CY 7790

Special Topics in Security and Privacy: Machine Learning Security and Privacy

Fall 2021

Alina Oprea
Associate Professor
Khoury College of Computer Science

September 27 2021

Adversarial Machine Learning: Taxonomy

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

Threat Model

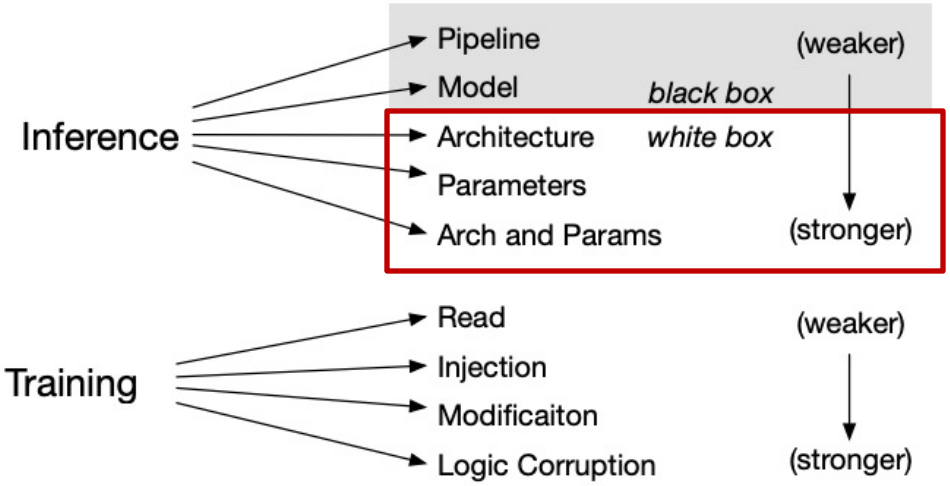
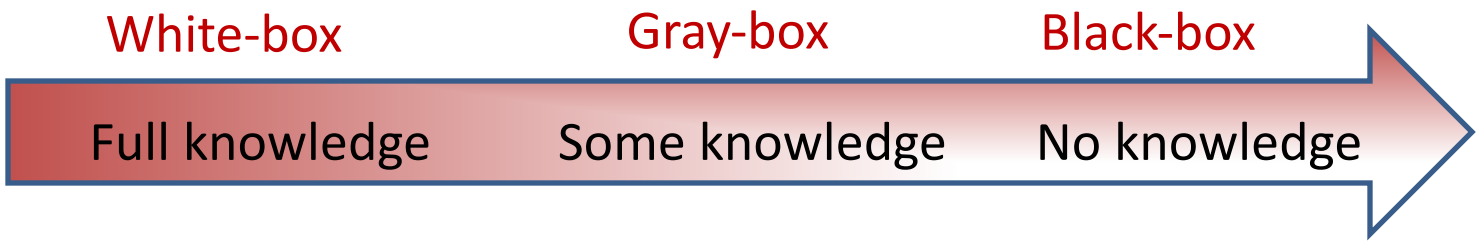
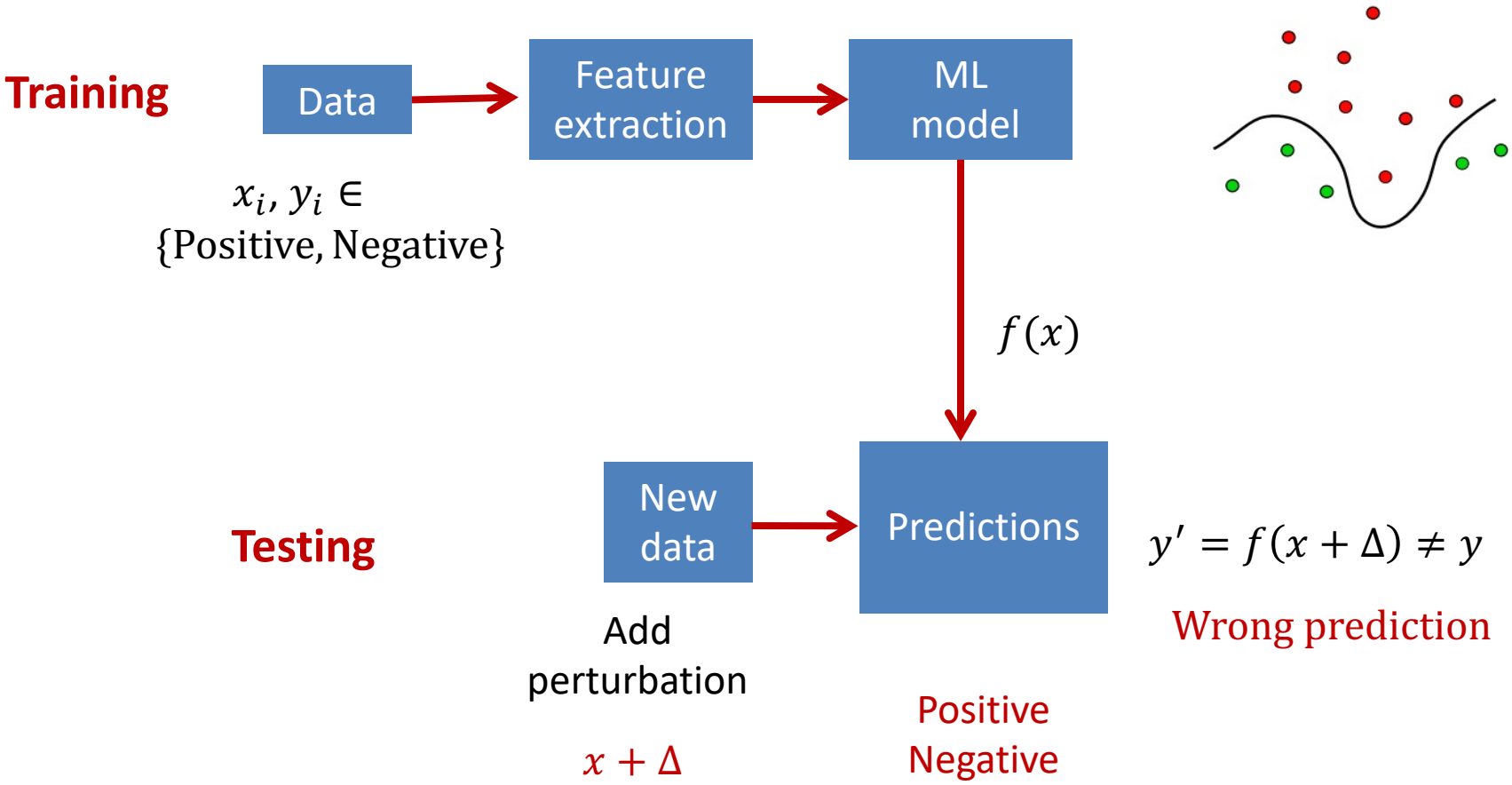


