A short introduction to DeepFool

fm-seminar

Daniel Bakkelund

Link to article: https://arxiv.org/abs/1511.04599

12th June, 2018

Overview

Prelude

Prelude

Disclaimer:
The following is not proven to be an NN instability issue.

https://translate.google.com/#no/en/skogssvale%0Astorting

DeepFool

Article metadata

Arxiv

- Link to article: https://arxiv.org/abs/1511.04599
- Submitted on 14 Nov 2015 (v1), last revised 4 Jul 2016 (this version, v3)
- In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016

Authors

- Seyed-Mohsen Moosavi-Dezfooli
- Alhussein Fawzi
- Pascal Frossard

École Polytechnique Fédérale de Lausanne

Problem statement

Given a classification function

$$f : \mathbb{R}^n \to \{1, \ldots, c\},$$

a neural network provides an approximation

$$\hat{k} : \mathbb{R}^n \to \{1, \ldots, c\}$$

which is known to work well on a test set $\mathcal{D} \subset \mathbb{R}^n$, i.e.

$$f|_{\mathcal{D}} = \hat{k}.$$

Question

Given $x_0 \in \mathcal{D}$, what is the vector $r_0 \in \mathbb{R}^n$ of lowest norm (i.e. "shortest length") s.t.

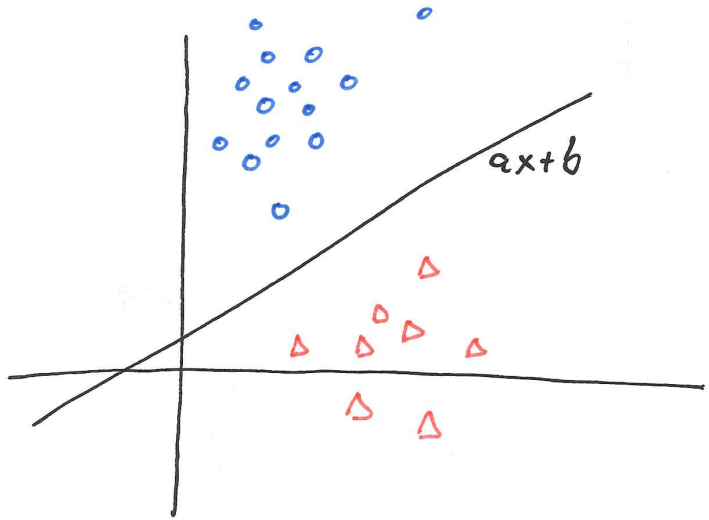
$$\hat{k}(x_0) \neq \hat{k}(x_0 + r_0) \quad ?$$

Motivating example I

Let $g_0(x) = ax + b$ be an affine function separating two sets of points $A, B \subset \mathbb{R}^2$, and let the binary classifier $\hat{g} : \mathbb{R}^2 \to \mathbb{R}$ be defined as

$$\hat{g}\big((x, y)\big) \;=\; g_0(x) - y.$$

Then $\hat{g}(x, y) > 0$ indicates $(x, y) \in A$, and vice versa.

