# Chapter 3

# Adversarial Attack

Consider a data point $\boldsymbol{x}_0 \in \mathbb{R}^d$ belonging to class $\mathcal{C}_i$. Adversarial attack is a malicious attempt which tries to perturb $\boldsymbol{x}_0$ to a new data point $\boldsymbol{x}$ such that $\boldsymbol{x}$ is misclassified by the classifier. Goodfellow et al. made this explicit by showing examples perceptually indistinguishable but sufficient to fool a classifier (See Figure 3.1). This raises many important questions such as vulnerability of machine learning algorithms and the associated security threats. While many people believe that adversarial attack is unique to deep neural networks as these classifiers tend to overfit, adversarial attack is in fact an intrinsic property of all classifier. In this chapter, we will study the fundamentals of adversarial attack through various examples of linear classifiers. Our goal is to understand the source of these attacks, the geometry, and possible defensive strategies.

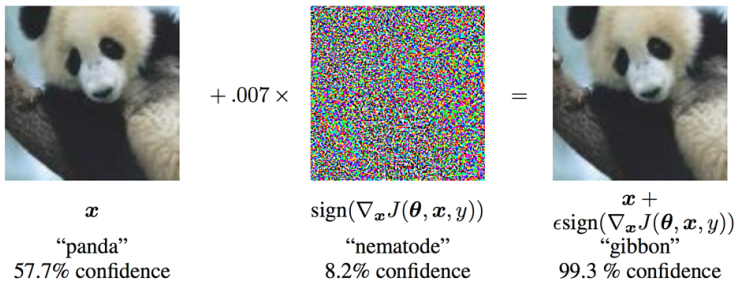


Figure 3.1: A classical example by Goodfellow et al 2014. By perturbing the input data $\boldsymbol{x}_0$ with a perceptually undistinguishable "noise", the perturbed data is misclassified with high confidence. We will discuss how to generate these "noise" pattern, and their geometric meanings.

## 3.1 Problem Formulation

### What is Adversarial Attack?

Adversarial attack is an malicious attempt to perturb a data point $\boldsymbol{x}_0 \in \mathbb{R}^d$ to another point $\boldsymbol{x} \in \mathbb{R}^d$ such that $\boldsymbol{x}$ belongs to certain **target** adversarial class. For example if $\boldsymbol{x}_0$ is

a feature vector of a `cat` image, by adversarial attack we meant to create another feature vector $\boldsymbol{x}$ which will be classifier as a `dog` (or another class specified by the attacker). In some scenarios, the goal may not be to push $\boldsymbol{x}_0$ to a specific target class $\mathcal{C}_t$, but just push it away from its original class $\mathcal{C}_i$. Attacks of this type is known as the **untargeted** attack. For the time being, let us focus on targeted attack first.

There are various approaches to move a data point $\boldsymbol{x}_0$ from $\mathcal{C}_i$ to $\mathcal{C}_t$. The most general definition of such mapping is to consider an operator $\mathcal{A}: \mathbb{R}^d \to \mathbb{R}^d$ such that $\boldsymbol{x} = \mathcal{A}(\boldsymbol{x}_0)$ is the perturbed data.

**Definition 1** (Adversarial Attack)**.** *Let $\boldsymbol{x}_0 \in \mathbb{R}^d$ be a data point belong to class $\mathcal{C}_i$. Define a target class $\mathcal{C}_t$. An **adversarial attack** is a mapping $\mathcal{A}: \mathbb{R}^d \to \mathbb{R}^d$ such that the perturbed data*

$$\boldsymbol{x} = \mathcal{A}(\boldsymbol{x}_0)$$

*is misclassified as $\mathcal{C}_t$.*

Among many adversarial attack models, the most commonly used one is the additive model, where we define $\mathcal{A}$ as a linear operator that adds perturbation to the input.