Figure 3. **Adversarial Capabilities.** Adversaries attack ML systems at inference time by exploiting model internal information (white box) or probing the system to infer system vulnerabilities (black box). Adversaries use read or write access to the training data to mimic or corrupt the model.



Evasion Attacks



- Modify testing point by adding small perturbation to misclassify it

Adversarial Examples

Given seed sample, x, x' is an *adversarial example* iff:

$f(x') = t$ Class is t (targeted)

$\Delta(x, x') \leq \delta$ Difference below threshold

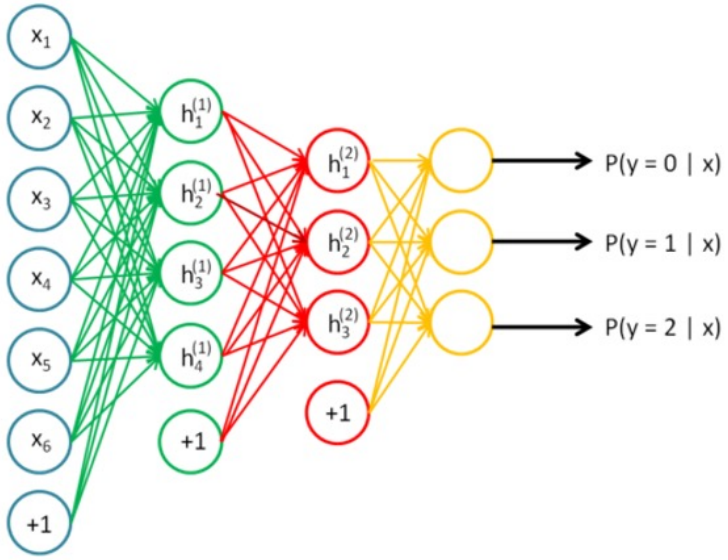
$\Delta(x, x')$ is defined in some (simple!) metric space:

L_0 norm (# different), L_1, L_2 norm ("Euclidean distance"), L_∞

Assumption (to map to earlier definition):
small perturbation does not change class in "Reality Space"

Evasion Attacks For ML Classifiers

WHITE-BOX



Optimization Formulation

Given input x
Find adversarial example
$x' = x + \delta$
Optional: target class t

$$\min_{\delta} Obj\ (x + \delta, t)$$

subject to constraints (max confidence, min perturbation)

- Assume continuous domains, white-box settings
- Optimization problem solved with gradient descent
- Attacks differ in objective formulation and method to solve optimization
  - Variants: maximize confidence or minimize distance

White-Box Evasion Attacks

- Fast Gradient Sign Method (FGSM)
  - One step attack
  - Goodfellow et al. Explaining and Harnessing Adversarial Examples, 2015
- Iterative attacks
  - Biggio et al. Evasion attacks against machine learning at test time, 2013
  - Szedegy et al. Intriguing properties of neural networks, 2014 13
  - Carlini and Wagner. Towards Evaluating the Robustness of Neural Networks, 2017

Biggio et al. Evasion attacks against machine learning at test time ECML-KDD 2013

Problem

- How to construct max confidence adversarial examples
- Attacker scenarios
  - Perfect Knowledge (PK)
  - Limited Knowledge (LK)
- Attacker capabilities
  - Modifications of testing data
  - Some constraints on feature space (PDF case, attacker can only increase features)