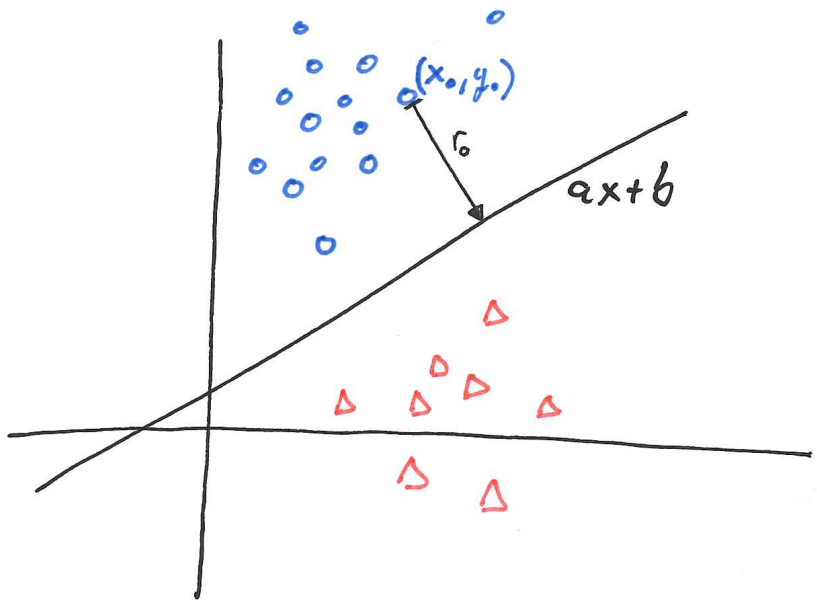


Given $(x_0, y_0) \in \mathcal{D}$, defining $r_0 \in \mathbb{R}^2$ as

$$r_0 \;=\; \frac{ax_0 + b - y_0}{a^2 + 1} \cdot (-a, 1) \;=\; -\frac{\hat{g}\big((x_0, y_0)\big)}{\|\nabla \hat{g}\|_2^2} \nabla \hat{g}$$

for any $\eta > 1$, we have that

$$\hat{k}\big((x_0, y_0) + \eta r_0\big) \;\neq\; \hat{k}\big((x_0, y_0)\big).$$



Motivating example II

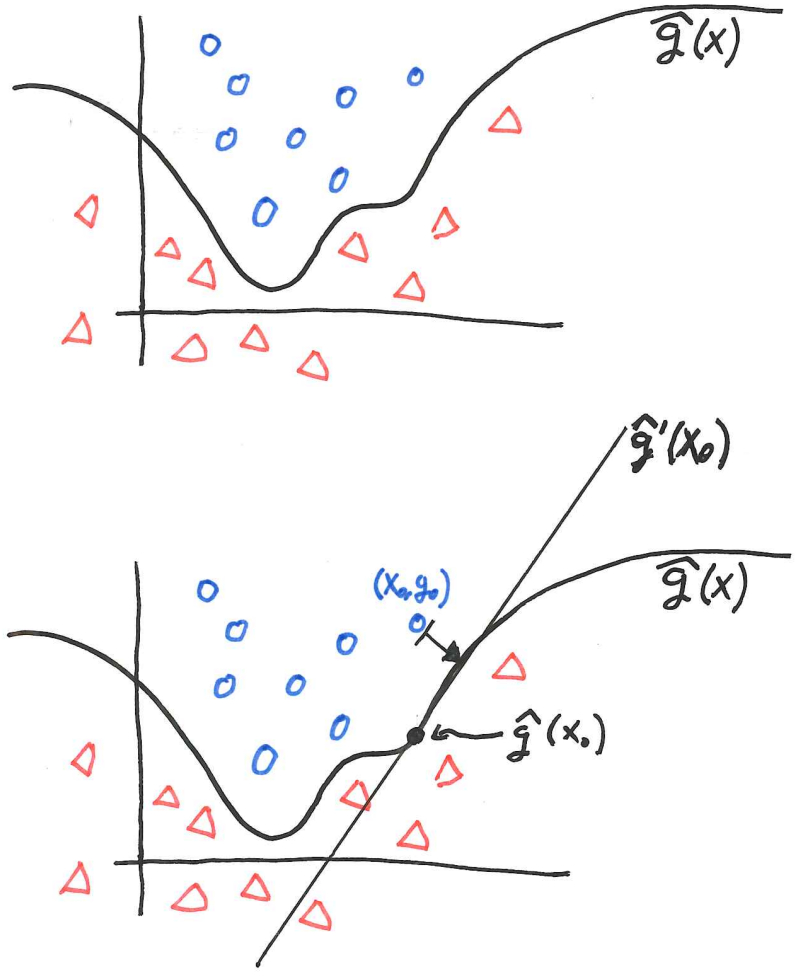
When $\hat{g}$ is non-linear, we use the gradient $\nabla \hat{g}|_{(x_0,y_0)}$ as a local, linear estimate for the perturbation r_0.

If $\hat{k}((x_0, y_0) + \eta r_0) = \hat{k}((x_0, y_0))$, we repeat the procedure, now starting at

$$(x_1, y_1) \;=\; (x_0, y_0) + \eta r_0.$$

The final perturbation, after changing class, becomes $r = \sum_i r_i$.

