ECE 595: Homework 5

Yi Qiao, Class ID 187

(Spring 2019)

Exercise 1: Adversarial Attacks on Gaussian Classifier

(a) minimum-norm attacks

(i) minimum l_2 and l_∞ attack

Since we only have 2 classes, the question becomes

$$\begin{aligned} &\underset{\boldsymbol{x}}{minimize}\ ||\boldsymbol{x}-\boldsymbol{x}_0|| \\ &subject\ to\ \boldsymbol{w}^T\boldsymbol{x}+w_0=0 \end{aligned} \tag{1}$$

using l_2 norm
the problem is the same as

$$\begin{aligned} &\underset{\boldsymbol{x}}{minimize}\ \frac{1}{2}||\boldsymbol{x}-\boldsymbol{x}_0||_2^2 \\ &subject\ to\ \boldsymbol{w}^T\boldsymbol{x}+w_0=0 \end{aligned} \tag{2}$$

The lagrangian is

$$\mathcal{L}(\boldsymbol{x},\lambda)=\frac{1}{2}||\boldsymbol{x}-\boldsymbol{x}_0||_2^2+\lambda(\boldsymbol{w}^T\boldsymbol{x}+w_0) \tag{3}$$

Taking the derivative

$$\begin{aligned} \nabla_{\boldsymbol{x}}\mathcal{L}(\boldsymbol{x},\lambda)&=\boldsymbol{x}-\boldsymbol{x}_0+\lambda\boldsymbol{w}=0 \\ \frac{\partial}{\partial\lambda}\mathcal{L}(\boldsymbol{x},\lambda)&=\boldsymbol{w}^T\boldsymbol{x}+w_0=0 \end{aligned} \tag{4}$$

$$\begin{aligned} \lambda\boldsymbol{w}&=\boldsymbol{x}_0-\boldsymbol{x} \\ \lambda\boldsymbol{w}^T\boldsymbol{w}&=\boldsymbol{w}^T\boldsymbol{x}_0-\boldsymbol{w}^T\boldsymbol{x} \\ \lambda&=\left(\boldsymbol{w}^T\boldsymbol{w}\right)^{-1}(\boldsymbol{w}^T\boldsymbol{x}_0+w_0) \end{aligned} \tag{5}$$

$$\begin{aligned} \boldsymbol{x}&=\boldsymbol{x}_0-\lambda\boldsymbol{w} \\ &=\boldsymbol{x}_0-\frac{\boldsymbol{w}(\boldsymbol{w}^T\boldsymbol{x}_0+w_0)}{||\boldsymbol{w}||_2^2} \end{aligned} \tag{6}$$

using l_∞ norm

$$\begin{aligned} &\underset{\boldsymbol{x}}{minimize}\ ||\boldsymbol{x}-\boldsymbol{x}_0||_\infty \\ &subject\ to\ \boldsymbol{w}^T\boldsymbol{x}+w_0=0 \end{aligned} \tag{7}$$

Let $\boldsymbol{r}=\boldsymbol{x}-\boldsymbol{x}_0$, $b_0=-(\boldsymbol{w}^T\boldsymbol{x}_0+w_0)$, the problem becomes:

$$\begin{aligned} &\underset{\boldsymbol{x}}{argmin}||\boldsymbol{x}-\boldsymbol{x}_0||_\infty \\ &subject\ to\ \boldsymbol{w}^T\boldsymbol{r}=b_0 \end{aligned} \tag{8}$$

The lagrangian is

$$\mathcal{L}(\boldsymbol{r}, \lambda) = ||\boldsymbol{r}||_\infty + \lambda(b_0 - \boldsymbol{w}^T\boldsymbol{r}) \tag{9}$$

Taking derivative,

$$\frac{\partial}{\partial\lambda}\mathcal{L}(\boldsymbol{r}, \lambda) = b_0 - \boldsymbol{w}^T\boldsymbol{r} = 0 \tag{10}$$

By Holder's Inequality:

$$\begin{gathered} |b_0| = |\boldsymbol{w}^T\boldsymbol{r}| \leq ||\boldsymbol{w}||_1||\boldsymbol{r}||_\infty \\ ||\boldsymbol{r}||_\infty \geq \frac{|b_0|}{||\boldsymbol{w}||_1} \end{gathered} \tag{11}$$

Consider $\boldsymbol{r} = \eta \cdot sign(\boldsymbol{w})$, for some constant η tbd. We can show that

$$||\boldsymbol{r}||_\infty = \underset{i}{argmax}\ |\eta \cdot sign(w_i)| = |\eta| \tag{12}$$

let $\eta = \frac{b_0}{||\boldsymbol{w}||_1} \cdot sign(\boldsymbol{w})$, then we have,

$$||\boldsymbol{r}||_\infty = |\eta| = \frac{b_0}{||\boldsymbol{w}||_1} \tag{13}$$