Methodology

Mal → Benign

LINEAR: $g(x) = wx + b$

2.3 Attack Strategy

Under the above assumptions, for any target malicious sample $\mathbf{x}^0$ (the adversary's desired instance), an optimal attack strategy finds a sample $\mathbf{x}^*$ to minimize $g(\cdot)$ or its estimate $\hat{g}(\cdot)$, subject to a bound on its distance[6] from $\mathbf{x}^0$:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \ \hat{g}(\mathbf{x}) \quad \text{s.t.} \quad d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max}. \tag{1}$$

- Binary classification: +1 (Malicious), -1 (Benign)
- $g(x)$ is the model prediction on x
  - Provides an estimate of $p(y = \text{Malicious}|x)$
- Issue: it might get into regions where $p(x) \approx 0$

Optimization Objective

$$\arg\min_{x} \; F(\mathbf{x}) = \hat{g}(\mathbf{x}) + \frac{\lambda}{n} \sum_{i|y_i^c=-1} k\left(\frac{\mathbf{x}-\mathbf{x}_i}{h}\right)$$

$$\text{s.t.} \;\; d(\mathbf{x}, \mathbf{x}^0) \leq d_{\max} \;\; ,$$

- Naive Bayes?
- Density Estimation

KDE

L_2: $\|x - x^0\|_2 \leq d_{max}$

- Solution: add a second term to maximize the probability density around training samples
- Parameters
  - n: number of samples available to the adversary
  - λ: weight of the second term in the objective

Density Estimation

Given a set of n data samples $\mathbf{x}_1$,..., $\mathbf{x}_n$, we can estimate the density function $p(\mathbf{x})$, so that we can output $p(\mathbf{x})$ for any new sample $\mathbf{x}$. This is called *density estimation*.

Slide from Xia Hong

Kernel Density Estimation

A kernel function K is a function such that...

- $K(x) \geq 0$ for all $-\infty < x < \infty$
- $K(-x) = K(x)$
- $\int_{-\infty}^{\infty} K(x)\mathrm{d}x = 1$

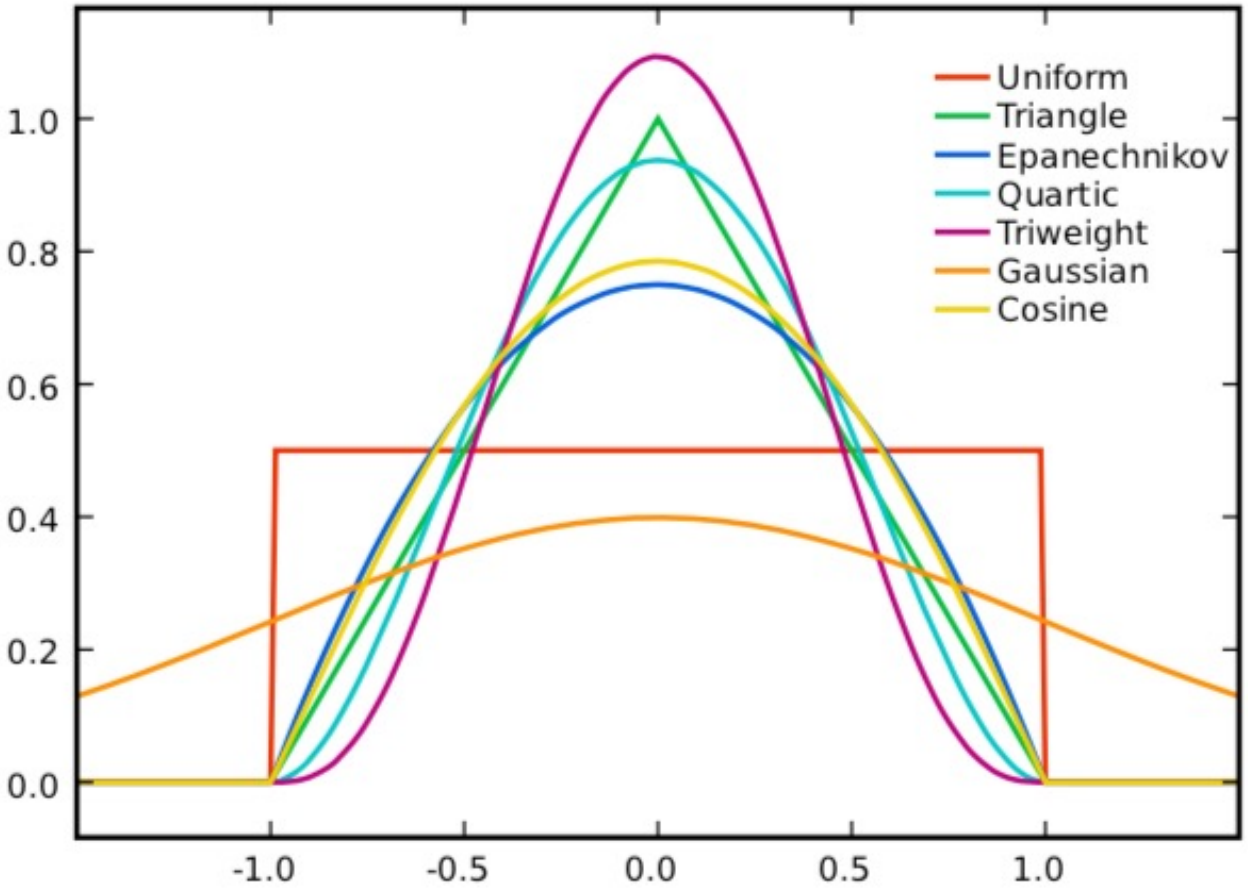
A simple example is the uniform (or box) kernel:

$$K(x) = \begin{cases} 1 & \text{if } -1/2 \leq x < 1/2 \\ 0 & \text{otherwise} \end{cases}$$

