# Homework 5: Adversarial Attacks

Spring 2019
(Due: April 5, 2019)

## Objective

The objective of this project is twofold:

(a) Understanding the details of how some of the popular adversarial attacks operate on a Gaussian classifier

(b) Implementing the Carlini-Wagner attack on a Gaussian classifier

## Important Concepts and Background Information

### Adversarial Attack Methods

Table 1 below summarizes the adversarial attacks in the **2-class** case we studied in lecture.

| Attack Name | Optimization Problem | Solution/Iterate |
|---|---|---|
| Min $l_2$ attack, linear | $\underset{\boldsymbol{x}}{\text{argmin}}\ \lVert\boldsymbol{x}-\boldsymbol{x}_0\rVert_2^2$ subject to $\boldsymbol{w}^T\boldsymbol{x}+w_0=0$ (1) | $\boldsymbol{x}^* = \boldsymbol{x}_0 - \left(\frac{\boldsymbol{w}^T\boldsymbol{x}_0+w_0}{\lVert\boldsymbol{w}\rVert_2^2}\right)\boldsymbol{w}$ (2) |
| Min $l_\infty$ attack, linear | $\underset{\boldsymbol{x}}{\text{argmin}}\ \lVert\boldsymbol{x}-\boldsymbol{x}_0\rVert_\infty$ subject to $\boldsymbol{w}^T\boldsymbol{x}+w_0=0$ (3) | $\boldsymbol{x}^* = \boldsymbol{x}_0 - \left(\frac{\boldsymbol{w}^T\boldsymbol{x}_0+w_0}{\lVert\boldsymbol{w}\rVert_1}\right)\text{sign}(\boldsymbol{w})$ (4) |
| DeepFool Attack, general | $\underset{\boldsymbol{x}}{\text{argmin}}\ \lVert\boldsymbol{x}-\boldsymbol{x}_0\rVert_2^2$ subject to $g(\boldsymbol{x})=0$ (5) | $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \left(\frac{g(\boldsymbol{x}^{(k)})}{\lVert\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})\rVert_2^2}\right)\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})$ (6) |
| Max allowable $l_\infty$ attack, linear | $\underset{\boldsymbol{x}}{\text{argmin}}\ \boldsymbol{w}^T\boldsymbol{x}+w_0$ subject to $\lVert\boldsymbol{x}-\boldsymbol{x}_0\rVert_\infty<\eta$ (7) | $\boldsymbol{x}^* = \boldsymbol{x}_0 - \eta\,\text{sign}(\boldsymbol{w})$ (8) |
| Fast Gradient Sign Method (FGSM), general, $l_\infty$ | $\max_{\boldsymbol{r}}\ \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\boldsymbol{w})^T\boldsymbol{r} + J(\boldsymbol{x}_0;\boldsymbol{w})$ subject to $\lVert\boldsymbol{r}\rVert_\infty<\eta$ (9) | $\boldsymbol{r} = -\eta\,\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\boldsymbol{w}))$, $\boldsymbol{x}^* = \boldsymbol{x}_0 - \boldsymbol{r}$ (10) |
| Fast Gradient Sign Method (FGSM), general, $l_2$ | $\max_{\boldsymbol{r}}\ \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\boldsymbol{w})^T\boldsymbol{r} + J(\boldsymbol{x}_0;\boldsymbol{w})$ subject to $\lVert\boldsymbol{r}\rVert_2<\eta$ (11) | $\boldsymbol{r} = -\eta\,\frac{\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\boldsymbol{w})}{\lVert\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\boldsymbol{w})\rVert_2}$, $\boldsymbol{x}^* = \boldsymbol{x}_0 - \boldsymbol{r}$ (12) |

| | | |
|---|---|---|
| Iterative Fast Gradient Sign Method (I-FGSM), general, $l_\infty$ | $\boldsymbol{x}^{(k+1)} = \underset{0\le\boldsymbol{x}\le 1}{\text{argmax}}\, \nabla_{\boldsymbol{x}} J(\boldsymbol{x}^{(k)};\boldsymbol{w})^T\boldsymbol{x}$ subject to $\|\boldsymbol{x}-\boldsymbol{x}_0\|_\infty < \eta$ (13) | $\boldsymbol{x}^{(k+1)} = \mathcal{P}_{[0,1]}\{\boldsymbol{x}_0 + \eta\,\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}^{(k)};\boldsymbol{w}))\}$ (14) |
| Regul'n-based Attack, linear | $\underset{\boldsymbol{x}}{\text{argmin}}\, \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|_2^2 + \lambda(\boldsymbol{w}^T\boldsymbol{x}+w_0)$ (15) | $\boldsymbol{x}^* = \boldsymbol{x}_0 - \lambda\boldsymbol{w}$ (16) |
| Carlini-Wagner (CW) Attack | $\underset{\boldsymbol{x}}{\text{argmin}}\,\varphi(\boldsymbol{x})$, 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 $j\neq t$ (17) | $\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha_k\nabla_{\boldsymbol{x}}\varphi(\boldsymbol{x}^{(k)})$ $\nabla_{\boldsymbol{x}}\varphi(\boldsymbol{x}) = 2(\boldsymbol{x}-\boldsymbol{x}_0) + \lambda\mathbb{I}\{g_j(\boldsymbol{x})-g_t(\boldsymbol{x})>0\}\times(\nabla_{\boldsymbol{x}}(g_j(\boldsymbol{x})-g_t(\boldsymbol{x})))$ (18) |