Lower bound is achieved, thus the solution is,

$$\boldsymbol{r} = \frac{|b_0|}{||\boldsymbol{w}_1||} \cdot sign(\boldsymbol{w}) \tag{14}$$

(ii) DeepFool attack

$$\begin{aligned} &\underset{\boldsymbol{x}}{argmin}\ ||\boldsymbol{x} - \boldsymbol{x}_0||_2^2 \\ &subject\ to\ g(\boldsymbol{x}) = 0 \end{aligned} \tag{15}$$

First order approximation

$$g(\boldsymbol{x}) \approx g(\boldsymbol{x}^{(k)}) + \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})^T(\boldsymbol{x} - \boldsymbol{x}^{(k)}) \tag{16}$$

Then the problem can be approximate by

$$\begin{aligned} &\underset{\boldsymbol{x}}{argmin}\ ||\boldsymbol{x} - \boldsymbol{x}_0||_2^2 \\ &subject\ to\ g(\boldsymbol{x}^{(k)}) + \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})^T(\boldsymbol{x} - \boldsymbol{x}^{(k)}) = 0 \end{aligned} \tag{17}$$

Let $\boldsymbol{w}^{(k)} = \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})$ and $w_0^{(k)} = g(\boldsymbol{x}^{(k)}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})^T\boldsymbol{x}^{(k)}$
Then the problem is equivalent to

$$\begin{aligned} &\underset{\boldsymbol{x}}{argmin}\ ||\boldsymbol{x} - \boldsymbol{x}_0||_2^2 \\ &subject\ to\ (\boldsymbol{w}^{(k)})^T\boldsymbol{x} + w_0^{(k)} = 0 \end{aligned} \tag{18}$$

This is the same problem as minimum l_2 norm attack, Thus the solution will be,

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \frac{((\boldsymbol{w}^{(k)})^T x^{(k)} + w_0^{(k)})\boldsymbol{w}^{(k)}}{||\boldsymbol{w}^{(k)}||_2^2} \tag{19}$$

substitute $\boldsymbol{w}$ and w_0 back, we get

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \left(\frac{g(\boldsymbol{x}^{(k)})}{||\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})||^2}\right)\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)}) \tag{20}$$

(iii) An example DeepFool never converge

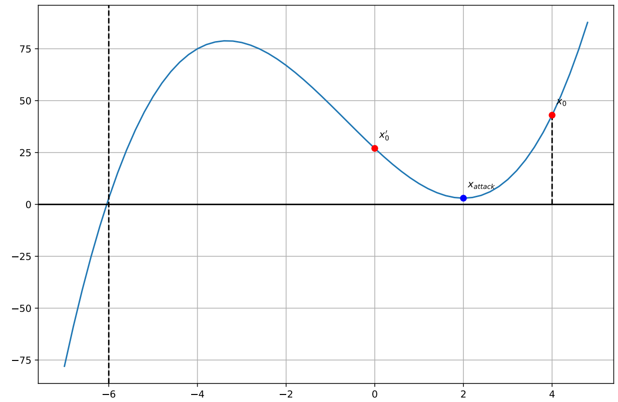


Figure 1: 1D example classifier

Above is a example discriminate function $g(x)$. there are two classes, separated by the vertical line at $g(x) = 0$. If x_0 at the position shown above, the Deep Fool attack will result in x_{attack} in the end, and thus never converge to $g(x) = 0$.

(b) maximum-allowable attack

(i) l_∞ attack in the linear case

The problem is,

$$\begin{aligned} &\underset{\boldsymbol{x}}{argmin}\ \boldsymbol{w}^T\boldsymbol{x} + w_0 \\ &subject\ to\ ||\boldsymbol{x} - \boldsymbol{x}_0||_\infty < \eta \end{aligned} \tag{21}$$

let $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{r}$, $b_0 = (\boldsymbol{w}^T\boldsymbol{x}_0 + w_0)$, the problem becomes,

$$\begin{aligned} &\underset{\boldsymbol{r}}{argmin}\ \boldsymbol{w}^T\boldsymbol{r} + b_0 \\ &subject\ to\ ||\boldsymbol{r}||_\infty < \eta \end{aligned} \tag{22}$$

by Holder's inequality,

$$\boldsymbol{w}^T\boldsymbol{r} \geq -||\boldsymbol{r}||_\infty||\boldsymbol{w}||_1 \geq -\eta||\boldsymbol{w}||_1 \tag{23}$$

as shown in the lecture note, the solution

$$\boldsymbol{r} = -\eta \cdot sign(\boldsymbol{w}) \tag{24}$$

(ii) FGSM attack

$$\begin{aligned} &\underset{\boldsymbol{x}}{argmax}\ J(\boldsymbol{x}, \boldsymbol{w}) \\ &subject\ to\ ||\boldsymbol{x} - \boldsymbol{x}_0||_\infty \leq \eta \end{aligned} \tag{25}$$

