50.039 – Theory and Practice of Deep Learning

Alex

Week 06: Attacks 2

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

Key content

- classification of attacks: white box vs black box attacks
- classification of attacks: targeted vs untargeted attacks
- gradient and gradient sign attacks against white boxes
- surrogate attacks against black boxes (be able to explain the rough idea how it works)
- boundary attacks against black boxes / and against saturation defenses (be able to explain the rough idea how it works, no need to be able to reproduce all formulas)
- understand that messing up the gradient of your predictor does not provide defense against attacks – see boundary attack (and surrogate attacks) as counterexamples.
- `https://aaai18adversarial.github.io/`

1 Attacks

1.1 two dimensions to classify attacks

white box vs black box

- White box attacks: one has access to the prediction model as a whole. Can be quite realistic if one can guess... (e.g. github of collaborators)
- Black box: one has access to the outputs of prediction models only.

targeted vs untargeted attacks

- targeted: sample x to be classified into a certain fixed class label. You want a cat image to be classified as a fridge ... or a fridge image to be misclassified as a cat.
- untargeted: sample x to be misclassified relative to its original label.

Most white box attacks follow two ideas:

- compute a gradient
  - either of the training loss function with one hot labels for the ground truth class (gradient descent), or for a target class (gradient ascent)
  - or of the logits of the neural network, not of the softmax (softmax saturates like a sigmoid), again for the ground truth class or for the target class

1.2 gradient and gradient sign attacks against white boxes

targeted Iterative gradient:

key idea for graded knowledge

simple gradient for a fixed target class c: white box, targeted. Iterate until class c is predicted.

$$x_{n+1} = x_n + \epsilon \frac{\partial f_c}{\partial x}(x_n)$$

This is gradient ascent towards target class.
Important: it is much better if one **uses logits** and not the softmax probabilities.

The softmax will be problematic if $f_c(x)$ is close to one, because then gradients are near zero, while in later stages with larger gradients one needs much smaller step sizes.

untargeted Iterative gradient:

key idea for graded knowledge

white box, untargeted. Iterate until prediction switches from the original class either

$$c^0 = \text{argmax}_c f_c(x_0)$$
$$x_{n+1} = x_n - \epsilon \frac{\partial f_{c^0}}{\partial x}(x_n)$$

(This is gradient descent away from target class.)

or (Let c^* be the least probable class):

$$c^* = \text{argmin}_c f_c(x_n)$$
$$x_{n+1} = x_n + \epsilon \frac{\partial f_{c^*}}{\partial x}(x_n)$$

Use logits.

(This is gradient ascent towards the least probable class. Points towards low probability/Outlier regions?)

The prediction switches when the current label is no the original label of image x_0, that is: $\text{argmax}_c f_c(x_n) \neq \text{argmax}_c f_c(x_0)$. Note the sign difference to the targeted case.

Fast gradient sign: `https://arxiv.org/pdf/1412.6572.pdf`:

$L(c, f(x_n))$ is the loss of predictor f for the class label c. In case of cross-entropy loss

$$L(c, f(x)) = -\log p(Y = c|X = x)$$

key idea for graded knowledge

Let $c^=$ be the true class label. Then the fast gradient sign is based on the sign of the gradient of the training loss:

$$x_{n+1} = x_n - \epsilon \text{sign}(\frac{\partial L(c^=, f)}{\partial x}(x_n))$$

ϵ chosen so large that 1-2 iterations are sufficient. The sign is applied for each dimension.

Why sign ? Yes it is ℓ_∞-norm constrained, but it is not a sound explanation to me. Figure 1 in `https://arxiv.org/pdf/1611.02770.pdf` on a complex dataset, ImageNet, suggests that fast sign is not good actually - but fast. One possible advantage: distorting the gradient by taking the sign may help to avoid

getting stuck in local extrema, thus increasing attack success rates at the cost of decreased quality.

This is white box, untargeted. Fast – bcs one aims usually at 1-2 iterations, but often coarse images as results.