Another popular kernel function is the Normal kernel (pdf) with $\mu = 0$ and σ fixed at some constant:

$$K(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

Kernel Visualization



From http://upload.wikimedia.org/wikipedia/commons/4/47/Kernels.svg

Kernel Density Estimate

$x_1, \ldots, x_n$

$$K_h^{(x_i)}(x) = \frac{1}{h} K\left(\frac{x - x_i}{h}\right)$$

Scaled and centered kernel

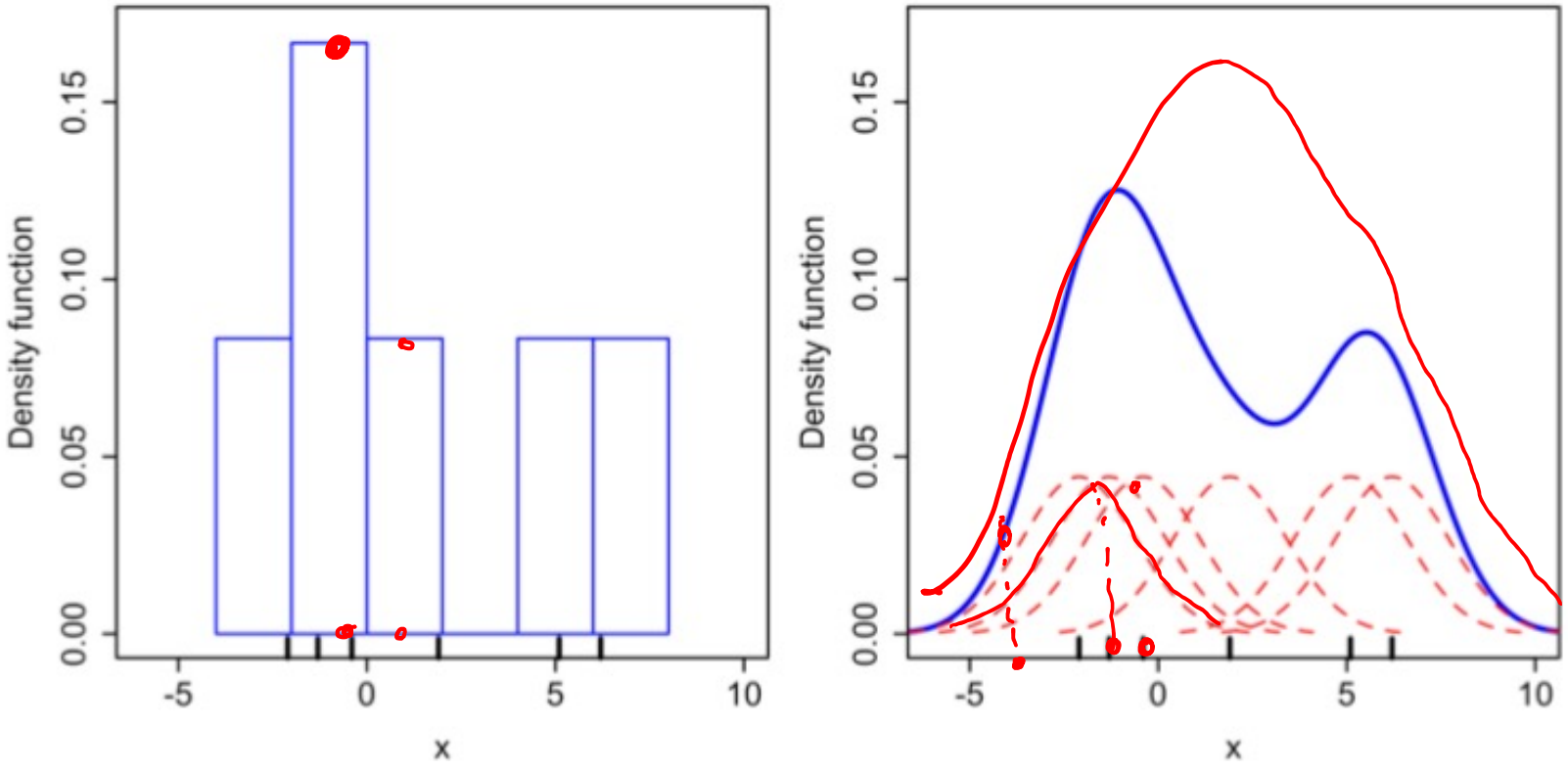
Given a random sample $x_i \overset{\text{iid}}{\sim} f(x)$, the kernel density estimate of f is

$$\hat{f}(x) = \frac{1}{n} \sum_{i=1}^{n} K_h^{(x_i)}(x)$$

$$= \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

where h is now referred to as the bandwidth (instead of bin width).

