Paper

- **Title:** DeepFool: a simple and accurate method to fool deep neural networks
- **Authors:** Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Pascal Frossard
- **arXiv link:** https://arxiv.org/abs/1511.04599

TL;DR

They introduce a new algorithm to generate adversarial examples for a neural network such that the adversarial example minimizes any L_p distance to the original input.

Binary classifier

- The label given to input x it just the sign of the classifier's output, $\hat{k}(x) = sign(f(x))$
- Let f be affine, $f(x) = w^T x + b$
- The minimum perturbation required to change the label will be

  $$\begin{aligned} r_*(x_0) &= arg\ min\ ||r||_2 \text{ subject to } sign(f(x_0)) \neq sign(f(x_0 + r)) \\ &= -\frac{f(x_0)}{||w||_2^2} w \end{aligned}$$

  which is just the distance of $f(x_0)$ from the separating hyperplane.
- For a general binary classifier, adopt an iterative procedure. At each iteration, f is linearized around the current point x_i and the minimal perturbation of the linearized classifier is computed

  $$\underset{r_i}{arg\ min}\ ||r_i||_2 \text{ subject to } f(x_i) + \nabla f(x_i)^T r_i = 0$$

  where r_i is calculated using the affine binary classifier case.

Algorithm 1 DeepFool for binary classifiers

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.
Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\text{sign}(f(\boldsymbol{x}_i)) = \text{sign}(f(\boldsymbol{x}_0))$ **do**
    $\boldsymbol{r}_i \leftarrow -\frac{f(\boldsymbol{x}_i)}{\|\nabla f(\boldsymbol{x}_i)\|_2^2} \nabla f(\boldsymbol{x}_i)$,
    $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$,
    $i \leftarrow i + 1$.
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$.

Multiclass classifier

- For c labels, the label given to input x will be
$$\hat{k}(x) = \underset{k}{arg\ max}\ f_k(x)$$
- Let f be affine i.e. $f(x) = w^T x + b$, the minimal perturbation will be
$$\underset{r}{arg\ min}\ ||r||_2 \text{ such that } \exists k : w_k^T(x_0 + r) + b_k \geq w_{\hat{k}(x_0)}^T(x_0 + r) + b_{\hat{k}(x_0)}$$
- This is nothing but the distance between x_0 and complement of convex polyhedron P,
$$\bigcap_{k=1}^{c} \{x : f_{\hat{k}(x_0)}(x) \geq f_k(x)\}$$
- Let $\hat{l}(x_0)$ be the closest boundary of P to x_0,
$$\hat{l}(x_0) = \underset{k \neq \hat{k}(x_0)}{arg\ min}\ \frac{|f_k(x_0) - f_{\hat{k}(x_0)}(x_0)|}{||w_k - w_{\hat{k}(x_0)}||_2}$$
- And the minimum perturbation will be the projection on P:
$$r_*(x_0) = \frac{|f_{\hat{l}(x_0)}(x_0) - f_{\hat{k}(x_0)}(x_0)|}{||w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)}||_2^2}(w_{\hat{l}(x_0)} - w_{\hat{k}(x_0)})$$
- For non-linear classifiers, use iterative algorithm and at each step i, estimate P by $\widetilde{P}_i$
$$\widetilde{P}_i = \{x : f_k(x) - f_{\hat{k}(x_0)}(x) + \nabla f_k(x_i)^T x - \nabla f_{\hat{k}(x_0)}(x_i)^T x \leq 0\}$$

Algorithm 2 DeepFool: multi-class case

input: Image $\boldsymbol{x}$, classifier f.
output: Perturbation $\hat{\boldsymbol{r}}$.

Initialize $\boldsymbol{x}_0 \leftarrow \boldsymbol{x}$, $i \leftarrow 0$.
while $\hat{k}(\boldsymbol{x}_i) = \hat{k}(\boldsymbol{x}_0)$ **do**
  for $k \neq \hat{k}(\boldsymbol{x}_0)$ **do**
    $\boldsymbol{w}'_k \leftarrow \nabla f_k(\boldsymbol{x}_i) - \nabla f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
    $f'_k \leftarrow f_k(\boldsymbol{x}_i) - f_{\hat{k}(\boldsymbol{x}_0)}(\boldsymbol{x}_i)$
  end for
  $\hat{l} \leftarrow \arg\min_{k \neq \hat{k}(\boldsymbol{x}_0)} \frac{|f'_k|}{\|\boldsymbol{w}'_k\|_2}$
  $\boldsymbol{r}_i \leftarrow \frac{|f'_{\hat{l}}|}{\|\boldsymbol{w}'_{\hat{l}}\|_2^2} \boldsymbol{w}'_{\hat{l}}$
  $\boldsymbol{x}_{i+1} \leftarrow \boldsymbol{x}_i + \boldsymbol{r}_i$
  $i \leftarrow i + 1$
end while
return $\hat{\boldsymbol{r}} = \sum_i \boldsymbol{r}_i$

L_p norms

To find adversarial examples for any L_p norm ($p \in [1, \infty)$), let q be the conjugate norm of p.
Replace the two updates in the algorithm with

$$\hat{l} \leftarrow \underset{k \neq \hat{k}(x_0)}{arg\ min} \frac{|f'_k|}{||w'_k||_q}$$

$$r_i \leftarrow \frac{|f'_{\hat{l}}|}{||w'_{\hat{l}}||_q^q} \odot sign(w'_{\hat{l}})$$