Optimization for Deep Learning

Lecture 10-1: Adversarial Learning

Kun Yuan

Peking University

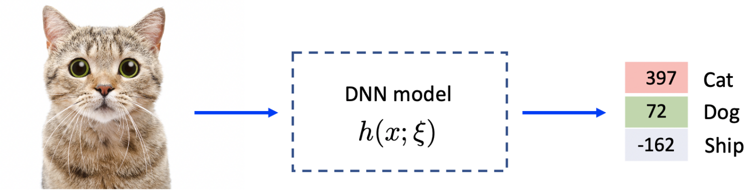
Main contents in this lecture

- Motivation and applications
- Adversarial attack
- Adversarial defense

DNN model

- We model DNN as $h(x;\xi) : \mathbb{R}^d \to \mathbb{R}^c$
  - $x \in \mathbb{R}^d$ is the DNN model parameter to be trained
  - ξ is the input data sample
  - c is the number of classes

- Given the model parameter x, DNN outputs prediction scores $\hat{y}_i$ for input ξ_i



ξ_i

$\hat{y}_i = h(x;\xi_i) \in \mathbb{R}^c$

DNN model training

- Given a dataset $\{\xi_i, y_i\}_{i=1}^m$ where y_i is the ground-truth label for data ξ_i
- Define $L(\hat{y}_i, y_i) = L(h(x; \xi_i), y_i)$ as a loss function to measure the difference/mismatch between predictions and ground-truth labels
- DNN training is to find a model parameter x such that the mismatch (between pred and real) are minimized across the entire dataset:

$$x^\star = \arg\min_{x \in \mathbb{R}^d} \left\{ \frac{1}{m} \sum_{i=1}^{m} L(h(x; \xi_i), y_i) \right\}$$

DNN is fragile to adversarial attacks

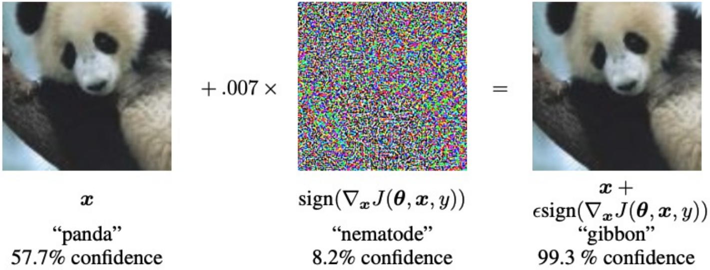


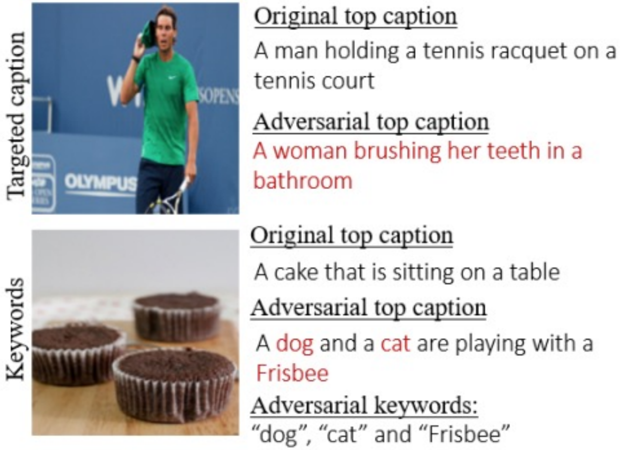
Figure: A demonstration of the adversarial example [Goodfellow et.al., 2015].

Adversarial stop sign

Adversarial T-shirt

Adversarial glass

Adversarial attacks in NLP



Adversarial attacks in NLP

Original Input	Connoisseurs of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: **Positive (77%)**
Adversarial example [Visually similar]	**Aonnoisseurs** of Chinese film will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: **Negative (52%)**
Adversarial example [Semantically similar]	Connoisseurs of Chinese **footage** will be pleased to discover that Tian's meticulous talent has not withered during his enforced hiatus.	Prediction: **Negative (54%)**

How to construct adversarial examples?