Table 1: Adversarial Attacks on 2-Class Classifier

To make this document more self-contained, the following are some elaborations on the content of table 1; the lecture notes are the better reference if you prefer more detail:

1. We denote $\mathbb{R}^d$ as the feature space of any classification algorithm we will discuss in this project;
2. We assume the decision boundary of the linear classifiers are of the form $\{\boldsymbol{x}\in\mathbb{R}^d | \boldsymbol{w}^T\boldsymbol{x}+w_0=0\}$;
3. For $\boldsymbol{x}=(x_1,x_2,...,x_d)$,
$$\text{sign}(\boldsymbol{x}) = (\text{sign}(x_1),\text{sign}(x_2),...,\text{sign}(x_d)) \in \{+1,-1\}^d \tag{19}$$
4. In (5), the function $g$ defines the decision boundary of the classifier: $\{\boldsymbol{x}\in\mathbb{R}^d | g(\boldsymbol{x})=0\}$;
5. For the FGSM and IFGSM methods, the function $J$ is some loss function of interest, for instance, for a 2-class linear classifier, $J(\boldsymbol{x};\boldsymbol{w}_j-\boldsymbol{w}_t) = (\boldsymbol{w}_j-\boldsymbol{w}_t)^T\boldsymbol{x} + (w_{j,0}-w_{t,0})$ can be considered a loss function. See lecture notes for details;
6. In (18), $\alpha_k$ denotes the step size of the iterate, coming from the gradient descent algorithm.

This project is divided into two main sections. The first section is exercise 1, in which we will theoretically investigate how the adversarial attacks presented above operate on a 2-class Gaussian classifier; this will give us some concrete intuition of the attacks' mechanisms. The second section consists of exercises 2 and 3, in which we implement the CW attack on the two versions of the Gaussian classifier (cat-grass classifier) we trained and tested in project 4. We will see the differences of the attack on the two versions, and in addition, the power of the attack, in the sense that the attacked images will look very similar to the original image to us, but they can fool the classifiers.

## Exercise 1: Adversarial Attacks on Gaussian Classifier - Theory

To get started with this project, let us derive the adverarial attacks' solutions/iterates on 2-class Gaussian classifiers first, and observe some interesting theoretical properties of these attacks.

We recall the general form of the decision boundary of a 2-class Gaussian classifier:

$$\{\boldsymbol{x}\in\mathbb{R}^d |\, g(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T(\boldsymbol{W}_j-\boldsymbol{W}_t)\boldsymbol{x} + (\boldsymbol{w}_j-\boldsymbol{w}_t)^T\boldsymbol{x} + (w_{j,0}-w_{t,0}) = 0\}, \quad j\neq t \tag{20}$$

where

$$\boldsymbol{W}_i = \boldsymbol{\Sigma}_i^{-1},\ \boldsymbol{w}_j = -\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i,\ w_{i,0} = \frac{1}{2}\boldsymbol{\mu}_i\boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i + \frac{1}{2}\log(|\boldsymbol{\Sigma}_i|) - \log(\pi_i) \tag{21}$$

$\boldsymbol{\Sigma}_i$ is the covariance matrix of class $i$, $\boldsymbol{\mu}_i$ is the mean vector of class $i$, and $\pi_i$ is the prior probability of class $i$. In the case of equal covariance matrix $\boldsymbol{\Sigma}$ among the two classes, (20) becomes linear

$$\{\boldsymbol{x} \in \mathbb{R}^d | \, g(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0\} \tag{22}$$

where

$$\boldsymbol{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_t), \; w_0 = -\frac{1}{2}(\boldsymbol{\mu}_j + \boldsymbol{\mu}_t)\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_j - \boldsymbol{\mu}_t) + \log(\frac{\pi_j}{\pi_t}) \tag{23}$$

**Tasks**:

(a) We first study the minimum-norm attacks.

  (i) Derive the minimum $l_2$ and $l_\infty$ attacks in the linear case, based on (22).

  (ii) Derive the iterates that solve the DeepFool attack, based on (20).