Approximately, $J(\boldsymbol{x}, \boldsymbol{w}) = J(\boldsymbol{x}_0 + \boldsymbol{r}, \boldsymbol{w}) \approx J(\boldsymbol{x}_0, \boldsymbol{w}) + \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0, \boldsymbol{w})^T \boldsymbol{r}$
Then, the problem becomes

$$\begin{aligned} &\underset{\boldsymbol{x}}{argmin} \ - J(\boldsymbol{x}_0, \boldsymbol{w}) - \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0, \boldsymbol{w})^T \boldsymbol{r} \\ &subject\ to\ ||\boldsymbol{r}||_\infty \leq \eta \end{aligned} in \tag{26}$$

of which the solution is given by

$$\boldsymbol{x} = \boldsymbol{x}_0 + \eta \cdot (\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0, \boldsymbol{w})) \tag{27}$$

In the problem setup, we get $J(\boldsymbol{x}) = -g(\boldsymbol{x})$, thus the solution is

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{x}_0 - \eta \cdot (\nabla_{\boldsymbol{x}} g(\boldsymbol{x}_0)) \\ &= \boldsymbol{x}_0 - \eta \cdot ((\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x}_0 + (\boldsymbol{w}_j - \boldsymbol{w}_t)) \\ &= (\eta(\boldsymbol{W}_t - \boldsymbol{W}_j) + 1)\boldsymbol{x}_0 + \eta(\boldsymbol{w}_t - \boldsymbol{w}_j) \end{aligned} \tag{28}$$

(iii) I-FGSM attack

$$\begin{aligned} J(\boldsymbol{x}, \boldsymbol{w}) &= J(\boldsymbol{x}_0 + \boldsymbol{r}, \boldsymbol{w}) \\ &\approx J(\boldsymbol{x}_0, \boldsymbol{w}) + \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0, \boldsymbol{w})^T \boldsymbol{r} \\ &= J(\boldsymbol{x}_0, \boldsymbol{w}) + \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0, \boldsymbol{w})^T (\boldsymbol{x} - \boldsymbol{x}_0) \\ &= J(\boldsymbol{x}_0, \boldsymbol{w}) + \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0, \boldsymbol{w})^T \boldsymbol{x} - \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0, \boldsymbol{w})^T \boldsymbol{x}_0 \end{aligned} \tag{29}$$

$$\begin{aligned} \boldsymbol{x}^{(k+1)} &= \underset{0 \leq \boldsymbol{x} \leq 1}{argmax}\ J(\boldsymbol{x}^{(k)}, \boldsymbol{w})\ subject\ to\ ||\boldsymbol{x} - \boldsymbol{x}_0|| \leq \eta \\ &= \underset{0 \leq \boldsymbol{x} \leq 1}{argmax}\ \nabla_{\boldsymbol{x}} J(\boldsymbol{x}^{(k)}, \boldsymbol{w})^T \boldsymbol{x}\ subject\ to\ ||\boldsymbol{x} - \boldsymbol{x}_0|| \leq \eta \\ &= \mathcal{P}\left\{ \boldsymbol{x}^{(k)} + \eta \cdot sign\left(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}^{(k)}, \boldsymbol{w}) \right) \right\} \end{aligned} \tag{30}$$

(c) Regularization based attack

(i) linear case

$$\underset{\boldsymbol{x}}{argmin}\ \frac{1}{2}||\boldsymbol{x} - \boldsymbol{x}_0||_2^2 + \lambda(\boldsymbol{w}^T \boldsymbol{x} + w_0) \tag{31}$$

Taking derivative

$$\begin{aligned} &\nabla_{\boldsymbol{x}}\ \frac{1}{2}||\boldsymbol{x} - \boldsymbol{x}_0||_2^2 + \lambda(\boldsymbol{w}^T \boldsymbol{x} + w_0) \\ &= \boldsymbol{x} - \boldsymbol{x}_0 + \lambda \boldsymbol{w} = 0 \end{aligned} \tag{32}$$

Solve for $\boldsymbol{x}$,

$$\boldsymbol{x} = \boldsymbol{x}_0 - \lambda \boldsymbol{w} \tag{33}$$

(ii)

$$
\begin{aligned}
&\underset{\boldsymbol{x}}{argmin}\ \varphi(\boldsymbol{x}), where \\
&\varphi(\boldsymbol{x}) = ||\boldsymbol{x} - \boldsymbol{x}_0||_2^2 + \lambda\zeta(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x})), \\
&\zeta(y) = max(y, 0),\ and\ j \neq t
\end{aligned} \tag{34}
$$