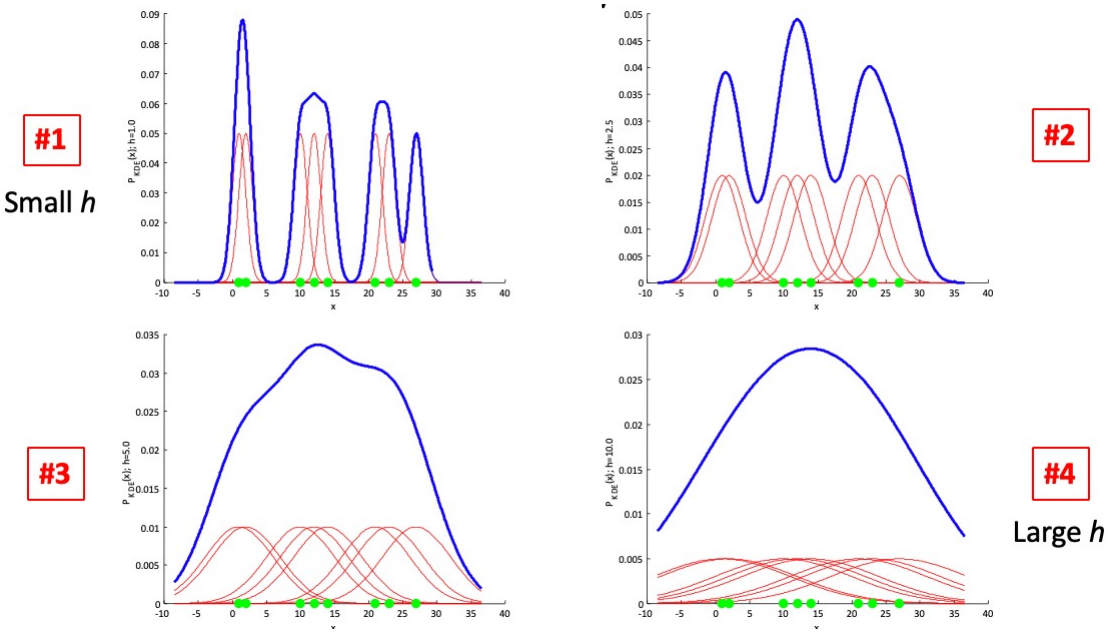
Kernel Density Estimate: Visualization



From http://en.wikipedia.org/wiki/Kernel_density_estimation

Bandwidth Selection

- The problem of choosing $h = \sigma$ is crucial in density estimation
- Small bandwidth: over-fitting
- Large bandwidth: can mask the data structure



Gradient Descent Attack

$\min_x F(x)$ $g(x)$

Algorithm 1 Gradient-descent evasion attack

Input: $\mathbf{x}^0$, the initial attack point; t, the step size; λ, the trade-off parameter; $\epsilon > 0$ a small constant.

Output: $\mathbf{x}^*$, the final attack point.

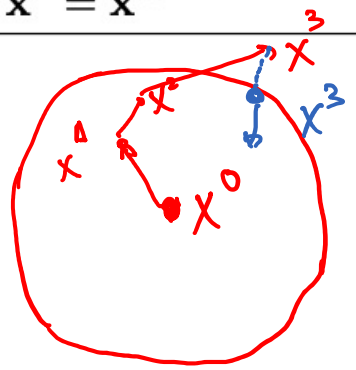
$y = \frac{x}{\|x\|}$, $\|y\| = 1$

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:** $\mathbf{x}^* = \mathbf{x}^m$

KDE.

Projected Gradient Descent

STOP OR BOUND ITERAT.



Gradients

Linear classifiers. Linear discriminant functions are $g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$ where $\mathbf{w} \in \mathbb{R}^d$ is the feature weights and $b \in \mathbb{R}$ is the bias. Its gradient is $\nabla g(\mathbf{x}) = \mathbf{w}$.

Support vector machines. For SVMs, $g(\mathbf{x}) = \sum_i \alpha_i y_i k(\mathbf{x}, \mathbf{x}_i) + b$. The gradient is thus $\nabla g(\mathbf{x}) = \sum_i \alpha_i y_i \nabla k(\mathbf{x}, \mathbf{x}_i)$. In this case, the feasibility of our approach depends on whether the kernel gradient $\nabla k(\mathbf{x}, \mathbf{x}_i)$ is computable as it is for many numeric kernels. For instance, the gradient of the RBF kernel, $k(\mathbf{x}, \mathbf{x}_i) = \exp\{-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\}$, is $\nabla k(\mathbf{x}, \mathbf{x}_i) = -2\gamma \exp\{-\gamma \|\mathbf{x} - \mathbf{x}_i\|^2\}(\mathbf{x} - \mathbf{x}_i)$, and for the polynomial kernel, $k(\mathbf{x}, \mathbf{x}_i) = (\langle \mathbf{x}, \mathbf{x}_i \rangle + c)^p$, it is $\nabla k(\mathbf{x}, \mathbf{x}_i) = p(\langle \mathbf{x}, \mathbf{x}_i \rangle + c)^{p-1} \mathbf{x}_i$.

