CSCI 4260/6260: Data Security & Privacy

Adversarial Attack

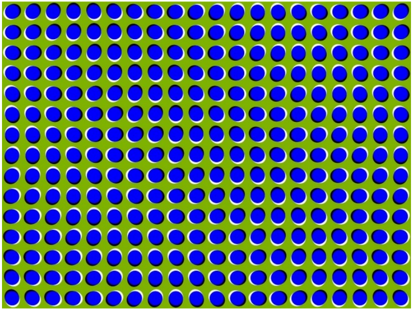
Jaewoo Lee
jaewoo.lee@uga.edu

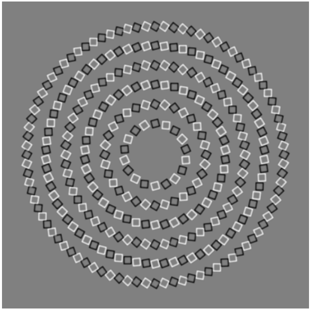
October 12, 2021

Department of Computer Science

UNIVERSITY OF GEORGIA

Human Vision





- Optical illusions for human vision

Fooling NN

- Machine learning algorithms can be fooled by *perturbed* images.

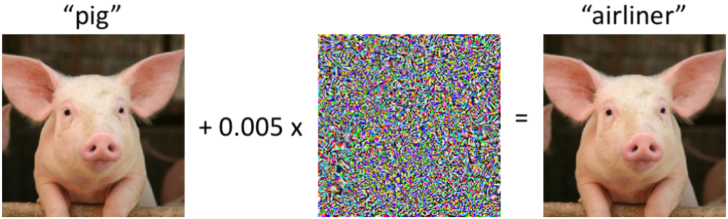
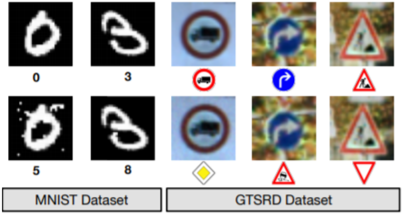


Fig. 1. An adversarial example $f(x) \neq f(x+h)$

- Perturbation is *not* human recognizable.

Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy.
Explaining and Harnessing Adversarial Examples.
ICLR 2015

Fooling NN



Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus.
Intriguing Properties of Neural Networks.
ICLR 2014

Fooling NN in a real-world

- Adversarial Patch
- Watch this video.

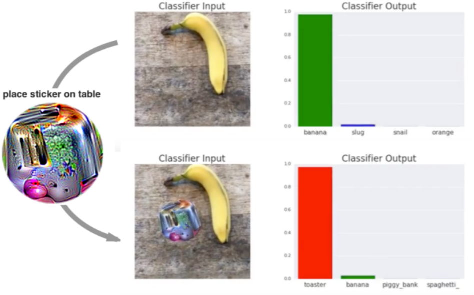


Fig. 2. Banana or toaster?

Brown, Tom B., Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer.
Adversarial Patch
May 16, 2018, http://arxiv.org/abs/1712.09665.

Fooling NN in a real-world

- Classifying turtles
- Watch this video

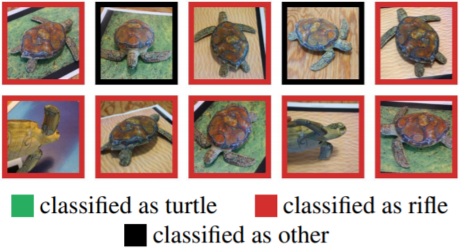


Fig. 3. 3D-printed turtles

Fig. 4. Impersonation attack

Sharif, Mahmood, Sruti Bhagavatula, Lujo Bauer, and Michael K. Reiter.
Accessorize to a crime: Real and stealthy attacks on state-of-the-art face recognition
In Proceedings of the 2016 acm sigsac conference on computer and communications security, 2016

Adversarial Attack: definition

Let $x_0 \in \mathbb{R}^d$ be a data point and y_0 denote its class *label*. Suppose we have a *classifier* $f : \mathcal{X} \to \mathcal{Y}$.