**Definition 2** (Additive Adversarial Attack)**.** *Let $\boldsymbol{x}_0 \in \mathbb{R}^d$ be a data point belong to class $\mathcal{C}_i$. Define a target class $\mathcal{C}_t$. An **additive** adversarial attack is an addition of a perturbation $\boldsymbol{r} \in \mathbb{R}^d$ such that the perturbed data*

$$\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{r}$$

*is misclassified as $\mathcal{C}_t$.*

Additive attacks are appealing for at least two reasons. First, an additive attack guarantees the input space remains unchanged. For example, if the input $\boldsymbol{x}_0$ is an image in $\mathbb{R}^d$, then the perturbed data $\boldsymbol{x}$ remains in $\mathbb{R}^d$ and still represents an image. A general mapping $\mathcal{A}(\cdot)$ may not preserve such relationship, e.g., $\mathcal{A}$ is a sub-sampling operator or a dense matrix that transforms the input space to a feature space. Second, additive attack allows interpretable analysis with simple geometry. A general mapping $\mathcal{A}(\cdot)$ could be very complicated to analyze. Therefore, unless specified, in the rest of this chapter we will refer adversarial attack exclusively to the additive attacks.

## Formulating Adversarial Attack

Since the primary goal of adversarial attack is to perturb a data point for misclassification, we need to first consider the input class $\mathcal{C}_i$ and the target class $\mathcal{C}_t$. The input class $\mathcal{C}_i$ is the class where $\boldsymbol{x}_0$ belongs to, and $\mathcal{C}_t$ is the class which we wish the attack data $\boldsymbol{x}$ to be.

 

Consider a multi-class scenario where we have classes $\mathcal{C}_1, \ldots, \mathcal{C}_k$. The decision boundaries of these $k$ classes are specified by the $k$ discriminant functions $g_i(\cdot)$ for $i = 1, \ldots, k$. If we want $\boldsymbol{x}$ to be classified as class $\mathcal{C}_t$, then the discriminant functions should satisfy the condition that

$$g_t(\boldsymbol{x}) \geq g_j(\boldsymbol{x}), \qquad \text{for all} \qquad j \neq t, \tag{3.1}$$

so that the target class $\mathcal{C}_t$ will have a discriminant value $g_t(\boldsymbol{x})$ greater than all other classes in the classifier. Rewriting the $k-1$ inequalities, we can equivalently express them as

$$g_t(\boldsymbol{x}) \geq \max_{j \neq t} \{g_j(\boldsymbol{x})\} \qquad \Longleftrightarrow \qquad \max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \leq 0. \tag{3.2}$$

Therefore, the goal of adversarial attack is to find $\boldsymbol{x}$ such that the inequality in Equation (3.2) is satisfied.

The inequality in Equation (3.2) defines a set of $\boldsymbol{x}$ that are allowed to be potential attacks. If this set is non-empty, then any element in the set could be used as the attack. However, some attacks are closer to the input $\boldsymbol{x}_0$ than others. In order to pick a unique attack, it is often necessary to post additional criteria in terms of the adversarial magnitude. That is, we want the perturbation as small as possible, or the perturbed data should stay as close to $\boldsymbol{x}_0$ as possible. Once such criteria are added to the problem, we will reach an optimization formulation. The following three definitions are the most commonly used definitions of attacks in the literature.

**Definition 3** (Minimum Norm Attack)**.** *The* ***minimum norm attack*** *finds a perturbed data* $\boldsymbol{x}$ *by solving the optimization*

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \|\boldsymbol{x} - \boldsymbol{x}_0\| \\ \text{subject to} \quad & \max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \leq 0, \end{aligned} \tag{3.3}$$

*where* $\|\cdot\|$ *can be any norm specified by the user.*

As shown in the definition, the goal of the minimum norm attack is to minimize the perturbation magnitude while ensuring the new data $\boldsymbol{x}$ is classified as $\mathcal{C}_t$. If the constraint set is empty, then we say that it is impossible to attack the classifier for a particular input $\boldsymbol{x}_0$.

An alternative to the minimum norm attack is the maximum allowable attack.

**Definition 4** (Maximum Allowable Attack)**.** *The* ***maximum allowable attack*** *finds a perturbed data* $\boldsymbol{x}$ *by solving the optimization*

$$\begin{aligned} \underset{\boldsymbol{x}}{\text{minimize}} \quad & \max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \\ \text{subject to} \quad & \|\boldsymbol{x} - \boldsymbol{x}_0\| \leq \eta, \end{aligned} \tag{3.4}$$

*where* $\|\cdot\|$ *can be any norm specified by the user, and* $\eta > 0$ *denotes the magnitude of the attack.*

 

In maximum allowable attack, the magnitude of the attack is upper bounded by $\eta$. That is, the attack cannot go beyond the range $\|\boldsymbol{x} - \boldsymbol{x}_0\| \leq \eta$. The objective function seeks $\boldsymbol{x}$ to make $\max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})$ as negative as possible. If the best $\boldsymbol{x}$ still cannot make the objective value negative, then we say that it is impossible to attack the classifier for a particular input $\boldsymbol{x}_0$.