Taking the derivative,

$$
\nabla\varphi(\boldsymbol{x}) = 2(\boldsymbol{x} - \boldsymbol{x}_0) + \lambda\mathbb{I}\{g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}) > 0\} \cdot (\nabla_{\boldsymbol{x}} g_j(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} g_t(\boldsymbol{x})) \tag{35}
$$

Using my favorite gradient descent, we can tell,

$$
\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha\nabla_{\boldsymbol{x}}\varphi(\boldsymbol{x}^{(k)}) \tag{36}
$$

substituting in $g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T(\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x} + (\boldsymbol{w}_j - \boldsymbol{w}_t)^T\boldsymbol{x} + (w_{j,0} - w_{t,0})$, we get

$$
\begin{aligned}
\nabla_{\boldsymbol{x}} g(\boldsymbol{x}) &= (\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x} + (\boldsymbol{w}_j - \boldsymbol{w}_t) \\
\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - 2\alpha(\boldsymbol{x}^{(k)} - \boldsymbol{x}_0) - \alpha\lambda\mathbb{I}\{g(\boldsymbol{x}) > 0\} \cdot (\nabla_{\boldsymbol{x}} g(\boldsymbol{x})) \\
&= \boldsymbol{x}^{(k)} - 2\alpha(\boldsymbol{x}^{(k)} - \boldsymbol{x}_0) - \alpha\lambda\mathbb{I}\{g(\boldsymbol{x}) > 0\} \cdot ((\boldsymbol{W}_j - \boldsymbol{W}_t)\boldsymbol{x} + (\boldsymbol{w}_j - \boldsymbol{w}_t))
\end{aligned} \tag{37}
$$

Exercise 2: CW-Attack on Gaussian Classifier - Non-overlapping Patches

(a)&(b) Implementation

refer to code in the back

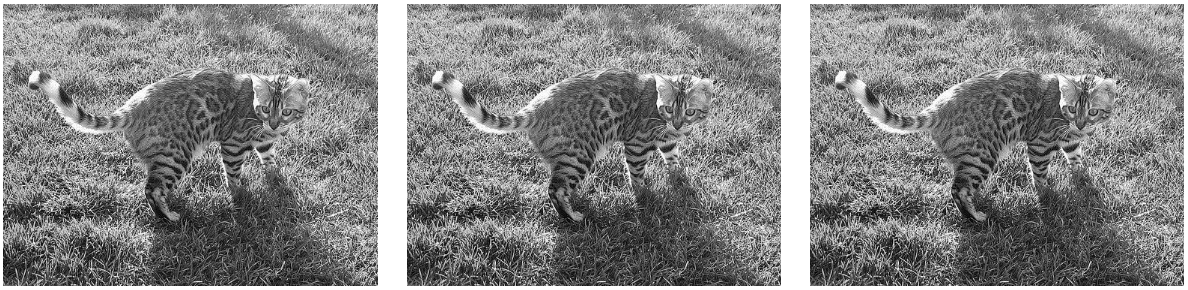
(c) Results

(i) & (ii) The final perturbed images & their perturbation respectively

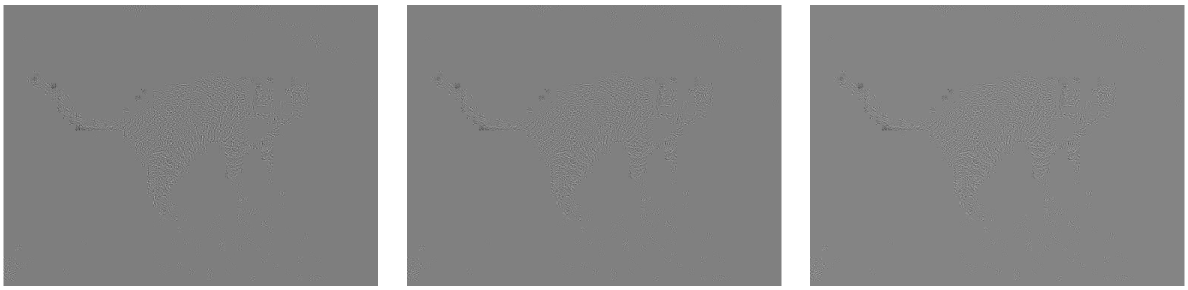
(iii) Frobenius norm of the perturbation

λ	1	5	10
Frobenius norm	8.253352243144604	8.928742803780679	9.697938770929634

Table 1: Frobenius norm vs. λ for CW-Attack on Gaussian classifier with non-overlapping patches



(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 10$



(d) $\lambda = 1$ (e) $\lambda = 5$ (f) $\lambda = 10$

Figure 2: perturbed images and their perturbations with $\lambda = 1, 5, 10$

(iv) The classifiers output

(a) $\lambda = 1$ (b) $\lambda = 5$ (c) $\lambda = 10$

Figure 3: classified perturbed image with $\lambda = 1, 5, 10$

For all three cases, all patches are classified as grass after applying perturbation.