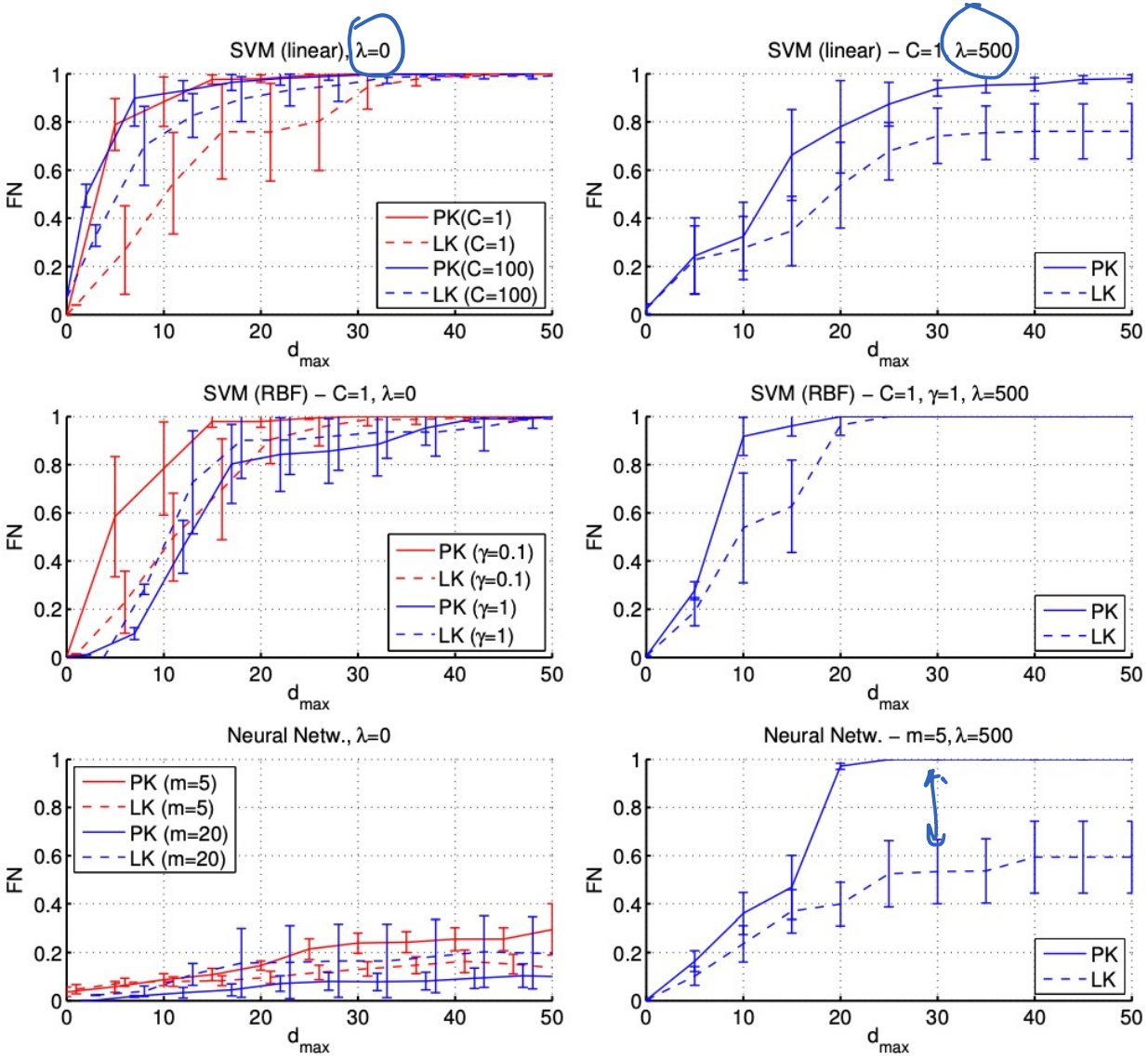
Neural networks. For a multi-layer perceptron with a single hidden layer of m neurons and a sigmoidal activation function, we decompose its discriminant function g as follows (see Fig. 2): $g(\mathbf{x}) = (1+e^{-h(\mathbf{x})})^{-1}$, $h(\mathbf{x}) = \sum_{k=1}^m w_k \delta_k(\mathbf{x}) + b$, $\delta_k(\mathbf{x}) = (1 + e^{-h_k(\mathbf{x})})^{-1}$, $h_k(\mathbf{x}) = \sum_{j=1}^d v_{kj} x_j + b_k$. From the chain rule, the i^{th} component of $\nabla g(\mathbf{x})$ is thus given by:

$$\frac{\partial g}{\partial x_i} = \frac{\partial g}{\partial h} \sum_{k=1}^m \frac{\partial h}{\partial \delta_k} \frac{\partial \delta_k}{\partial h_k} \frac{\partial h_k}{\partial x_i} = g(\mathbf{x})(1 - g(\mathbf{x})) \sum_{k=1}^m w_k \delta_k(\mathbf{x})(1 - \delta_k(\mathbf{x})) v_{ki} \quad .$$

Gradients of KDE

$$\text{KDE} = \exp\left(-\frac{\|x - x_i\|_2^2}{h}\right)$$

Results – PDF Malware



Strengths

- One of the first papers showing how to use Gradient Descent optimization for evasion attacks
- General method: SVMs, supporting different kernels, neural networks
- Different threat models: PK and LK