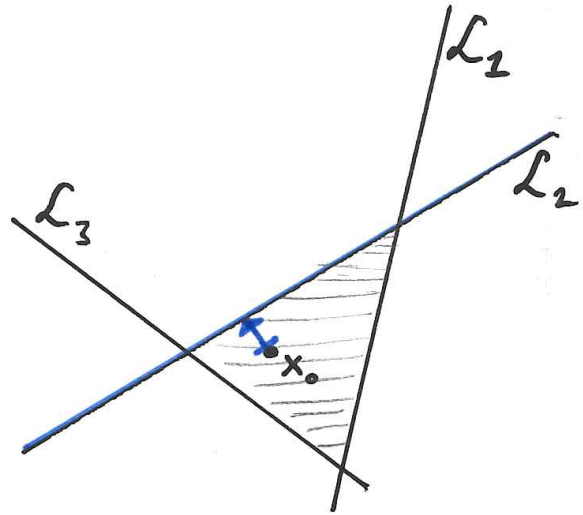
Notice that this is not necessarily a global optimum. The algorithm may run stuck in a local minimum, just as for Newton's algorithm.



Motivating example III

The multiclass classifier $\mathbb{R}^n \to \{1, \ldots, c\}$ can be thought of as a function $\hat{g} : \mathbb{R}^n \to \mathbb{R}^c$ composed with the function picking the index of the highest valued component, i.e.

$$\hat{k}(x) \;=\; \arg\max_i \{\hat{g}_i(x)\}.$$



For an affine multiclass classifier

$$\hat{g}(x) \;=\; Wx + b, \qquad (W \in \mathbb{R}^c \times \mathbb{R}^n)$$

this partitions $\mathbb{R}^n$ into a set of convex regions separated by $(n-1)$-dimensional hyperplanes.

Changing class amounts to finding the smallest vector r_0 that makes us cross one of these separating planes.

The solution is to iterate all the components (i.e. class boundaries), and find the nearest one:

$$\hat{\ell}(x_0) \;=\; \arg\min_{k \neq \hat{k}(x_0)} \frac{\left|\hat{g}_k(x_0) - \hat{g}_{\hat{k}(x_0)}(x_0)\right|}{\left\|w_k - w_{\hat{k}(x_0)}\right\|},$$

and use this to compute the perturbation vector as in the binary case.

The non-linear multiclass classifier is then just a local linearisation, using the gradient of the neural network as the linear representation.

Empirical results I

Comparison between DeepFool, fast gradient sign ([4]) and the method from [18].

$$\hat{\rho}_{\text{adv}}(\hat{g}) = \frac{1}{\mathcal{D}} \sum_{x \in \mathcal{D}} \frac{\|r(x)\|_2}{\|x\|_2}.$$

Classifier	**Test error**	$\hat{\rho}_{\text{adv}}$ [DeepFool]	time	$\hat{\rho}_{\text{adv}}$ [4]	time	$\hat{\rho}_{\text{adv}}$ [18]	time
LeNet (MNIST)	1%	2.0×10^{-1}	110 ms	1.0	20 ms	2.5×10^{-1}	> 4 s
FC500-150-10 (MNIST)	1.7%	1.1×10^{-1}	50 ms	3.9×10^{-1}	10 ms	1.2×10^{-1}	> 2 s
NIN (CIFAR-10)	11.5%	2.3×10^{-2}	1100 ms	1.2×10^{-1}	180 ms	2.4×10^{-2}	>50 s
LeNet (CIFAR-10)	22.6%	3.0×10^{-2}	220 ms	1.3×10^{-1}	50 ms	3.9×10^{-2}	>7 s
CaffeNet (ILSVRC2012)	42.6%	2.7×10^{-3}	510 ms*	3.5×10^{-2}	50 ms*	-	-
GoogLeNet (ILSVRC2012)	31.3%	1.9×10^{-3}	800 ms*	4.7×10^{-2}	80 ms*	-	-