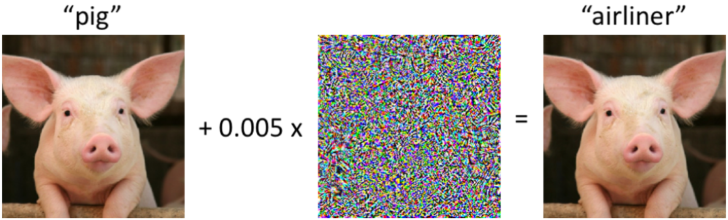


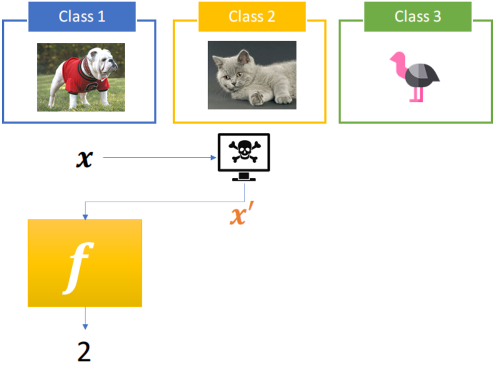
Fig. 5. An adversarial example $f(x) \neq f(x+h)$

- **Adversarial example**: *perturbation* of x_0 such that
  - (closenss): $\|x - x_0\| < \epsilon$ for a small constant ϵ
  - (mis-classification): $y = f(x) \neq f(x_0) = y_0$

⊗ A common mis-belief: AE's are unique to deep learning

Setup

Suppose we have an example x_0 from class y_0.



- **targeted**: $f(x') = y_{\text{target}}$
- **untargeted**: $f(x') \neq y_0$

Adversarial Attack

How to generate adversarial examples?

- Recall we have picked an example $\mathbf{x}_0$ (with label y_0).
- Adversary's algorithm $\mathbf{x} = \mathcal{A}(\mathbf{x}_0)$
  - Additive $\mathbf{x} = \mathbf{x}_0 + \mathbf{h}$
  - Multiplicative $\mathbf{x} = \mathbf{x}_0 \odot \mathbf{h}$
  - Non-linear general mapping $\mathcal{A}(\cdot)$
- We will focus on the *additive* form.
  - domain
  - interpretation

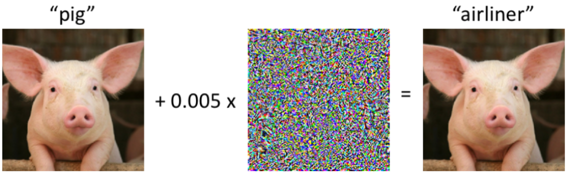
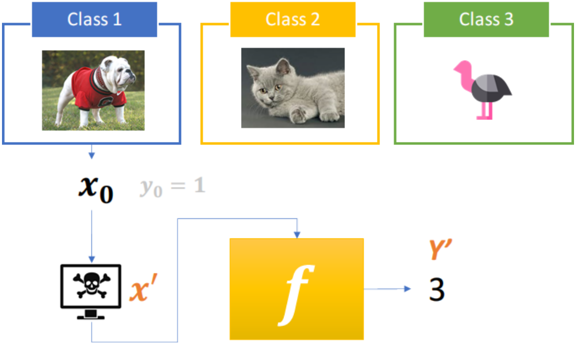


Fig. 6. An adversarial example $f(x) \neq f(x+h)$

Formulation

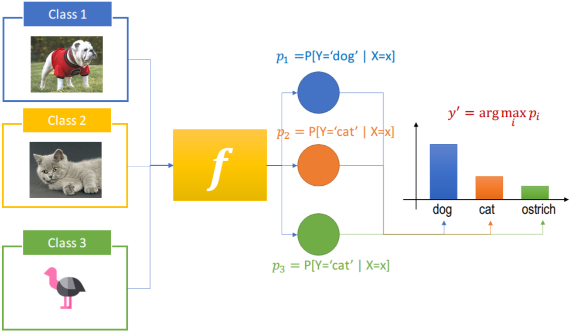


- $f : \mathcal{X} \to \mathcal{Y}$: a classifier, y_{target}: target class
- Given $(\mathbf{x}_0, y_0)$, we aim to generate $\mathbf{x}$ such that

$$\mathcal{A}(\mathbf{x}_0) = \mathbf{x}_0 + \mathbf{h} = \mathbf{x}\,, \quad f(\mathbf{x}) = y_{\text{target}}\,.$$

Multiclass Classification

- We have a classifier f.