The third formulation is an unconstrained optimization based on regularization.

**Definition 5** (Regularization-based Attack)**.** *The* ***regularization-based attack*** *finds a perturbed data $\boldsymbol{x}$ by solving the optimization*

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\| + \lambda \left(\max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})\right) \tag{3.5}$$

*where $\|\cdot\|$ can be any norm specified by the user, and $\lambda > 0$ is a regularization parameter.*

The parameter in the regularization-based attack is also known as the Lagrange multiplier. Intuitively, Equation (3.5) tries to simultaneously minimize two conflicting objectives $\|\boldsymbol{x} - \boldsymbol{x}_0\|$ and $\max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})$. Depending on the choice of $\lambda$, relative emphasis on the two terms can be controlled.

**Which Optimization?** While Equation (3.3), Equation (3.4) and Equation (3.5) offer three different ways of determining the adversarial attack (and hence three different solutions), it can be shown that for appropriately chosen $\eta$ and $\lambda$, the three optimizations are in fact equivalent in the sense that the solutions are identical. In other words, if we can solve one problem, we will be able to solve the other two problems provided we have the right $\eta$ and $\lambda$. Because of the equivalence between the optimization problems, we will spend most of the time on the minimum norm attack. There are two reasons.

- No Parameter. Unlike maximum allowable attack and regularization-based attack, minimum norm attack does not require user defined parameters $\lambda$ and $\eta$. This allows us to determine unique solutions independent of the parameters.
- Simple geometry. It is easier to interpret $\max_{j \neq t} \{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})$ as a constraint set rather an objective function, for minimizing $\|\boldsymbol{x} - \boldsymbol{x}_0\|$ over a set can be treated as a projection.

## Geometry of the Objective Function

The norm $\|\boldsymbol{x} - \boldsymbol{x}_0\|$ is measuring the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_0$. There are various options, with geometric interpretation shown in Figure 3.2.

- $\ell_0$-norm: $\varphi(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_0\|_0$, which gives the most sparse solution. Useful when we want to limit the number of attack pixels.
- $\ell_1$-norm: $\varphi(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_0\|_1$, which is a convex surrogate of the $\ell_0$-norm.

 

- $\ell_\infty$-norm: $\varphi(\boldsymbol{x}) = \|\boldsymbol{x}-\boldsymbol{x}_0\|_\infty$, which minimizes the maximum element of the perturbation.

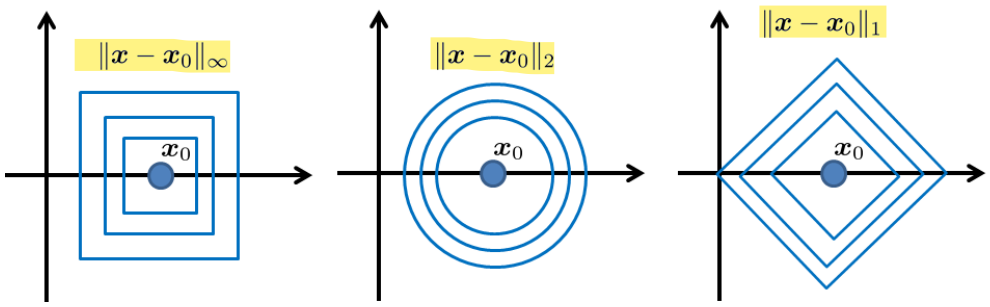


Figure 3.2: Geometry of different objective functions.