Limitations

- Binary classification only
- Using discriminant function g directly into the optimization
  - Future attacks will use a loss function
- Not much evaluation on MNIST
- Features for PDF are not very resilient

Discussion Points

- Optimization objective including KDE component to get adversarial examples in high-density regions of training data
- Limited Knowledge vs Perfect Knowledge Adversary
- Application-specific constraints (PDF malware)
- Robustness of different models: linear SVM, kernel SVM, neural networks

Szegedy et al. Intriguing properties of neural networks. 2013

Two observations

- Individual activations from last layer are as semantically meaningful as linear combination of activations
  - Last layer neurons do not correspond to features with semantic meaning
- Neural networks have blind spots
  - Can find adversarial examples at small distortion for a number of architectures
  - Adversarial examples transfer across architectures and across training sets

Semantic Meaning of Last Layer

(a) Unit sensitive to white flowers.

(c) Unit senstive to round, spiky flowers.

Images maximizing activations of single neurons in last layer

(a) Direction sensitive to white, spread flowers.

(c) Direction sensitive to spread shapes.

Images maximizing activations in a random direction (linear combination of neurons)

Optimization Formulation

We denote by $f : \mathbb{R}^m \longrightarrow \{1 \dots k\}$ a classifier mapping image pixel value vectors to a discrete label set. We also assume that f has an associated continuous loss function denoted by $\text{loss}_f : \mathbb{R}^m \times \{1 \dots k\} \longrightarrow \mathbb{R}^+$. For a given $x \in \mathbb{R}^m$ image and target label $l \in \{1 \dots k\}$, we aim to solve the following box-constrained optimization problem:

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

x_0, y_0

$l \neq y_0$

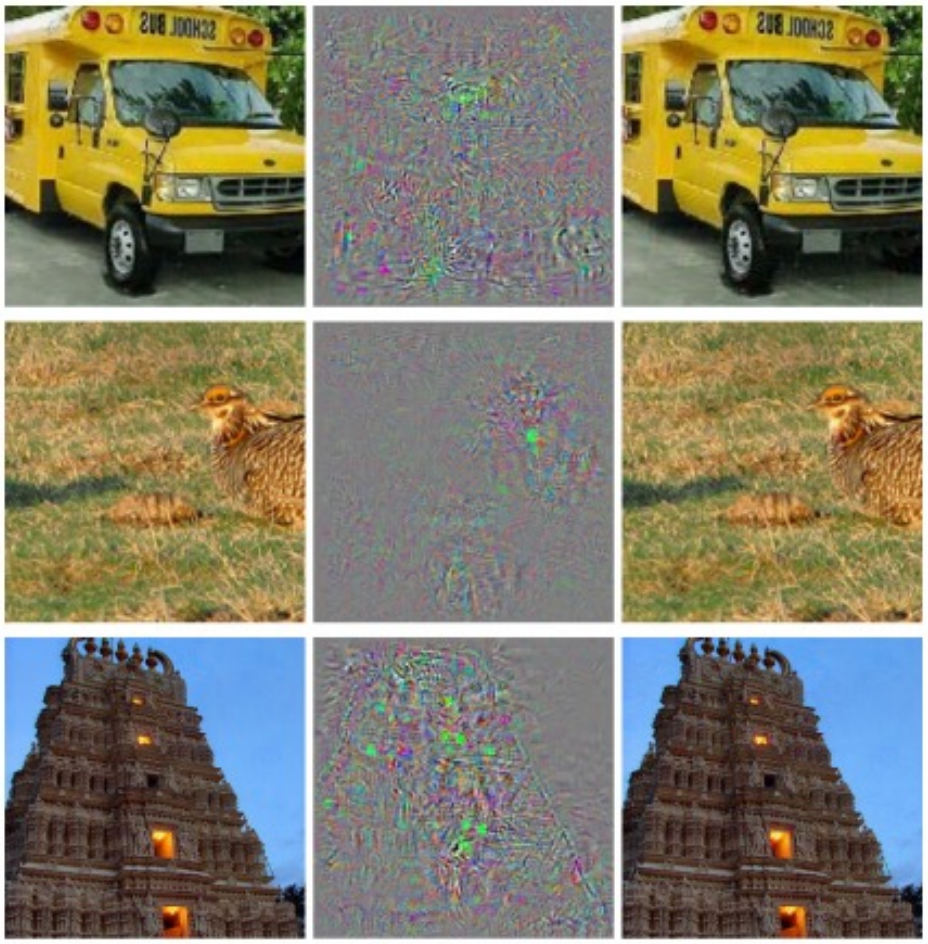
Penalty method

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

Solved using L-BFGS method

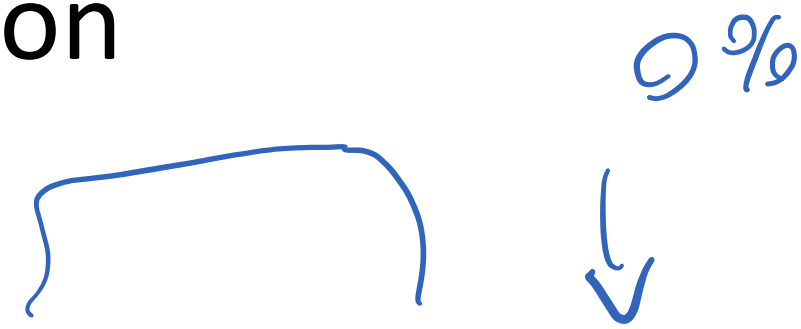
- Second-order method (in the class of quasi Newton methods)