- An adversarial example is a perturbation η to maximize misclassification
- Given an input pair (ξ, y), its adversarial example $\eta \in \mathbb{R}^d$ is defined as

$$\eta \in \arg \max_{\eta : \|\eta\| \leq \epsilon} L\Big(h(x^\star, \xi + \eta), y\Big)$$

  where $x^\star$ is the optimal DNN model.

- ℓ_∞-norm is most commonly-used. Hard to perceive.

How to construct adversarial examples?

- An adversarial example is a perturbation η to maximize misclassification

- Given an input pair (ξ, y), its adversarial example $\eta \in \mathbb{R}^d$ is defined as

$$\eta \in \arg \max_{\eta : \|\eta\| \leq \epsilon} L\big(h(x^\star, \xi + \eta), y\big)$$

  where $x^\star$ is the optimal DNN model.

- ℓ_1-norm promotes sparse perturbation. Change a few elements.

Why does adversarial example exists?

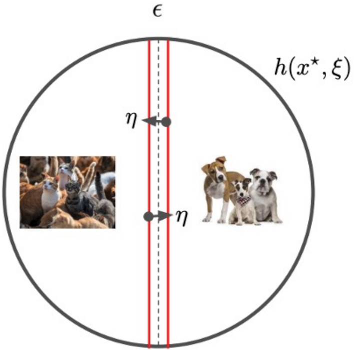


Figure: A small perturbation leads to misclassification.

Algorithm: Fast gradient sign method (FGSM)

- Recall the problem to solve η

$$\eta \in \arg \max_{\eta : \|\eta\|_\infty \le \epsilon} L_y\Big(h_{x^\star}(\xi + \eta)\Big)$$

where $L_y\Big(h_{x^\star}(\xi + \eta)\Big) := L\Big(h(x^\star, \xi + \eta), y\Big)$.

- By linearizion, we have (Goodfellow et al., 2014)

$$\begin{aligned}
\eta^\star &= \arg \max_{\eta : \|\eta\|_\infty \le \epsilon} L_y\Big(h_{x^\star}(\xi + \eta)\Big) \\
&\approx \arg \max_{\eta : \|\eta\|_\infty \le \epsilon} \langle \nabla_\xi \Big[L_y\Big(h_{x^\star}(\xi)\Big)\Big], \eta \rangle \\
&= \epsilon \, \mathsf{sign}\bigg(\nabla_\xi \Big[L_y\Big(h_{x^\star}(\xi)\Big)\Big]\bigg) \\
&= \epsilon \, \mathsf{sign}\bigg(\nabla_h \Big[L_y\Big(h_{x^\star}(\xi)\Big)\Big]^T \nabla_\xi \Big[h_{\boldsymbol{x}^\star}(\xi)\Big]\bigg)
\end{aligned}$$

Results of FGSM on MNIST

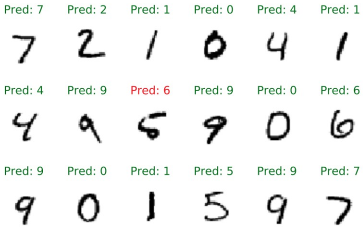


Figure: MNIST images with the predicted digit.

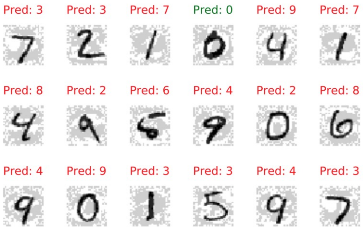
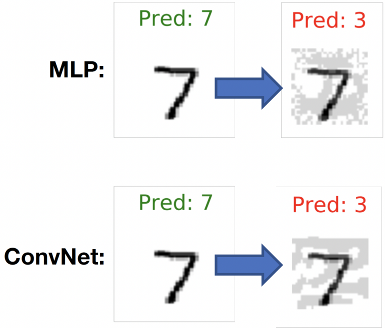
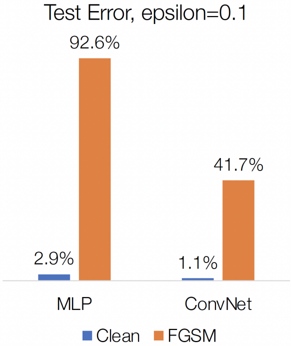


Figure: MNIST images perturbed by a FGSM attack.

