer NN.

Failure of gradient descent: Assuming that the optimization problem is only minimizing $g(x)$, what is the intuition as to why the gradient descent optimization may fail to find adversarial samples with high confidence? The gradient descent procedure might stop in a local minima, while not crossing the decision boundary of the network. The results on NN were much worse than results on linear SVM or kernel SVM for this formulation.

KDE and Kernel functions: KDE provides an estimate of how likely a data point x is. KDE uses a kernel function k. The main difference between kernel functions and probability density functions is that Kernels could take values above 1, while PDF functions $p(x)$ have the output constrained to $[0, 1]$.

The difference between KDE and Guassian mixture models (GMMs) is that in KDE the training data defines the centers of the Gaussians, while in GMM the clusters are learned from data using the Expectation Maximization algorithm.

Extending KDE: Can you swap the KDE component with another estimate of likelihood of x such as Naive Bayes? The main challenge is to select a probability estimate that is differentiable.

Interestingly, KDE is not used in modern formulation of evasion attacks. Possible reasons are that modern datasets are larger and more evenly distributed in the space, compared to the small datasets used in this paper.

Multiple clusters in the data: What happens when the data is not really naturally clustered, does the KDE intuition still work? Most likely, if KDE is trained using representatives from different regions of the space (or clusters), it could work.

Distance metrics: What is the distance metric used in the paper? It is L_2, Euclidean distance. Is the threshold d_{max} application specific? Which distance metrics make sense for different applications?

Convergence criteria for GD: Bounding the maximum number of iterations is a necessary parameter if convergence is not reached.

NN results: Why are the neural network attack results poor for PK without the KDE component? Most likely, due to the gradient descent optimization getting into a local minimum.

2 Szegedy et al. Intriguing properties of neural networks

Problem Statement. This work analyzes the properties of neural network classifiers used in computer vision tasks. The authors expose two interesting and previously unknown properties of these models. First, the paper shows that the semantic information provided by single neurons of the last feature layer are indistinguishable by that provided by a randomized linear combination of the same layer. Second, this work proves that the assumption of good local generalization capacity of these networks is unfounded, by showing a method to generate samples leading to mis-classifications by adding imperceptible perturbations, r, to correctly classified images, x. This paper introduced the name *adversarial examples*, which has been used in follow up research in evasion attacks.

Threat Model. This work assumes the capacity of the adversary to inspect the entirety of the victim model, f, and compute gradients relative to the loss function. Moreover, to induce the mis-classification, the adversary must also be capable of modifying data points at inference time. Given the domain considered, computer vision, the modifications to pixel values are only bounded by the scaled pixel range $[0, 1]$: $x + r \in [0, 1]^m$.

Methodology. The first observation derives from the visual analysis of test set images maximizing the activation of single neuron units, compared against images maximizing activation values along a randomized combination of the neurons of the last feature layer.

To show that neural network classifiers are locally unstable, the authors formulate the problem of finding an adversarial perturbation as an optimization objective, as shown in Figure 4.

- Minimize $||r||_2$ subject to:
  1. $f(x + r) = l$
  2. $x + r \in [0, 1]^m$

Figure 4: Optimization formulation for finding the minimal perturbation that causes a targeted mis-classification.

To solve it, the problem is reformulated as concurrently minimizing the size of the perturbation and the loss of the model with respect to the target class, Figure 5, using the penalty method. L-BFGS is then used to find a numerical solution.

- Minimize $c|r| + \text{loss}_f(x + r, l)$ subject to $x + r \in [0, 1]^m$

Figure 5: Concurrent minimization of both the size of the perturbation and the loss of the model with respect to the target class.

Finally, the authors introduce a method to compute upper bounds to the instability of the model based on computing the Lipschitz constant of each layer of the network and aggregating them through multiplication. Large values for these upper bounds do not directly implicate the presence of adversarial samples, however, small values would indicate that such points cannot exist.

Class Discussion.

Semantic meaning: What does semantic meaning imply? In the paper they visually inspect the images and claim that there is semantic meaning along a random direction in the representation space. Traditional computer vision uses human-designed filters that are activated for specific classes. Does this hold for NN? Section 3 shows that individual neurons in the activation space do not correspond to individual features.

Layers of the network: In which layer are the activations computed for Section 3? It's the last representation layer before softmax, which carries semantic meaning.

Other layers: Did they try measuring activations in other layers of the CNN? This is not specified in the paper.

Targeted and untargeted attacks: Let x_0 be a sample with correct class y_0. A targeted attack minimizes the loss on the target class $\ell \neq y_0$, and the objective becomes: $c||r||_2 + \text{loss}(x + r, \ell)$. For untargeted attack, the loss on the correct class can be maximized and the objective becomes: $c||r||_2 - \text{loss}(x + r, y_0)$.

Optimization method: The optimization is solved with L-BFGS, a quasi Newton method. Newton methods use the inverse of the Hessian matrix H, but in L-BFGS, the Hessian matrix is approximated numerically for efficiency.