- pro: fast
- con: coarse images
- for better results use iterative methods

Iterative gradient sign: `https://arxiv.org/pdf/1607.02533.pdf`

Variant1:

$$x_{n+1} = \text{clipto}(\gamma)(x_n - \epsilon \text{sign}(\frac{\partial L(c^=, f)}{\partial x}(x_n)))$$

– walk opposite to the direction that increases the current class label as measured by the training loss. clip changes to γ relative to original image.

Variant2:

$$x_{n+1} = \text{clipto}(\gamma)(x_n + \epsilon \text{sign}(\frac{\partial L(c^*, f)}{\partial x}(x_n)))$$

where $c^*(x) = \text{argmin}_c p(Y = c|X = x)$ is the least likely prediction class of image x – walk in the direction of the least likely class – converges faster than variant 1, see Fig2 in `https://arxiv.org/pdf/1607.02533.pdf`. This method works on non-trivial datasets!

Note the sign difference between variant 1 and variant 2.

clipto(γ) clips the changes of an image in every dimension to at most γ difference to the original/ source image x_0.

$$\gamma > 0$$
$$clip0(\gamma)(x) = \max(0, x_0 - \gamma, x) \text{clipto}(\gamma)(x) = \min(255, x_0 + \gamma, clip0(\gamma)(x))$$

This is white box, untargeted.

1.3 defensive distillation

Idea: softmax with a temperature T

$$\text{softmax}(x, T)[i] = \frac{e^{x_i/T}}{\sum_k e^{x_k/T}}$$

$T \to \infty$: goes to uniform

$T \to 0$: converges to a one hot-vector at the maximum dimension

rough idea: train at temperature $T >> 1$. at test time: use temperature $T = 1$, then at test time prediction probabilities get much closer to 0 and 1 – saturation range. In this range gradients become near zero, so that methods which use gradients fail. in detail:

- train a teacher network at $T >> 1$, e.g. $T = 100$
- collect prediction probilities $p(x)$ for all training samples x
- train a second network at the same large $T >> 1$ using $p(x)$ as soft labels instead of 0/1 hard labels from the original dataset.
- at test time use the second network with $T = 1$ – then most predictions have probabilities near 0/1, thus gradients are almost zero ands numerically instable.

1.4 use logits to avoid problems with defensive distillation if possible

Carlini & Wagner: `https://arxiv.org/pdf/1608.04644.pdf`, `https://arxiv.org/pdf/1705.07263.pdf` This is white box, targeted.

key idea for graded knowledge

- Goal: synth an image x which is close to a target image x_0 and which is classified as target class t
- optimize an objective of two components,
- first component $\|x - x_0\|^2$ makes synth an image x which is close to a target image x_0
- second component measures whether target class has highest score already
- use logit outputs of the classifier
- a way tavoid defensive distillation, but is white box

Given a target class t, and let $f_c(x)$ be the logits output of a classifier for class c (not the softmax!). Solve the following optimization problem

$$\min_x \|x - x_0\|^2 + c \max(d(x), 0)$$
$$d(x) = \max_{c \neq t} f_c(x) - f_t(x)$$

Why this objective?

- $\|x - x_0\|^2$ makes synth an image x which is close to a target image x_0
- understand: $d(x) > 0$ means: $f_t(x) < \max_{c \neq t} f_c(x)$, which means that the prediction has not reached the target class t yet.
- cap $d(x)$ at zero, so that one does not minimize the objective by making the prediction of $f_t(x)$ very large ($d(x)$ very negative) while wandering far away from the start image x_0.

Important: it uses logits (the output of the last layer), not the softmax probabilities. Advantage: no problems with softmax saturation, no protection by defensive distillation. But: one cannot always access the logits!

1.5 making defensive distillation fail with better attacks I: Surrogate attacks against black boxes

`https://arxiv.org/pdf/1602.02697.pdf`
Setting: can observe only output $f(x)$ of target classifier.

key idea for graded knowledge

- train a replacement $g(x)$ which mimicks $f(x)$ and after that white-box attack the replacement $g(x)$.
- iterate training. at every step increase the training data set by augmenting the existing samples along the gradient of the surrogate

Some finer details:

- iterate the following training. At each step r train a surrogate g mimicking f
  - input many samples x into f as queries, collect probability labels $f(x)$
  - train g using a dataset S_r of $(x, f(x))$

    
  - note: can use cross-entropy with soft labels as well

    $$L(g(x), f) = \sum_c -f_c \log g_c(x)$$

    Cross-entropy works NOT ONLY with one hot labels!
  - enlarge existing training dataset by augmentation: $S_{r+1} = S_r \cup \{x + \lambda \text{sign}(\sum_c f_c(x) \nabla g_c(x)), x \in S_r\}$

    

    Here one uses only the gradient of the surrogate.
- perform white box attacks on your trained surrogate $g(x)$

- uses a slightly different adversarial attack for the white box.