Taken from `https://adversarial-ml-tutorial.org/adversarial_examples/`

Results of FGSM on MNIST[1]





[1] This figure is from `https://adversarial-ml-tutorial.org`

Algorithm: Projected gradient descent

- We rewrite the adversarial attack problem into

$$\max_{\eta \in \mathcal{C}} \quad f(\eta), \quad \text{where} \quad f(\eta) = L\Big(h(x^\star, \xi + \eta), y\Big)$$

and $\mathcal{C} = \{\eta : \|\eta\|_\infty \leq \epsilon\}$.

- The projected gradient descent is

$$\eta_{k+1} = \mathsf{Proj}_{\mathcal{C}}\{\eta_k + \gamma \nabla f(\eta_k)\}, \quad k = 0, 1, \cdots$$

where the projection is element-wise clipping between $[-\epsilon, \epsilon]$.

- Result in a more powerful adversarial attack than FGSM; but results in more rounds of update per sample

Algorithm: Projected steepest descent

- We rewrite the adversarial attack problem into

$$\max_{\eta \in \mathcal{C}} \quad f(\eta), \quad \text{where} \quad f(\eta) = L\big(h(x^\star, \xi + \eta), y\big)$$

and $\mathcal{C} = \{\eta : \|\eta\|_\infty \leq \epsilon\}$.

- The projected steepest descent is

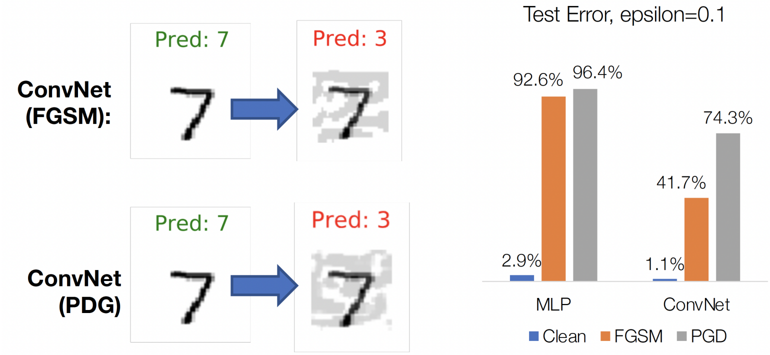
$$g_k = \arg\min_{g:\|g\|_\infty \leq \gamma} \{g^T \nabla f(\eta_k)\}$$

$$\eta_{k+1} = \mathsf{Proj}_{\mathcal{C}}\{\eta_k + g_k\}, \quad k = 0, 1, \cdots$$

- The above recursion can be simplified as (Madry et al., 2017)

$$\eta_{k+1} = \mathsf{Proj}_{\mathcal{C}}\{\eta_k + \gamma \text{sign}(\nabla f(\eta_k))\}, \quad k = 0, 1, \cdots$$

Results of PGD on MNIST[2]



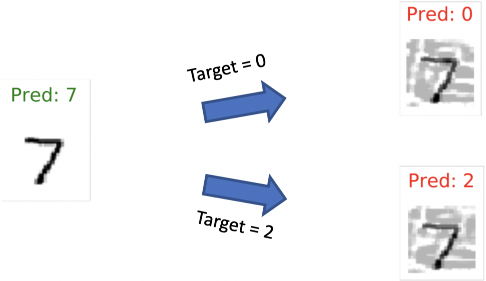
[2] This figure is from `https://adversarial-ml-tutorial.org`