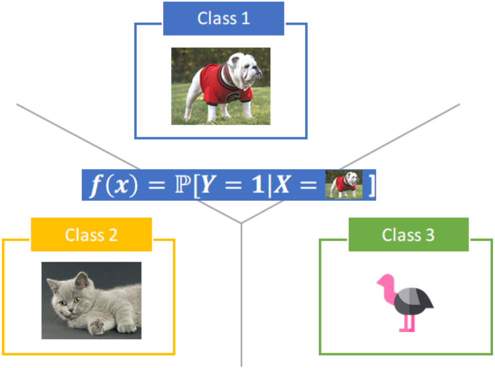
  - f outputs *probabilities* $\mathbf{p} = (p_1, p_2, \ldots, p_K)$.

$$p_k = \mathbb{P}[Y = k \mid X = \mathbf{x}]$$

  - We want $f(\mathbf{x}) = y_{\text{target}}$. That is,

$$\mathcal{A}(\mathbf{x}_0) = \mathbf{x}_0 + \mathbf{h} = \mathbf{x}\,, \quad y_{\text{target}} = \arg\max_i \, p_i (\text{Equivalently, } p_{y_{\text{target}}} = \max_i p_i.)$$

Multiclas Classification



Optimization Formulation

- $\mathbf{x}_0$: original image (with label y_0)
- $\mathcal{A}(\mathbf{x}_0) = \mathbf{x}_0 + \mathbf{h} = \mathbf{x}$: perturbed image
- $\mathbf{x}$ is *misclassified* to y_{target}, meaning
  - $p_{y_{\text{target}}} \geq p_1$
  - $p_{y_{\text{target}}} \geq p_2$
  - $\vdots$
  - $p_{y_{\text{target}}} \geq p_K$

Minimum Perturbation Attack

The *minimum perturbation attack* finds a perturbed data $\mathbf{x}$ by solvign

$$\begin{array}{ll} \underset{\mathbf{h}}{\text{minimize}} & \|\mathbf{h}\| \\ \text{subject to} & \max_j p_j(\mathbf{x}) - p_{y_{\text{target}}} \leq 0\,. \end{array}$$

Alternative Formulation

In the minimum norm attack, we

- find the *smallest* perturbation $\mathbf{h} = \mathbf{x} - \mathbf{x}_0$
- while maintaining $p_{y_{\text{target}}}$ is the largest.
- minimal perturbation $\to$ difference unrecognizable

Alternatively, we can

- allow any perturbations with magnitude smaller than τ
- while maximizing the *confidence* in misclassification.

Constrained Perturbation Attack

The *constrained perturbation attack* finds a perturbed data $\mathbf{x}$ by solving

$$\begin{aligned} \underset{\mathbf{x}}{\text{minimize}} \quad & \max_{j \neq t} p_j(\mathbf{x}) - p_t(\mathbf{x}) \\ \text{subject to} \quad & \|\mathbf{x} - \mathbf{x}_0\| \leq \tau\,. \end{aligned}$$

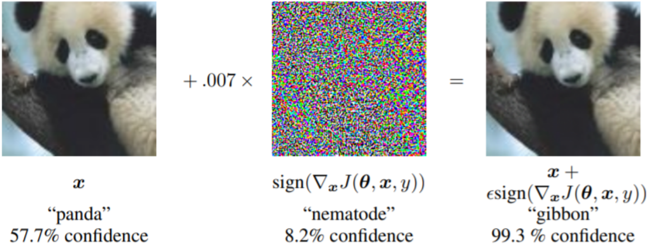
Which Optimization to use?

Two optimization may looks different, but they are the same.

- For every solution of the minimum perturbation attack,
- we can obtain the same solution by appropriately choosing τ.