  Compute a saliency score for a target class t and every dimension i of a sample (image) x

  $$S(x,t)[i] = \begin{cases} 0 & \text{if } \frac{\partial f_t}{\partial x_i} < 0 \text{ or } \sum_{c:c\neq t} \frac{\partial f_c}{\partial x_i} > 0 \\ \frac{\partial f_t}{\partial x_i} \left| \sum_{c:c\neq t} \frac{\partial f_c}{\partial x_i} \right| & \text{else} \end{cases}$$

  idea: a dimension i of an input x is salient if either the gradient for the target class is positive or the sum of gradients along all other classes is negative.

  sparse selection of dimensions – may help to suppress noise from dimensions where the gradient of the surrogate does not match the gradient of the target due to a falsely learnt surrogate.

No results on larger networks known.

1.6 making defensive distillation fail with better attacks II: Boundary attacks against black boxes

`https://openreview.net/pdf?id=SyZI0GWCZ`

key idea for graded knowledge

- input is target image. output: synth image
- synth image should look like target image, but have differing or wrong prediction (see examples in the paper)
- idea: move along decision boundary to the predicted class of target image, but always on the wrong side
- feel your way in mole-style until synth image looks close to target image but has wrong prediction
- how to feel you way ? do not use gradients. Sample perturbations from a distribution. Accept perturbed image if it is adversarial.
- for accept/ reject one needs only the predicted label, no probabilities. The box is as black is it can get for this attack.
- approach has iteration of two steps to get close to target image.
- Result: adversarial image which is close to the start image.
- See Fig 4 and Fig 7 in the paper.

Data: original image $\mathbf{o}$, adversarial criterion $c(.)$, decision of model $d(.)$
Result: adversarial example $\tilde{\boldsymbol{o}}$ such that the distance $d(\boldsymbol{o}, \tilde{\boldsymbol{o}}) = \|\boldsymbol{o} - \tilde{\boldsymbol{o}}\|_2^2$ is minimized
initialization: $k = 0$, $\tilde{\boldsymbol{o}}^0 \sim \mathcal{U}(0, 1)$ s.t. $\tilde{\boldsymbol{o}}^0$ is adversarial;
while $k <$ *maximum number of steps* **do**
  draw random perturbation from proposal distribution $\boldsymbol{\eta}_k \sim \mathcal{P}(\tilde{\boldsymbol{o}}^{k-1})$;
  if $\tilde{\boldsymbol{o}}^{k-1} + \boldsymbol{\eta}_k$ *is adversarial* **then**
    set $\tilde{\boldsymbol{o}}^k = \tilde{\boldsymbol{o}}^{k-1} + \boldsymbol{\eta}_k$;
  else
    set $\tilde{\boldsymbol{o}}^k = \tilde{\boldsymbol{o}}^{k-1}$;
  end
  $k = k + 1$
end

Algorithm 1: Minimal version of the Boundary Attack.

Has, parameters δ, ϵ. How to sample η_k for proposal testing ? Let o be the target image. o^{k-1} is your current synthetsized adversarial image.