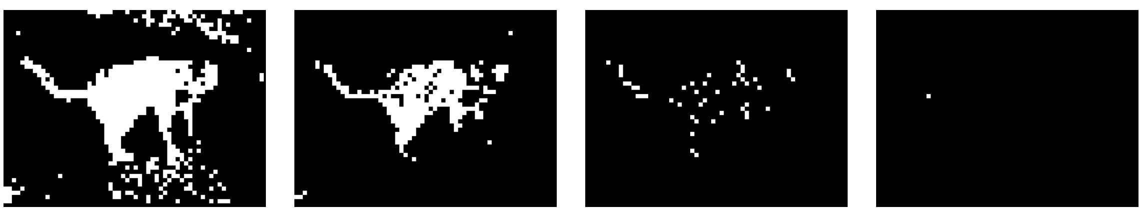
(v) Plots during gradient descent

(a) after 0 rounds (b) after 50 rounds (c) after 100 rounds

Figure 4: classified perturbed image during gradient descent with $\lambda = 1$

(a) after 0 rounds (b) after 10 rounds (c) after 20 rounds

Figure 5: classified perturbed image during gradient descent with $\lambda = 5$



(a) after 0 rounds (b) after 5 rounds (c) after 10 rounds (d) after 15 rounds

Figure 6: classified perturbed image during gradient descent with $\lambda = 10$

Comments

From the above plots, we can obviously see that increase λ will speed up the attack. However, as λ goes up, the quality of the attack goes down since the Frobenius norm also goes up while achieving the same result.

Exercise 3: CW Attack on Gaussian Classifier - Overlapping Patches

$$
\begin{aligned}
\boldsymbol{X}^* &= \underset{\boldsymbol{X}}{argmin} \sum_{i=1}^{L} \{||\boldsymbol{P}_i(\boldsymbol{X} - \boldsymbol{X}_0)||_2^2 + \lambda max(g_j(\boldsymbol{P}_i\boldsymbol{X}) - g_t(\boldsymbol{P}_i\boldsymbol{X}), 0)\} \\
&= \underset{\boldsymbol{X}}{argmin} ||\boldsymbol{X} - \boldsymbol{X}_0||_2^2 + \lambda \sum_{i=1}^{L} max(g_j(\boldsymbol{P}_i\boldsymbol{X}) - g_t(\boldsymbol{P}_i\boldsymbol{X}), 0)
\end{aligned}
\tag{38}
$$

(a) Theory

the gradient of the above function is,

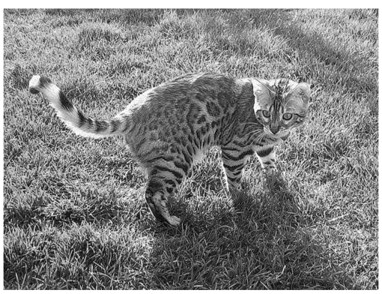
$$
\nabla_{\boldsymbol{X}} = 2(\boldsymbol{X} - \boldsymbol{X}_0) + \lambda \mathbb{I}\{g_j(\boldsymbol{P}_i\boldsymbol{X}) - g_t(\boldsymbol{P}_i\boldsymbol{X}) > 0\} \times (\nabla_{\boldsymbol{X}}(g_j(\boldsymbol{P}_i\boldsymbol{X}) - g_t(\boldsymbol{P}_i\boldsymbol{X}))) \tag{39}
$$

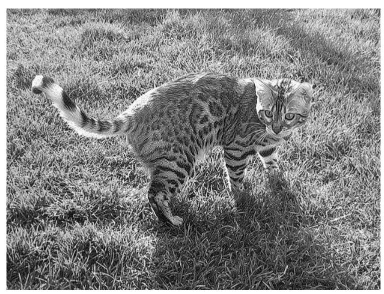
(b) Implementation

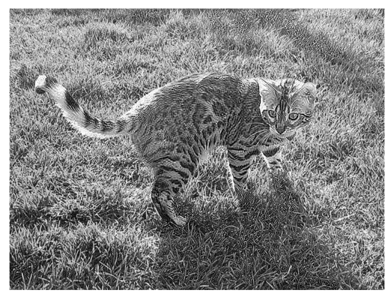
Refer to code in the back.