Fast Gradient Sign Method

- A method to generate adversarial examples

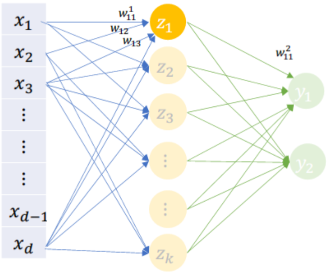


Goodfellow, Ian J., Jonathon Shlens, and Christian Szegedy.
Explaining and Harnessing Adversarial Examples.
ICLR 2015

Linear Classifiers: setup

Consider a linear classifier $f(\mathbf{x}) = \mathbf{w}^\intercal \mathbf{x} = \langle \mathbf{w}, \mathbf{x} \rangle$.

- $\mathbf{x} \in \mathbb{R}^d$: input feature vector $\mathbf{x} = (x_1, x_2, \ldots, x_d)^\intercal$
- $\mathbf{w} \in \mathbb{R}^d$: a set of *weights* assigned to x_i's



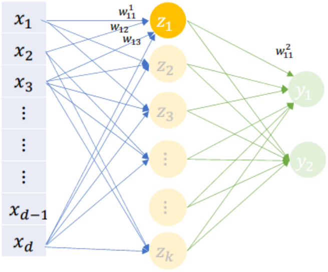
$$\begin{bmatrix} w_1 & w_2 & \cdots & w_d \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix}$$

$$= w_1 x_1 + w_2 x_2 + \cdots w_d x_d$$

$$= \sum_{i=1}^{d} w_i x_i$$

Consider an additive perturbation $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\eta}$.

- $\mathbf{x}$: an image,
- $\boldsymbol{\eta} \in \mathbb{R}^d$: perturbation (small, $\|\boldsymbol{\eta}\|_\infty < \epsilon$)



- Output $f(\mathbf{x}) = \mathbf{w}^\intercal \mathbf{x}$
- $\mathbf{w}^\intercal \tilde{\mathbf{x}} = \mathbf{w}^\intercal (\mathbf{x} + \boldsymbol{\eta})$

$$= \underbrace{\mathbf{w}^\intercal \mathbf{x}}_{\text{orginal}} + \underbrace{\mathbf{w}^\intercal \boldsymbol{\eta}}_{\text{extra}}$$

- The extra term can *increase* the activation!

Linear Behavior

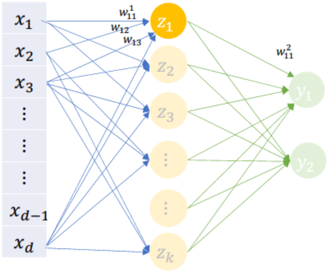
- $\mathbf{w}^\intercal \tilde{\mathbf{x}} = \underbrace{\mathbf{w}^\intercal \mathbf{x}}_{\text{orginal}} + \underbrace{\mathbf{w}^\intercal \boldsymbol{\eta}}_{\text{extra}}, \qquad \|\boldsymbol{\eta}\|_\infty < \epsilon$
- Suppose we set $\boldsymbol{\eta} = \text{sign}(\mathbf{w})$. (what will happen?)

$$\text{sign}(x) = \begin{cases} 1 & \text{if } x > 0, \\ 0 & \text{if } x = 0, \\ -1 & \text{if } x < 0. \end{cases}$$

- Let's take an example.
  - $\mathbf{w} = (0.1, -0.2, 0.9, -0.01)^\intercal$
  - $\text{sign}(\mathbf{w}) = (1, -1, 1, -1)^\intercal$
  - $\langle \mathbf{w}, \text{sign}(\mathbf{w}) \rangle = 0.1 + 0.2 + 0.9 + 0.01$
- To bound the magnitude of perturbation, we set $\boldsymbol{\eta} = \epsilon\, \text{sign}(\mathbf{w})$ (verify this).
  - jointly introduce a large increase in activation
  - but each dimensional value is small (ϵ)

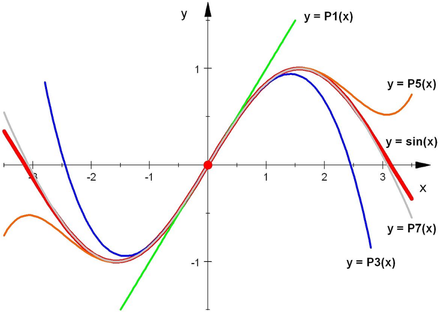
Deep Neural Network

Let $J(\boldsymbol{\theta}, \mathbf{x}, y)$ be the *cost/error* function of NN.