1. sample N different η^k from a gaussian distribution
2. rescale each η^k it so that the following two conditions hold
   - $o_i^{k-1} + \eta_i^k \in [0, 255]$ for every dimension i
   - the perturbation has a small relative size δ relative to the distance between the start image o and the current attack image o^{k-1} at step $k-1$:

$$\|\eta^k\|_2 \leq \delta d(o, o^{k-1})$$

3. project each of the rescaled η^k onto a sphere around o so that the following holds:
   - $o_i^{k-1} + \eta_i^k \in [0, 255]$ for every dimension i
   - $d(o, o^{k-1} + \eta_k) = d(o, o^{k-1})$ – can be achieved for a euclidean norm by solving for α:

$$\begin{aligned} v(\alpha) &= \alpha(o^{k-1} + \eta_k) + (1-\alpha)o \text{ such that} \\ \|v(\alpha) - o\| &= \|o^{k-1} - o\| \\ \Leftrightarrow \alpha\|o^{k-1} + \eta_k - o\| &= \|o^{k-1} - o\| \\ \Leftrightarrow \alpha &= \frac{\|o^{k-1} - o\|}{\|o^{k-1} + \eta_k - o\|} \\ then : \eta^{k,mod} &:= v(\alpha) - o^{k-1} \end{aligned}$$

   Obviously this does the job because

$$d(o, o^{k-1} + \eta^{k,mod}) = d(o, o^{k-1} + v(\alpha) - o^{k-1}) = d(o, v(\alpha)) = \|v(\alpha) - o\| = \|o^{k-1} - o\|$$

4. check among the N $\eta^{k,mod}$ whether your δ is good:
   - how many of the N $o^{k-1} + \eta^{k,mod}$ are misclassified (that is, their label is not equal to the label of o)? you want this to be in a region around $0.5N$, so:
   - if it is less than $0.3N$, then reduce delta by a factor of say 0.9, restart at step 1.

- if it is more than $0.7N$, then increase delta by a factor of say $1.0/0.9 > 1$, restart at step 1.

5. choose one of those $\eta^{k,mod}$, such that $o^{k-1} + \eta^{k,mod}$ is still adversarial, make a step towards original image o so that
   - $o_i^{k-1} + \eta_i^{k,mod2} \in [0, 255]$ for every dimension i
   - $d(o, o^{k-1}+\eta^{k,mod2}) = d(o, o^{k-1})(1-\epsilon)$ – the new image $o^{k-1}+\eta^{k,mod2}$ moves closer to the original image o by $(1-\epsilon)$ compared to the previous step o^{k-1}
6. is $o^{k-1} + \eta^{k,mod2}$ are adversarial ? if no, then decrease ϵ by a factor, if yes, then increase ϵ by a factor (= take a step closer), repeat a few times to get a $o^{k-1} + \eta^{k,mod2}$ which is close to the boundary.

- con: rejection sampling = many model calls = slowish!
- pro: black box, you need only the predicted label
- pro: methods which try to defend by hide/shatter gradients are ineffective against this type of attack

1.7 on transferability of attacks: outside of quizzes / exams

`https://arxiv.org/pdf/1611.02770.pdf`

non-targeted attacks generalize better across different models than targeted ones.

Chapter 5, table 4: **non-targeted ensemble attacks.** It helps to optimize against an ensemble of different classifiers to get strong attacks that are valid for many architectures.

2 Some papers: outside of quizzes / exams

`https://arxiv.org/pdf/1511.04599.pdf` – deepfool: whitebox, untargeted, works on Imagenet
`http://www.evolvingai.org/fooling` Fooling as art
`https://arxiv.org/pdf/1707.08945.pdf` – design attacks on images that look like physical-world plausible changes. hide them or make them look natural. Adversarial stickers for objects :-D .
`https://arxiv.org/pdf/1707.07397.pdf` 3d print adversarial objects.
`https://arxiv.org/pdf/1705.07263.pdf` detection is hard

3 In class: an untargeted attack from nothing installed state in 10 minutes – foolbox

- install via pip `https://github.com/bethgelab/foolbox` (in your virtual environment)
- check the example `https://foolbox.readthedocs.io/en/latest/user/examples.html#creating-an-untargeted-adversarial-for-a-pytorch-model`
- modify it a little bit to see the results:
  - display the adversarial image
  - display the difference between the images, unscaled (only move it into the center of $[0, 1]$ or $[0, 255]$, and divide by 2, so that the difference fits in) – this is the difference on the natural scale
  - display the difference between the images, scaled so that it becomes visible (so that the min value lies at the lower value of, and the max value $[0, 1]$ or $[0, 255]$)
  - print the mean, median and max difference over all rgb subpixels
- feel free to play with other images and attacks