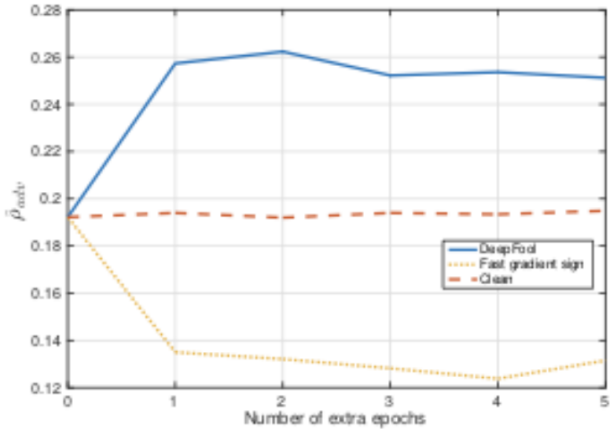
Classifier	DeepFool	Fast gradient sign
LeNet (MNIST)	0.10	0.26
FC500-150-10 (MNIST)	0.04	0.11
NIN (CIFAR-10)	0.008	0.024
LeNet (CIFAR-10)	0.015	0.028

Table 2: Values of $\hat{\rho}_{\text{adv}}^{\infty}$ for four different networks based on DeepFool (smallest l_∞ perturbation) and fast gradient sign method with 90% of misclassification.

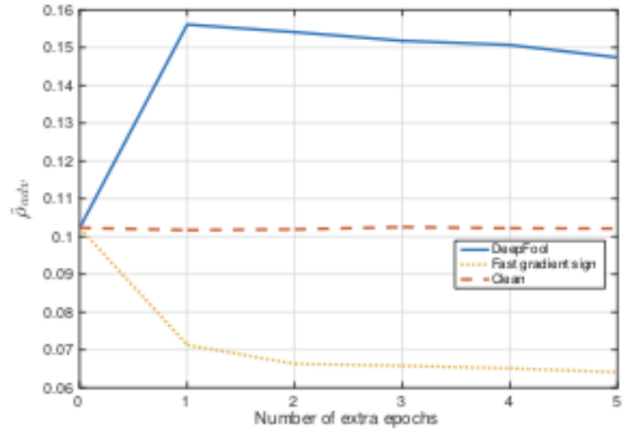
(Mostly for comparison to other solutions)

Empirical results II

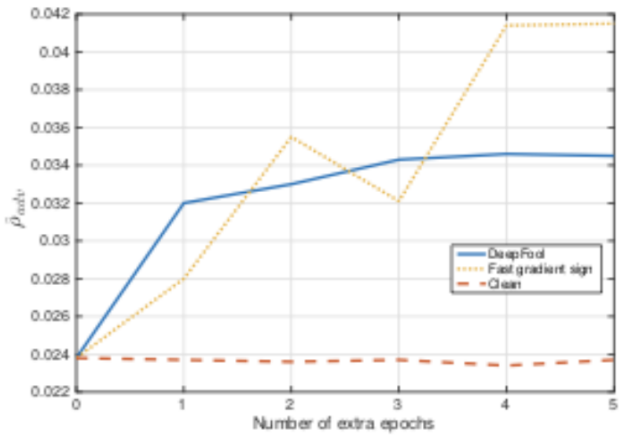
Effect of re-training the NN using adversarially perturbed images



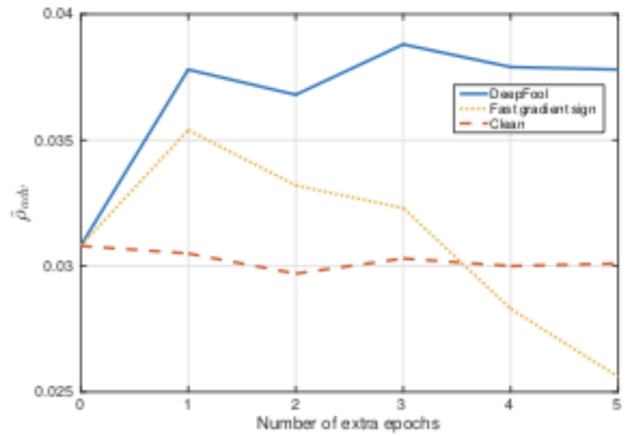
(a) Effect of fine-tuning on adversarial examples computed by two different methods for LeNet on MNIST.