Adversarial examples



(a)

Distortion



Model Name	Description	Training error	Test error	Av. min. distortion
FC10(10^{-4})	Softmax with $\lambda = 10^{-4}$	6.7%	7.4%	0.062
FC10(10^{-2})	Softmax with $\lambda = 10^{-2}$	10%	9.4%	0.1
FC10(1)	Softmax with $\lambda = 1$	21.2%	20%	0.14
FC100-100-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.64%	0.058
FC200-200-10	Sigmoid network $\lambda = 10^{-5}, 10^{-5}, 10^{-6}$	0%	1.54%	0.065
AE400-10	Autoencoder with Softmax $\lambda = 10^{-6}$	0.57%	1.9%	0.086

Table 1: Tests of the generalization of adversarial instances on MNIST.

Transfer of adversarial examples

	FC10(10^{-4})	FC10(10^{-2})	FC10(1)	FC100-100-10	FC200-200-10	AE400-10	Av. distortion
FC10(10^{-4})	100%	11.7%	22.7%	2%	3.9%	2.7%	0.062
FC10(10^{-2})	87.1%	100%	35.2%	35.9%	27.3%	9.8%	0.1
FC10(1)	71.9%	76.2%	100%	48.1%	47%	34.4%	0.14
FC100-100-10	28.9%	13.7%	21.1%	100%	6.6%	2%	0.058
FC200-200-10	38.2%	14%	23.8%	20.3%	100%	2.7%	0.065
AE400-10	23.4%	16%	24.8%	9.4%	6.6%	100%	0.086
Gaussian noise, stddev=0.1	5.0%	10.1%	18.3%	0%	0%	0.8%	0.1
Gaussian noise, stddev=0.3	15.6%	11.3%	22.7%	5%	4.3%	3.1%	0.3

Table 2: Cross-model generalization of adversarial examples. The columns of the Tables show the error induced by distorted examples fed to the given model. The last column shows average distortion wrt. original training set.

Explain Unstability

Mathematically, if $\phi(x)$ denotes the output of a network of K layers corresponding to input x and trained parameters W, we write

$$\phi(x) = \phi_K(\phi_{K-1}(\dots\phi_1(x; W_1); W_2)\dots; W_K) \ ,$$

where ϕ_k denotes the operator mapping layer $k-1$ to layer k. The unstability of $\phi(x)$ can be explained by inspecting the upper Lipschitz constant of each layer $k = 1 \dots K$, defined as the constant $L_k > 0$ such that

$$\forall\, x,\, r \ , \ \|\phi_k(x; W_k) - \phi_k(x + r; W_k)\| \leq L_k \|r\| \ .$$

The resulting network thus satsifies $\|\phi(x) - \phi(x+r)\| \leq L\|r\|$, with $L = \prod_{k=1}^{K} L_k$.

ReLU $\quad \phi_k(x; W_k, b_k) = \max(0, W_k x + b_k)$

$$\|\phi_k(x; W_k) - \phi_k(x+r; W_k)\| = \|\max(0, W_k x + b_k) - \max(0, W_k(x+r) + b_k)\| \leq \|W_k r\| \leq \|W_k\| \|r\| \ ,$$

$$L_k \leq \|W_k\|.$$

Bounds for AlexNet

Layer	Size	Stride	Upper bound
Conv. 1	$3 \times 11 \times 11 \times 96$	4	2.75
Conv. 2	$96 \times 5 \times 5 \times 256$	1	10
Conv. 3	$256 \times 3 \times 3 \times 384$	1	7
Conv. 4	$384 \times 3 \times 3 \times 384$	1	7.5
Conv. 5	$384 \times 3 \times 3 \times 256$	1	11
FC. 1	9216×4096	N/A	3.12
FC. 2	4096×4096	N/A	4
FC. 3	4096×1000	N/A	4

Table 5: Frame Bounds of each rectified layer of the network from [9].

Conclusions

- Lipschitz constant are fairly large for each layer and the full network's Lipschitz constant is the product of individual layers
- Networks are not stable in terms of Lipschitz definitions

Strengths

- Considered different NN architectures
- Try to explain unstability via Lipschitz analysis
- Different optimization formulation
- Transferability of adversarial examples
- Asks many questions that spark follow up work
  - Why are NN susceptible to adversarial examples?
  - What is the tradeoff between good generalization and resilience to adversarial examples?

Limitations

- Writing not always clear (Section 3)
  - Relying on visual analysis
- Only considered NN
- Definition of adversarial examples is related to human perception

Discussion Points

- Two different optimization approaches to generate adversarial examples
  - Different objectives
- Adversarial examples always exist
  - What is the min perturbation
- How to define adversarial examples in other domains (healthcare or cyber security) where human perception is not applicable?