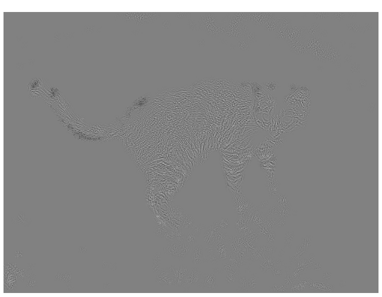
(c) Results

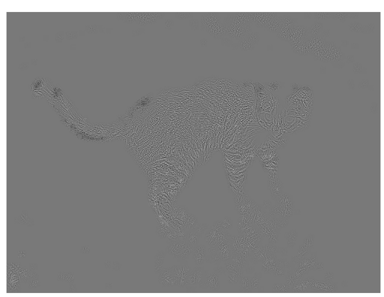
(i) & (ii) The final perturbed images & their perturbation respectively


(a) $\lambda = 0.5$


(b) $\lambda = 1.0$


(c) $\lambda = 5.0$


(d) $\lambda = 0.5$


(e) $\lambda = 1.0$


(f) $\lambda = 5.0$

Figure 7: perturbed images and their perturbations with $\lambda = 0.5, 1.0, 5.0$

(iii) Frobenius norm of the perturbation

λ	#grass patches	#cat patches	Frobenius norm
0.5	180561	3	14.059260687697552
1.0	180561	3	16.158563208747122
5.0	180564	0	33.617113348842395

Table 2: testing results for CW-Attack on Gaussian classifier with overlapping patches

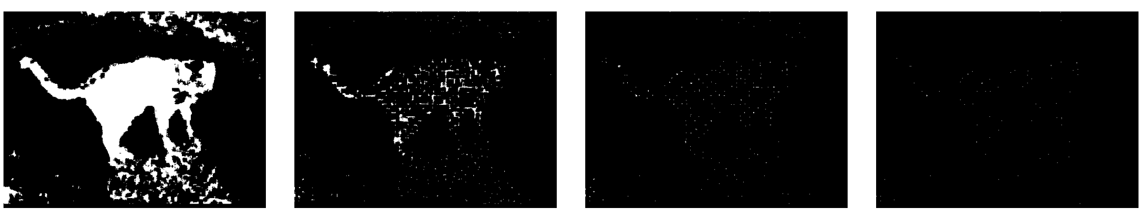
(iv) The classifiers output

(a) $\lambda = 0.5$ (b) $\lambda = 1.0$ (c) $\lambda = 5.0$

Figure 8: classified perturbed image with $\lambda = 0.5, 1.0, 5.0$

The precise result is in the table above, however, from observation, almost all pixels are classified as grass.

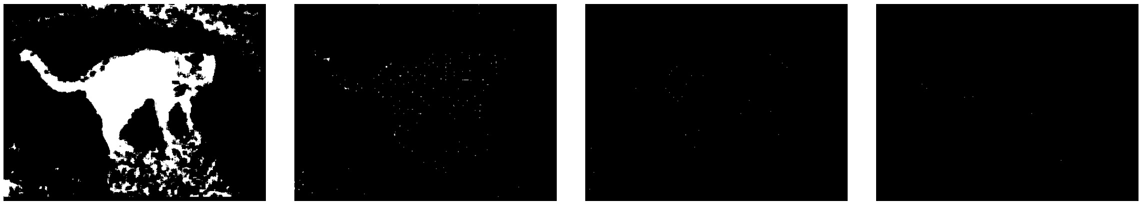
(v) Plots during gradient descent



(a) after 0 rounds (b) after 5 rounds (c) after 10 rounds (d) after 15 rounds

(e) after 20 rounds (f) after 25 rounds (g) after 30 rounds

Figure 9: classified perturbed image during gradient descent with $\lambda = 0.5$



(a) after 0 rounds (b) after 5 rounds (c) after 10 rounds (d) after 15 rounds

Figure 10: classified perturbed image during gradient descent with $\lambda = 1.0$

(a) after 0 rounds (b) after 2 rounds (c) after 4 rounds

Figure 11: classified perturbed image during gradient descent with $\lambda = 5$

Comments

In general, comparing to the non-overlapping case, the Frobenius norm is significantly larger. However, since the overlapping is a much harder problem, the result is acceptable. As *lambda* goes up, the strength of the attack is approximately the same, less rounds are needed to attack the classifier while sacrificing some quality of the attack.