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



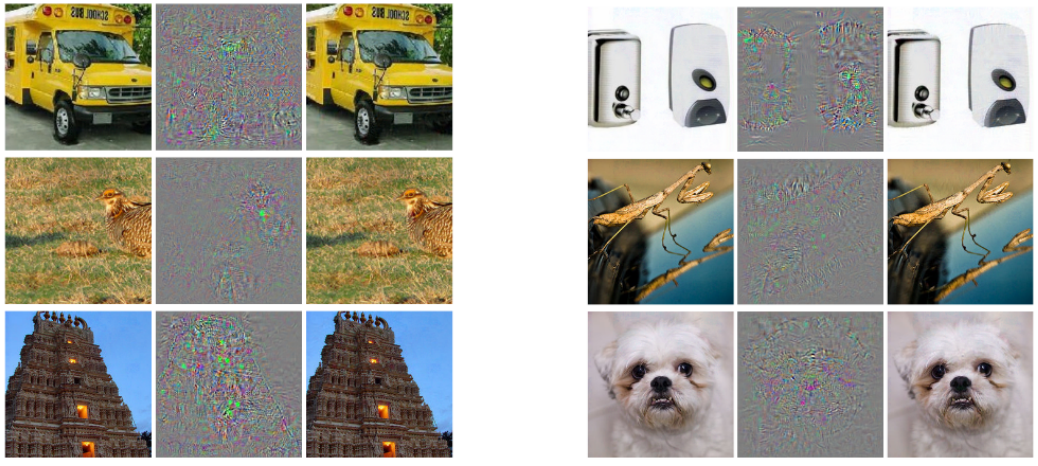
(c) Effect of fine-tuning on adversarial examples computed by two different methods for NIN on CIFAR-10.



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

More entertaining empirical results

From [18] "Intriguing properties of neural networks" by Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow and Rob Fergus:



(a) (b)

Figure 5: Adversarial examples generated for AlexNet [9].(Left) is a correctly predicted sample, (center) difference between correct image, and image predicted incorrectly magnified by 10x (values shifted by 128 and clamped), (right) adversarial example. All images in the right column are predicted to be an *"ostrich, Struthio camelus"*. Average distortion based on 64 examples is 0.006508. Plase refer to `http://goo.gl/huaGPb` for full resolution images. The examples are strictly randomly chosen. There is not any postselection involved.

(a) (b)

Figure 6: Adversarial examples for QuocNet [10]. A binary car classifier was trained on top of the last layer features without fine-tuning. The randomly chosen examples on the left are recognized correctly as cars, while the images in the middle are not recognized. The rightmost column is the magnified absolute value of the difference between the two images.

On NN-instability in general (Anders Hansen lecture)

Excerpt from a lecture by Anders Hansen, May 14th 2018

The First Paradox (stability)

There is an uncountable family of classification functions $f : \mathbb{R}^{N_0} \to \{0, 1\}$ such that for any neural network dimensions $\mathbf{N} = (N_L, N_{L-1}, \ldots, N_1, N_0)$ with $N_0, L \geq 2$ and any $0 < \epsilon < 1/(K + M)$ where M is arbitrarily large and $K \geq 3(N_1 + 1) \cdots (N_{L-1} + 1)$ we have the following. There exist uncountably many training sets $\mathcal{T} = \{x^1, \ldots, x^K\}$ and uncountably many classification sets $\mathcal{C} = \{y^1, \ldots, y^M\}$ such that there is a

$$\tilde{\phi} \in \operatorname*{argmin}_{\phi \in \mathcal{NN}_{\mathbf{N},L}} C(v, w), \quad v_j = \phi(x^j), \quad w_j = f(x^j),$$

where $1 \leq j \leq K$ such that

$$\tilde{\phi}(x) = f(x) \quad \forall x \in \mathcal{T} \cup \mathcal{C}.$$

However, there exists uncountably many $v \in \mathbb{R}^{N_0}$ such that

$$|\tilde{\phi}(v) - f(v)| \geq 1/2, \qquad \|v - x\|_\infty \leq \epsilon \text{ for some } x \in \mathcal{T}.$$

Moreover, there is a neural network $\hat{\phi}$, not necessarily trained, such that

$$\hat{\phi}(x) = f(x) \quad \forall x \in \mathcal{B}_\epsilon^\infty(\mathcal{T} \cup \mathcal{C}).$$

20 / 125

The full slide series can be found here:

http://www.damtp.cam.ac.uk/research/afha/lectures/trends2018/index.html

The essence in the above is concentrated in slides 5,6,10 and 20.