- Approximate J with a linear function (but how?)

Talyor Expansion



$$f(x+h) = f(x) + f'(x)h + \frac{1}{2!}f''(x)h^2 + \cdots$$

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x})^\mathsf{T}(\mathbf{x}-\mathbf{x}_0) + \frac{1}{2!}(\mathbf{x}-\mathbf{x}_0)^\mathsf{T}\nabla^2 f(\mathbf{x})(\mathbf{x}-\mathbf{x}_0) + \cdots$$

Deep Neural Network

Linear approximation of *cost* function:

Linearized Objective

$$J(\boldsymbol{\theta}, \tilde{\mathbf{x}}, y) \approx J(\boldsymbol{\theta}, \mathbf{x}, y) + \nabla J(\boldsymbol{\theta}, \mathbf{x}, y)^{\intercal}(\tilde{\mathbf{x}} - \mathbf{x})$$

- $J(\boldsymbol{\theta}, \tilde{\mathbf{x}}, y)$: the error of model with parameter $\boldsymbol{\theta}$
- Misclassification $\Leftrightarrow$ Large error

$$\begin{aligned} \underset{\tilde{\mathbf{x}}}{\text{maximize}} \quad & \underbrace{\nabla J(\boldsymbol{\theta}, \mathbf{x}, y)^{\intercal}}_{\mathbf{w}}(\tilde{\mathbf{x}} - \mathbf{x}) + J(\boldsymbol{\theta}, \mathbf{x}, y) \\ \text{subject to} \quad & \|\tilde{\mathbf{x}} - \mathbf{x}\|_{\infty} \leq \epsilon \end{aligned}$$

- Recall $\boldsymbol{\eta} = \tilde{\mathbf{x}} - \mathbf{x}$
- We set $\boldsymbol{\eta} = \epsilon\, \mathrm{sign}(\nabla J(\boldsymbol{\theta}, \mathbf{x}, y))$
- $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon\, \mathrm{sign}(\nabla J(\boldsymbol{\theta}, \mathbf{x}, y))$

FGSM Attack

$$\mathbf{x} = \mathbf{x} + \epsilon \cdot \text{sign}(\nabla_{\mathbf{x}} J(\boldsymbol{\theta}; \mathbf{x}, y))$$

```python
def fgsm_attack(image, epsilon, data_grad):
    # Collect the element-wise sign of the data gradient
    sign_data_grad = data_grad.sign()
    # Create the perturbed image by adjusting each pixel of the input image
    perturbed_image = image + epsilon*sign_data_grad
    # Adding clipping to maintain [0,1] range
    perturbed_image = torch.clamp(perturbed_image, 0, 1)
    # Return the perturbed image
    return perturbed_image

```

Result

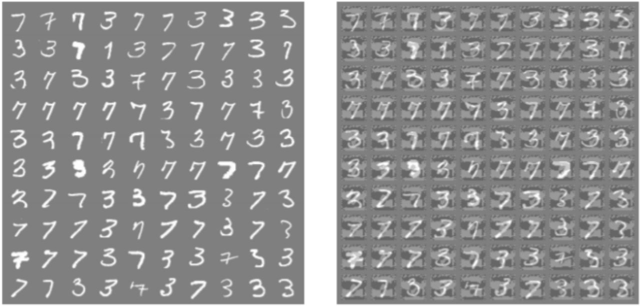


Fig. 7. Left: Original, Right: adversarial examples, Error rate on the origianl data is 1.6% but on the adversarial is 99%.