Targeted attack

- It is possible to explicitly change the label to a **particular** class

$$\max_{\eta:\|\eta\|\leq\epsilon} L\big(h(x^\star,\xi+\eta),y\big) - L\big(h(x^\star,\xi+\eta),y_{\text{target}}\big)$$

- An illustration[3]



[3] This figure is from `https://adversarial-ml-tutorial.org`

Adversarial learning

- Adversarial machine learning is to make models robust to attacks

$$\min_{x \in \mathbb{R}^d} \quad \frac{1}{m} \sum_{i=1}^{m} f_i(x) \quad \text{where} \quad f_i(x) = \max_{\eta : \|\eta\|_\infty \le \epsilon} L(h(x; \xi_i + \eta), y_i)$$

- We maximize η to construct adversarial examples but minimize x to construct robust machine learning models; minimax optimization!

- How to compute the gradient is a great challenge

$$\nabla_x f_i(x) = \nabla_x \Big(\max_{\eta : \|\eta\|_\infty \le \epsilon} L(h(x; \xi_i + \eta), y_i) \Big).$$

It involves differentiating with respect to a maximization

Danskin's theorem

Theorem 1

Let $\eta^\star(x)$ be the optimal (maximized) adversarial example given x, then

$$\nabla_x \Big(L(h(x; \xi_i + \eta^\star(x)), y_i) \Big)$$

is a descent direction of $f_i(x)$.

Therefore, an intuitive way to find $\nabla_x f_i(x)$ is

- Find $\eta^\star(x) \in \max_{\eta : \|\eta\|_\infty \leq \epsilon} \{L(h(x; \xi_i + \eta), y_i)\}$ (see previous slides)
- Compute $\nabla_x f_i(x) \approx \nabla_x \Big(L(h(x; \xi_i + \eta^\star(x)), y_i) \Big)$

Adversarial learning algorithm

Algorithm 1: Adversarial learning

Input: Learning rate γ, iterations K, batch size B

Initialization: Initialize neural network parameter x_0

for $k = 0, 1, \cdots, K$ **do**

  Initialize gradient $g_k = 0$;

  Select a mini-batch of data $\mathcal{B} \subseteq \{1, \cdots, m\}$;

  for $i \in \mathcal{B}$ **do**

    Find an attack $\eta^\star$ by solving

    $\eta^\star(x) \in \max_{\eta : \|\eta\|_\infty \le \epsilon} \{L(h(x; \xi_i + \eta), y_i)\}$;

    Accumulate gradient

    $g_k = g_k + \nabla_x \Big(L(h(x_k; \xi_i + \eta^\star(x)), y_i) \Big)$

  Update model by $x_{k+1} = x_k - \frac{\gamma}{B} g_k$

One can use FGSM or PGD to achieve the attack $\eta^\star(x)$

Adversarial learning with PGD oracle

For simplicity we set $B = 1$

Algorithm 2: Adversarial learning with PGD oracle (Madry et al., 2017)

Input: Learning rate γ, outer iterations K, inner iterations M

Initialization: Initialize neural network parameter x_0

for $k = 0, 1, \cdots, K$ **do**

  Initialize adversarial attack $\eta_0 = 0$;

  Sample a random data (ξ_{i_k}, y_{i_k}) ;

  for $j = 0, 1, \cdots, M-1$ **do**

    $\delta = \eta_j + \alpha \text{sign}[\nabla_\eta L(h(x_k; \xi_{i_k} + \eta_j), y_{i_k})]$

    $\eta_{j+1} = \max(\min(\delta, \epsilon), -\epsilon)$

  Update model by $x_{k+1} = x_k - \gamma \nabla_x L(h(x_k; \xi_{i_k} + \eta_M), y_{i_k})$

Strong defense performance, but very expensive due to the inner loop; incurs additional KM evaluations of the gradient

"Free" Adversarial learning with PGD oracle

This algorithm is from (Shafahi et al., 2019). We omit subscript for simplicity.