Code

Exercise 2: CW-Attack on Gaussian Classifier - Non-overlapping Patches

```
#! /usr/bin/env python3
import numpy as np
import scipy.ndimage
from scipy.misc import imsave

def readImg(pix):
    return  scipy.ndimage.imread(pix, mode='L') /255


def get_info():
    train_cat   = np.matrix(np.loadtxt('../data/train_cat.txt', delimiter=','))
    train_grass = np.matrix(np.loadtxt('../data/train_grass.txt', delimiter=','))

    mu_cat      = np.asmatrix(np.mean(train_cat, 1))
    mu_grass    = np.asmatrix(np.mean(train_grass, 1))

    cov_cat     = np.asmatrix(np.cov(train_cat))
    cov_grass   = np.asmatrix(np.cov(train_grass))
```

```
    pi_cat      = len(train_cat.T) / (len(train_cat.T) + len(train_grass.T))
    pi_grass    = len(train_grass.T) / (len(train_cat.T) + len(train_grass.T))

    return {"mean":mu_cat, "cov":cov_cat, "prior":pi_cat}, {"mean":mu_grass,
    ↪  "cov":cov_grass, "prior":pi_grass}



def CWAttack(cat_info, grass_info, img, rounds = 300, alpha = 0.0001, lm = 5):

    Wcat = np.linalg.inv(cat_info["cov"])
    wcat = -Wcat * cat_info["mean"]
    w0cat = (cat_info["mean"].T * Wcat * cat_info["mean"]) / 2 \
        + np.log(np.linalg.det(cat_info["cov"])) / 2 - np.log(cat_info["prior"])

    Wgrass = np.linalg.inv(grass_info["cov"])
    wgrass = -Wgrass * grass_info["mean"]
    w0grass = grass_info["mean"].T * Wgrass * grass_info["mean"] / 2 \
        + np.log(np.linalg.det(grass_info["cov"])) / 2 -
        ↪  np.log(grass_info["prior"])


    def __gee(x):

        # Wt == Wgrass,  Wj == Wcat
        ret = x.T * (Wgrass - Wcat) * x / 2 + (wgrass - wcat).T * x + (w0grass -
        ↪  w0cat)
        return ret


    def gradient(z_vector, z_0, lm):
        # Calculate g_j, g_t, determine if patch_vec is already in target class
        # If patch_vec is in target class, do not add any perturbation (return
        ↪  zero gradient!)
        # Else, calculate the gradient, using results from 1(c)(ii)

        if __gee(z_vector) > 0:
            return 2 * (z_vector - z_0) + lm * ((Wgrass - Wcat) * z_vector +
            ↪  (wgrass - wcat))
        else:
            #return 2 * (z_vector - z_0)
            return np.zeros((64,1))


    M,N = img.shape
    img_orig = img.copy()
    range_i = range(0,M-8,8)
```

```
range_j = range(0,N-8,8)


def classify(img):
    output = np.zeros((M,N))
    for i in range_i:
        for j in range_j:
            z = img[i:i+8, j:j+8]
            z_vector = np.asmatrix(z.flatten('F')).T
            if __gee(z_vector) > 0:
                for ii in range(8):
                    for jj in range(8):
                        output[i+ii][j+jj] = 1
    return output


def cntclass(img):
    num_cat = 0
    num_grass = 0
    for i in range_i:
        for j in range_j:
            z = img[i:i+8, j:j+8]
            z_vector = np.asmatrix(z.flatten('F')).T
            if __gee(z_vector) < 0:
                num_cat+=1
            else:
                num_grass+=1
    return num_grass, num_cat

for r in range(rounds):
    if r % (50 / lm) == 0:
        print(f"round: {r}")
        imsave(f'../pix/CW_Nonoverlap_{lm}_r{r}.png', 255*classify(img))

    img_prev = img.copy()
    for i in range_i:
        for j in range_j:
            z = img[i:i+8, j:j+8]
            z_vector = np.asmatrix(z.flatten('F')).T
            z_vector_orig = np.asmatrix(img_orig[i:i+8,
             ↪  j:j+8].flatten('F')).T
            z_vector -= alpha * gradient(z_vector, z_vector_orig, lm)
            img[i:i+8, j:j+8] = np.reshape(np.clip(z_vector, 0.0, 1.0),
             ↪  (8,8), order='F')


    change = np.linalg.norm(img - img_prev)
    if(change < 0.001):
```

```
            print(f"finished, total round: {r}")
            break;


    cnts = cntclass(img)
    print(f"number of grass patches {cnts[0]}, cat patches {cnts[1]}")
    change = np.linalg.norm(img - img_orig)
    print(f"Frobenius norm: {change}")

    imsave(f'../pix/CW_Nonoverlap_{lm}.png', img)
    imsave(f'../pix/CW_Nonoverlap_{lm}_perturbation.png',img-img_orig)
    imsave(f'../pix/CW_Nonoverlap_{lm}_classified_pertub.png', classify(img))

if __name__ == "__main__":
    cat_info, grass_info = get_info()
    CWAttack(cat_info, grass_info, readImg('../data/cat_grass.jpg'), lm = 1)
    CWAttack(cat_info, grass_info, readImg('../data/cat_grass.jpg'), lm = 5)
    CWAttack(cat_info, grass_info, readImg('../data/cat_grass.jpg'), lm = 10)
```