DeepFool

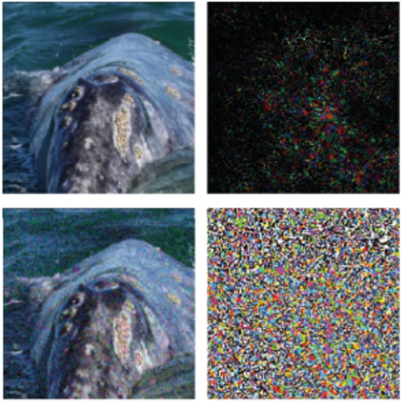


Fig. 8. Whale VS Turtle, $\mathbf{x} + \mathbf{h}$ is classified as “turtle”.

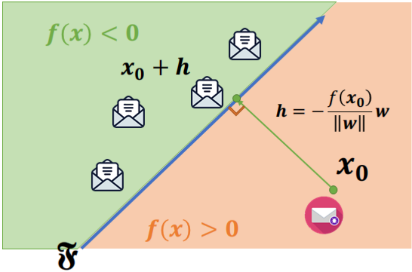
Moosavi-Dezfooli, S.-M., Fawzi, A., Frossard, P.,
DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks
IEEE Conference on Computer Vision and Pattern Recognition, 2016

Minimal Perturabation

Formulation

$$\Delta(\mathbf{x}; f) = \min_{\mathbf{h}} \quad \|\mathbf{h}\|_2$$
$$\text{s.t.} \quad f(\mathbf{x}+\mathbf{h}) \neq f(\mathbf{x})$$

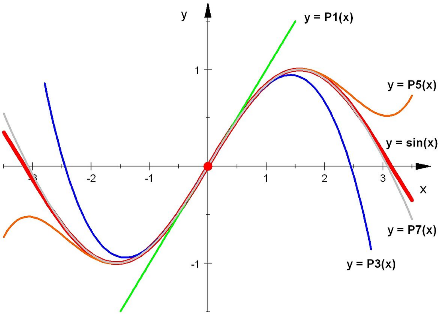
- Suppose a binary classifier $f(\mathbf{x}) = \text{sign}(\mathbf{w}\mathbf{x} + b)$.
- Define $\mathfrak{F} = \{\mathbf{x} \;:\; f(\mathbf{x}) = 0\}$ (what is this set?)



We have a closed form solution.

Setup

- How about *non-linear* general binary classfier $f : \mathbb{R}^n \to \mathbb{R}$?
- Iteratively approximate f with a *linear* function. (How?)
  - Talyer expansion



$$f(x+h) = f(x) + f'(x)h + \frac{1}{2!}f''(x)h^2 + \cdots$$

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \nabla f(\mathbf{x})^\intercal(\mathbf{x}-\mathbf{x}_0) + \frac{1}{2!}(\mathbf{x}-\mathbf{x}_0)^\intercal \nabla^2 f(\mathbf{x})(\mathbf{x}-\mathbf{x}_0) + \cdots$$

DeepFool

At iteration i, we have

- $\mathbf{x}_i$: a data point
- $\mathfrak{F} = \{\mathbf{x} \ : \ f(\mathbf{x}) = 0\}$, f is non-linear
- Approximate $f(\mathbf{x})$ at $\mathbf{x}_i$ (Taylor approximation of order 1)

$$\begin{aligned} f(\mathbf{x}) &\approx f(\mathbf{x}_i) + \nabla f(\mathbf{x}_i)^\intercal (\mathbf{x} - \mathbf{x}_i) \\ &= \underbrace{\nabla f(\mathbf{x}_i)^\intercal}_{\mathbf{w}} \mathbf{x} \underbrace{- \nabla f(\mathbf{x}_i)^\intercal \mathbf{x}_i + f(\mathbf{x}_i)}_{b} = 0 \end{aligned}$$

- Now we can use a *closed form* solution:

$$\mathbf{h} = -\frac{f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|_2^2} \nabla f(\mathbf{x}_i)\,.$$