Algorithm 3: "Free" Adversarial learning with PGD oracle

Input: Learning rate γ, outer iterations K, inner iterations M

Initialization: Initialize neural network parameter x and $\eta = 0$

for $k = 0, 1, \cdots, K/M$ **do**

  Sample a random data (ξ, y) ;

  for $j = 0, 1, \cdots, M-1$ **do**

    $\nabla_x, \nabla_\eta = \nabla_x L(h(x; \xi + \eta), y), \nabla_\eta L(h(x; \xi + \eta), y)$

    $\delta = \eta + \alpha \text{sign}(\nabla_\eta)$

    $\eta = \max(\min(\delta, \epsilon), -\epsilon)$

    $x = x - \gamma \nabla_x$

Simultaneously update η and x; one backward results in two gradients; incurs **no** additional evaluations of the gradient; much more efficient

Adversarial learning with FGSM oracle

"Free" adversarial learning only use K/M samples; Generally speaking, utilization of more samples will lead to better performance

Algorithm 4: Adversarial learning with FGSM oracle (Wong et al., 2020)

Input: Learning rate γ, outer iterations K, inner iterations M

Initialization: Initialize neural network parameter x and $\eta = 0$

for $k = 0, 1, \cdots, K$ **do**

  Sample a random data (ξ, y) ;

  Initialize $\eta = \mathrm{Uniform}(-\epsilon, \epsilon)$

  $\delta = \eta + \alpha \mathrm{sign}(\nabla_\eta L(h(x; \xi + \eta), y))$

  $\eta = \max(\min(\delta, \epsilon), -\epsilon)$

  $x = x - \gamma \nabla_x L(h(x; \xi + \eta), y)$

Sample K random data; converge much faster (in iterations) than "Free" adversarial learning; incurs additional K evaluations of gradient; less efficient than "Free" adversarial learning per iteration

Fast is better than free!

Method	Standard accuracy	PGD ($\epsilon = 8/255$)	Time (min)
FGSM + DAWNBench			
+ zero init	85.18%	0.00%	12.37
+ early stopping	71.14%	38.86%	7.89
+ previous init	86.02%	42.37%	12.21
+ random init	85.32%	44.01%	12.33
+ $\alpha = 10/255$ step size	83.81%	46.06%	12.17
+ $\alpha = 16/255$ step size	86.05%	0.00%	12.06
+ early stopping	70.93%	40.38%	8.81
"Free" ($m = 8$) (Shafahi et al., 2019)[1]	85.96%	46.33%	785
+ DAWNBench	78.38%	46.18%	20.91
PGD-7 (Madry et al., 2017)[2]	87.30%	45.80%	4965.71
+ DAWNBench	82.46%	50.69%	68.8

Fast is better than free!

Table 3: Time to train a robust CIFAR10 classifier to 45% robust accuracy using various adversarial training methods with the DAWNBench techniques of cyclic learning rates and mixed-precision arithmetic, showing significant speedups for all forms of adversarial training.

Method	Epochs	Seconds/epoch	Total time (minutes)
DAWNBench + PGD-7	10	104.94	17.49
DAWNBench + Free ($m = 8$)	80	13.08	17.44
DAWNBench + FGSM	15	25.36	6.34
PGD-7 (Madry et al., 2017)[5]	205	1456.22	4965.71
Free ($m = 8$) (Shafahi et al., 2019)[6]	205	197.77	674.39

Summary

- DNN is fragile; not robust to adversarial attacks
- Adversarial attack approaches: FGSM/PGD/PSD
- Adversarial learning promotes robustness
- Fast is better than Free

References I

I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

A. Shafahi, M. Najibi, M. A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" *Advances in Neural Information Processing Systems*, vol. 32, 2019.

E. Wong, L. Rice, and J. Z. Kolter, "Fast is better than free: Revisiting adversarial training," *arXiv preprint arXiv:2001.03994*, 2020.