Exercise 3: CW-Attack on Gaussian Classifier - Overlapping Patches

```
#! /usr/bin/env python3
import numpy as np
import scipy.ndimage
from scipy.misc import imsave

def readImg(pix):
    return  scipy.ndimage.imread(pix, mode='L') /255


def get_info():
    train_cat   = np.matrix(np.loadtxt('../data/train_cat.txt', delimiter=','))
    train_grass = np.matrix(np.loadtxt('../data/train_grass.txt', delimiter=','))

    mu_cat      = np.asmatrix(np.mean(train_cat, 1))
    mu_grass    = np.asmatrix(np.mean(train_grass, 1))

    cov_cat     = np.asmatrix(np.cov(train_cat))
    cov_grass   = np.asmatrix(np.cov(train_grass))

    pi_cat      = len(train_cat.T) / (len(train_cat.T) + len(train_grass.T))
    pi_grass    = len(train_grass.T) / (len(train_cat.T) + len(train_grass.T))

    return {"mean":mu_cat, "cov":cov_cat, "prior":pi_cat}, {"mean":mu_grass,
     ↪  "cov":cov_grass, "prior":pi_grass}
```

```python
def CWAttack(cat_info, grass_info, img, rounds = 300, alpha = 0.0001, lm = 0.5):

    Wcat = np.linalg.inv(cat_info["cov"])
    wcat = -Wcat * cat_info["mean"]
    w0cat = (cat_info["mean"].T * Wcat * cat_info["mean"]) / 2 \
        + np.log(np.linalg.det(cat_info["cov"])) / 2 - np.log(cat_info["prior"])

    Wgrass = np.linalg.inv(grass_info["cov"])
    wgrass = -Wgrass * grass_info["mean"]
    w0grass = grass_info["mean"].T * Wgrass * grass_info["mean"] / 2 \
        + np.log(np.linalg.det(grass_info["cov"])) / 2 -
        ↪ np.log(grass_info["prior"])


    def __gee(x):

        # Wt == Wgrass,  Wj == Wcat
        ret = x.T * (Wgrass - Wcat) * x / 2 + (wgrass - wcat).T * x + (w0grass -
        ↪ w0cat)
        return ret


    def gradient(z_vector, lm):
        # Calculate g_j, g_t, determine if patch_vec is already in target class
        # If patch_vec is in target class, do not add any perturbation (return
        ↪ zero gradient!)
        # Else, calculate the gradient, using results from 1(c)(ii)

        if __gee(z_vector) > 0:
            return lm * ((Wgrass - Wcat) * z_vector + (wgrass - wcat))
        else:
            return np.zeros((64,1))


    M,N = img.shape
    img_orig = img.copy()
    range_i = range(0,M-8)
    range_j = range(0,N-8)


    def classify(img):
        output = np.zeros((M,N))
        for i in range_i:
            for j in range_j:
                z = img[i:i+8, j:j+8]
                z_vector = np.asmatrix(z.flatten('F')).T
                if __gee(z_vector) > 0:
                    output[i][j] = 1
```

```
    return output


def cntclass(img):
    num_cat = 0
    num_grass = 0
    for i in range_i:
        for j in range_j:
            z = img[i:i+8, j:j+8]
            z_vector = np.asmatrix(z.flatten('F')).T
            if __gee(z_vector) > 0:
                num_cat+=1
            else:
                num_grass+=1
    return num_grass, num_cat

for r in range(rounds):
    if r % (5 if lm == 0.5 or lm == 1 else 2) == 0:
        print(f"round: {r}")
        imsave(f'../pix/CW_Overlap_{int(lm*10)}_r{r}.png', 255*classify(img))

    img_prev = img.copy()
    grad = np.zeros((M,N))
    for i in range_i:
        for j in range_j:
            z = img[i:i+8, j:j+8]
            z_vector = np.asmatrix(z.flatten('F')).T
            grad[i:i+8, j:j+8] += np.reshape(gradient(z_vector, lm), (8,8),
            ↪  order='F')

    grad += 2 * (img - img_orig)
    img = np.clip(img - alpha * grad, 0.0, 1.0)
    change = np.linalg.norm(img - img_prev)
    if(change < 0.01):
        print(f"finished, total round: {r}")
        break;


cnts = cntclass(img)
print(f"number of grass patches {cnts[0]}, cat patches {cnts[1]}")
change = np.linalg.norm(img - img_orig)
print(f"Frobenius norm: {change}")

imsave(f'../pix/CW_Overlap_{int(lm*10)}.png', img)
imsave(f'../pix/CW_Overlap_{int(lm*10)}_perturbation.png',img-img_orig)
imsave(f'../pix/CW_Overlap_{int(lm*10)}_classified_perturb.png',
↪  classify(img))
```

```python
if __name__ == "__main__":
    cat_info, grass_info = get_info()
    CWAttack(cat_info, grass_info, readImg('../data/cat_grass.jpg'), lm = 0.5)
    CWAttack(cat_info, grass_info, readImg('../data/cat_grass.jpg'), lm = 1.0)
    CWAttack(cat_info, grass_info, readImg('../data/cat_grass.jpg'), lm = 5)
```