DeepFool Algorithm

Algorithm 1: DeepFool for binary classifiers

Input: Image $\mathbf{x}$, classfier f

Output: Perturbation $\mathbf{h}$

Initialize $\mathbf{x}_0 \leftarrow \mathbf{x}$, $i \leftarrow 0$

while $\operatorname{sign}(f(\mathbf{x}_i)) = \operatorname{sign}(f(\mathbf{x}_0))$ **do**

$$\mathbf{h}_i \leftarrow -\frac{f(\mathbf{x}_i)}{\|\nabla f(\mathbf{x}_i)\|_2^2} \nabla f(\mathbf{x}_i)$$

$$\mathbf{x}_{i+1} \leftarrow \mathbf{x}_i + \mathbf{h}_i$$

$$i \leftarrow i + 1$$

end

return $\hat{\mathbf{h}} = \sum_i \mathbf{h}_i$

Robustness

Recall

$$\Delta(\mathbf{x}; f) = \min_{\mathbf{h}} \quad \|\mathbf{h}\|_2$$
$$\text{s.t.} \quad f(\mathbf{x}+\mathbf{h}) \neq f(\mathbf{x})$$

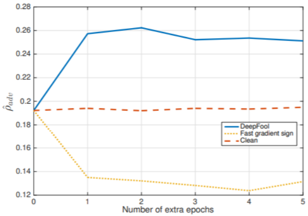
- Measure of *robustness*

$$\rho_{\text{adv}}(f) = \mathbb{E}_x \left[\frac{\Delta(\mathbf{x}; f)}{\|\mathbf{x}\|_2} \right]$$

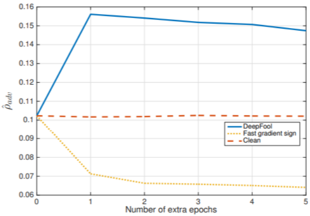
  - Relative magnitude of perturbation to fool the classifier

- Adversarial training
  - generate adversarial examples $\mathbf{x}_{\text{adv}}^1, \mathbf{x}_{\text{adv}}^2, \mathbf{x}_{\text{adv}}^3, \ldots$
  - include them into the training dataset
  - fine-tune f (re-train)

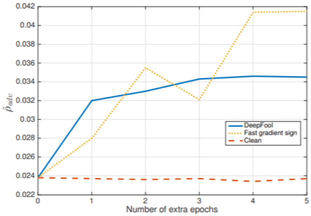
Fine-tuning Networks on Adversarial Examples



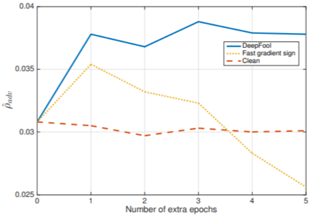
(a) Effect of fine-tuning on adversarial examples computed by two different methods for LeNet on MNIST.



(b) Effect of fine-tuning on adversarial examples computed by two different methods for a fully-connected network on MNIST.



(c) Effect of fine-tuning on adversarial examples computed by two different methods for NIN on CIFAR-10.



(d) Effect of fine-tuning on adversarial examples computed by two different methods for LeNet on CIFAR-10.

Adversarial Attack

- So far we looked at *white-box* attacks
- Adversarial attack methods
  - (gradient-based attacks) gradient $\nabla_x L(\theta, x, y)$
  - (score-based attacks) confidence score $f(x) = \mathbb{P}[Y = k \mid X = x]$
  - (transfer-based attacks) needs a substitute model
  - (decision-based attacks) relying only on the final model decision

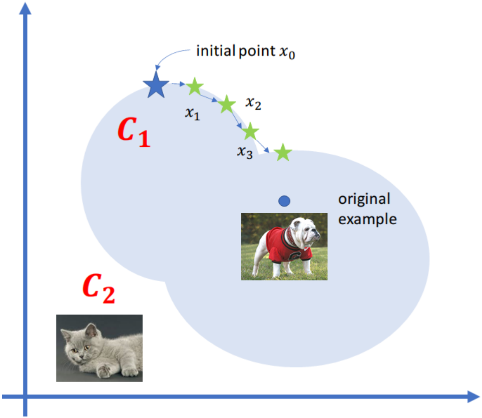


Fig. 9. Random walk along the decision boundary

- Can we find $\tilde{\mathbf{x}}$ such that
  - $\|\mathbf{x} - \tilde{\mathbf{x}}\|$ is small (closeness/minimal perturbation) and
  - $f(\mathbf{x}) \neq f(\tilde{\mathbf{x}})$ (misclassification) ?