  (iii) Find an example decision boundary where the DeepFool iterates **never** converge. A one-dimensional $g : \mathbb{R} \to \mathbb{R}$ is sufficient. Feel free to include visual illustrations.

(b) Now the maximum-allowable attacks.

  (i) Derive the maximum allowable $l_\infty$ attack in the linear case, based on (22).

  (ii) Derive the FGSM attack with $J(\boldsymbol{x}) = -g(\boldsymbol{x})$, based on (20).

  (iii) Derive the iterates of the I-FGSM attack with $J(\boldsymbol{x}) = -g(\boldsymbol{x})$, based on (20).

(c) Finally, we examine the regularization based attacks.

  (i) Derive the regularization-based attack in the linear case, based on (22).

  (ii) Derive the iterates of the CW attack, based on (20).

## Exercise 2: CW Attack on Gaussian Classifier - Non-Overlapping Patches

The goal of this exercise is to carry out the CW attack on the version of the cat-grass Gaussian classifier which operates on non-overlapping patches you trained and tested in project 4. Before going straight to coding, let us examine the problem setup first.

To attack the classifier, we just need to perturb each non-overlapping patch in the image independently using the CW attack method. More specifically, we are essentially solving for

$$\boldsymbol{x}_i^* = \underset{\boldsymbol{x}}{\operatorname{argmin}} \, \|\boldsymbol{x} - \boldsymbol{x}_{i,0}\|_2^2 + \lambda \, \max(g_j(\boldsymbol{x}) - g_t(\boldsymbol{x}), 0) \tag{24}$$

where $\boldsymbol{x}_{i,0} \in \mathbb{R}^{8\times 8}$ is the vector representing the $i^{th}$ patch in the original cat-grass image, and $\boldsymbol{x}_i^* \in \mathbb{R}^{8\times 8}$ is the optimally perturbed $i^{th}$ patch vector.

Let us now implement the attack. Note that the algorithm can be implemented in a variety of ways, parts (a) and (b) below only outlines one way of implementation, and provide you with the necessary parameters and conventions we adopt; feel free to implement the algorithm in any way you like.

**Tasks**:

(a) The CW attack's optimization problem is solved by gradient descent; examine the expression (18) carefully. To get started, implement the calculation of the gradient of the objective function first, based on the results you obtained from exercise 1(c)(ii). Use regularization coefficient $\lambda = 5$ for now. You can fill in the blanks of the following code if you prefer.

```
def gradient(patch_vec, lambda, target_index, other necessary parameters):
    # 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)
    return grad
```

(b) Write a function that implements the gradient descent algorithm on every non-overlapping patch in the image, using results from (a). Use $\alpha = 0.0001$. Terminate when the number of iterations reaches 300, or when the size of change $\|\boldsymbol{X}^{(k+1)} - \boldsymbol{X}^{(k)}\|_F$ is smaller than 0.001. You can fill in the blanks of the following code if you prefer.

```
def CW_attack(img_matrix, lambda, target_index, other necessary parameters)
    # Preprocess the data, and set up parameters
    # Start gradient descent
    while itr_num <= 300 and change >= 0.001:
        for i in range(M):
            for j in range(N):
                if i + 8 > M or j + 8 > N or i % 8 != 0 or j % 8 != 0:
                    continue
                # Perturb the patch with top left pixel (i,j) in img_matrix by
                # pert_patch = patch_vec + alpha*grad
        # Process the perturbed image matrix, display perturbed image, etc.
        change = np.linalg.norm(pert_img_curr - pert_img_prev)
        itr_num += 1
    # Return the attack image
```

**Remark**: $\|\cdot\|_F$ is the *Frobenius norm*, where $\|\boldsymbol{Y}\|_F = \|\widetilde{\boldsymbol{Y}}\|_2$, for $\widetilde{\boldsymbol{Y}}$ the vectorized $\boldsymbol{Y}$. The Python function `numpy.linalg.norm` can evaluate the Frobenius norm of a matrix.

**Hint**: The perturbed patch's entries might have values outside the bound $[0, 1]$. Use `numpy.clip` to avoid this problem, for example, write `np.clip(patch + alpha*grad, 0.0, 1.0)`. This is only one somewhat clumsy way to resolve the issue though, as there are many nicer functions to use, such as softmax, tanh, sigmoid, ReLU, ELU, but for the sake of simplicity, we will just use simple clipping.

(c) For $\lambda = 1, 5, 10$, obtain the following plots and data:

(i) The attack, i.e. the final perturbed image;

(ii) The perturbation added to the original image to obtain the attack;

(iii) Frobenius norm of the perturbation

(iv) The classifier's output on the attack, i.e., the number of patches classified as cat and as grass for the attack image;

(v) During gradient descent, plot the classifier's output on the perturbed image iterates per 50, 10, 5 iterations for $\lambda = 1, 5, 10$ respectively.