BA: initialization

- $\mathbf{x}$: original example (unperturbed)
- $\tilde{\mathbf{x}}$: perturbed (adversarial) example
- We will iteratively generate a sequence of examples:
  - $\tilde{\mathbf{x}}_0, \tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_k$
  - The initial point x_0 needs be *adversarial*.
- Initialization
  - untargeted: each pixel in $\tilde{\mathbf{x}}_0$ is sampled from Uniform $(0, 255)$
  - targeted: need $\tilde{\mathbf{x}}_0$ s.t. $f(\tilde{\mathbf{x}}_0) = y_{\text{target}}$

BA: proposal distribution

We generate $x_0, x_1, \ldots, x_k$ by perturbing the current example.

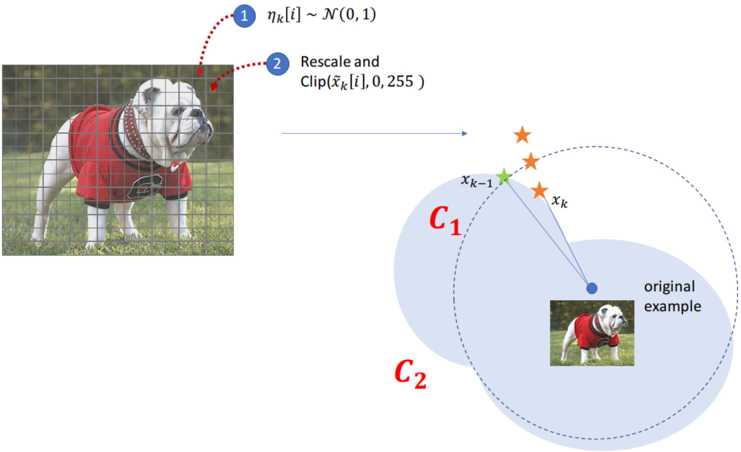
$$x_k[i] = \underbrace{x_{k-1}[i]}_{\text{original pixel}} + \underbrace{\eta_k[i]}_{\text{noise}}, \text{ for } i = 1, \ldots, d,$$

where

- $\eta_k[i] \sim \mathcal{P}$ (noise sampled from distribution $\mathcal{P}$),
- $x_k[i] \in [0, 255]$ (needs to be a *valid* image),
- the *magnitude* of perturbation $\|\boldsymbol{\eta}_k\|_2 = \delta \cdot \mathrm{d}(x, \tilde{x}_k)$, and
- the perturbation reduces the *distance*

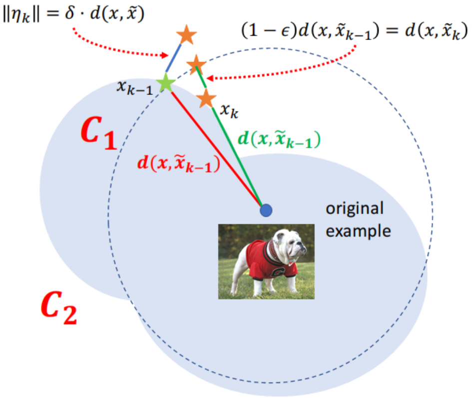
$$\mathrm{d}(\mathbf{x}, \tilde{\mathbf{x}}_{k-1}) - \mathrm{d}(\mathbf{x}, \tilde{\mathbf{x}}_{k-1} + \boldsymbol{\eta}_k) = \epsilon \cdot \mathrm{d}(\mathbf{x}, \tilde{\mathbf{x}}_{k-1}).$$

BA: practical implementation



1. $\tilde{\mathbf{x}}_{k-1}[i] + \boldsymbol{\eta}_k[i]$, where $\boldsymbol{\eta}_k[i] \sim \mathcal{N}(0,1)$
2. Project on the sphere centered at $\mathbf{x}$

Hyperparameters



- $\delta > 0$
- $\epsilon > 0$