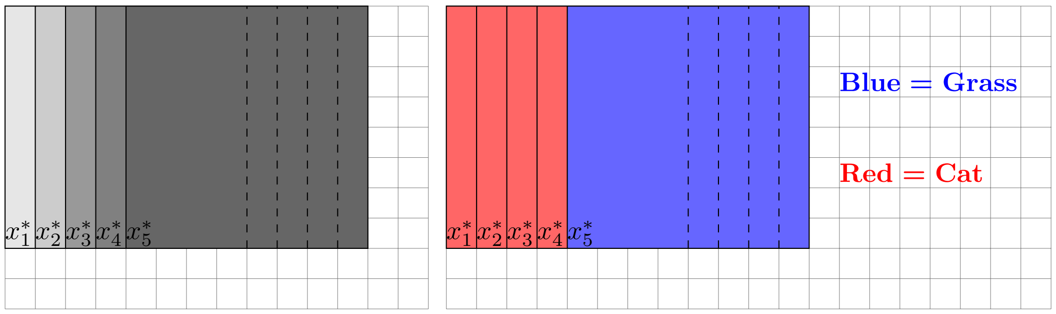
Comment on your plots and data. You can support your claims with more results than the above items.

(d) What do you observe when you increase the step size $\alpha$? Explain your answer. You can include results from the program to support your claims.

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

In this exercise, we implement the CW attack on the overlapping-patches version of the cat-grass Gaussian classifier. This exercise is very similar to the previous one in terms of implementation, however, we should be careful with the problem formulation of the attack.

Since the classifier operates on every patch in the image, still using (24) as our attack's optimization problem here is not a wise decision. The reason is that, even though we still want to force the resulting image to be as "close" to the original image as possible, we also want to take into consideration the "interference" between the overlapping patches. For example, directly using (24) on each overlapping patch could cause the situation in Figure 1 to happen.



$x_1^*$ $x_2^*$ $x_3^*$ $x_4^*$ $x_5^*$
$x_1^*$ $x_2^*$ $x_3^*$ $x_4^*$ $x_5^*$
Blue = Grass
Red = Cat

Figure 1: **Left**: five outputs from (24), all being perturbed from the **cat** class to the **grass** class by the algorithm; **Right**: classifier's decisions for the five patches, the first four are still classified as cat, contrary to our attack's intent, due to parts of them being overwritten by $x_5^*$

Consider the new optimization problem

$$
\begin{aligned}
\boldsymbol{X}^* &= \underset{\boldsymbol{X}}{\text{argmin}} \sum_{i=1}^{L} \left\{ \|\boldsymbol{P}_i(\boldsymbol{X}-\boldsymbol{X}_0)\|_2^2 + \lambda \max(g_j(\boldsymbol{P}_i\boldsymbol{X}) - g_t(\boldsymbol{P}_i\boldsymbol{X}), 0) \right\} \\
&= \underset{\boldsymbol{X}}{\text{argmin}} \|\boldsymbol{X}-\boldsymbol{X}_0\|_2^2 + \lambda \sum_{i=1}^{L} \max(g_j(\boldsymbol{P}_i\boldsymbol{X}) - g_t(\boldsymbol{P}_i\boldsymbol{X}), 0)
\end{aligned} \tag{25}
$$

where $\boldsymbol{X}_0 \in \mathbb{R}^{MN}$ is the **vectorized** original image, $L$ is the total number of patches in the image, $\boldsymbol{P}_i$ is the matrix that extracts the $i^{th}$ patch (vectorized) from the image vector. Notice that in the first equality of (25), each summand is the objective function in (24), but now we are summing over **all** of the patches; we are now attacking the whole image at once, instead of attacking independent patches. More specifically speaking, we simultaneously minimize the $l_2$ distance term for the whole image and the regularization term for every patch, thus we keep the perturbed image looking as "similar" to the original image as allowed, while making every patch in the perturbed image to be as "close" to the target class as possible. Note that this formulation counters the issue raised in Figure 1, as we are taking into consideration the "interference" among the overlapping patches now.

**Tasks**:

(a) Calculate the gradient of the objective function in the second line of (25) with respect to $\boldsymbol{X}$.

(b) Implement the CW attack on the overlapping patches classifier. For stopping criteria, use maximum iteration 300 and change in iterates 0.01. If your implementation followed the outline in the previous part, much of your code should be reusable.

(c) Obtain the plots and data in the same way you did in exercise 2(c), except this time, plot the perturbed image iterates per 5, 5, 2 iterations for $\lambda = 0.5, 1, 5$ respectively. Again, comment on your plots and data. Moreover, how does the quality of the attack compare to that of the last exercise in general? Explain.