Besides changing the norm of the objective function, we can also post constraints to the objective. For example, in some applications we may want to limit the attack at certain spatial location. This can be achieved by adding a masking $\boldsymbol{M} \in \mathbb{R}^{m\times d}$ so that the distance becomes

$$\varphi(\boldsymbol{x}) = \|\boldsymbol{M}(\boldsymbol{x}-\boldsymbol{x}_0)\|^2. \qquad (3.6)$$

M can be 0 or 1

In other examples, the objective function may contain a regularization term $h(\boldsymbol{x})$ so that $\varphi$ becomes

$$\varphi(\boldsymbol{x}) = \|\boldsymbol{x}-\boldsymbol{x}_0\|^2 + \beta h(\boldsymbol{x}). \qquad (3.7)$$

## Geometry of the Constraints

The constraint in Equation (3.3) is a compact form of Equation (3.1). If we do not mind the tediousness of the constraints, we can show that $\Omega = \{\boldsymbol{x} \mid \max_{j\neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \leq 0\}$ is equivalent to

$$\Omega = \left\{ \boldsymbol{x} \;\middle|\; \begin{array}{l} g_1(\boldsymbol{x}) - g_t(\boldsymbol{x}) \leq 0 \\ g_2(\boldsymbol{x}) - g_t(\boldsymbol{x}) \leq 0 \\ \vdots \\ g_k(\boldsymbol{x}) - g_t(\boldsymbol{x}) \leq 0 \end{array} \right\} \qquad (3.8)$$

Depending on the nature of the discriminant function $g_i(\boldsymbol{x})$, the geometry of $\Omega$ could be polygon, convex, concave, or arbitrary.

To understand the behavior of the constraint, we show in Figure 3.3 a typical example of the constraint set. Given the discriminant function $g_i(\boldsymbol{x})$ for $i = 1, \ldots, k$, the decision boundary between class $\mathcal{C}_i$ and $\mathcal{C}_t$ is $g_i(\boldsymbol{x}) - g_t(\boldsymbol{x}) = 0$. Depending on where $\boldsymbol{x}_0$ is located, the decision boundaries will change and hence the inequalities in $\Omega$ change with $\boldsymbol{x}_0$. If a class $\mathcal{C}_i$ does not have any direct contact with $\mathcal{C}_t$, a straight inequality will hold.

 

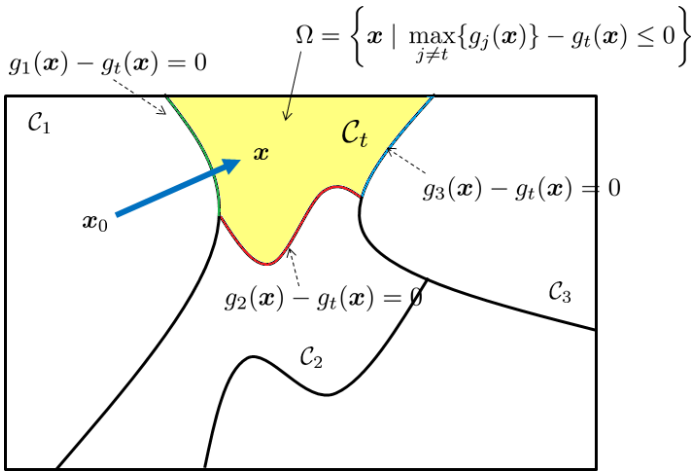


Figure 3.3: Geometry of the constraint set $\Omega$. The decision boundaries are given by $g_i(\boldsymbol{x}) - g_t(\boldsymbol{x}) = 0$. The set $\Omega$ constitute all $\boldsymbol{x}$ such that every $\boldsymbol{x} \in \Omega$ belongs to $\mathcal{C}_t$.

**Caution**: Note that we when say $\boldsymbol{x} \in \Omega$, we are referring to a point which simultaneously satisfies all the decision boundary constraints. It is therefore incorrect to say pick an index $i^*$ which maximizes $g_i(\boldsymbol{x})$ for a particular $\boldsymbol{x}$, and simply study $g_{i^*}(\boldsymbol{x}) - g_t(\boldsymbol{x}) \leq 0$. In general, the index $i^*$ must be a function of $\boldsymbol{x}$, i.e., $i^*(\boldsymbol{x})$. Therefore, the correct set is

$$\Omega = \left\{ \boldsymbol{x} \mid g_{i^*(\boldsymbol{x})}(\boldsymbol{x}) - g_t(\boldsymbol{x}) \leq 0 \right\}.$$

The reason for $i^*$ to depend on $\boldsymbol{x}$ is that the maximum is point-wise for every $\boldsymbol{x}$. As $\boldsymbol{x}$ changes, the index corresponding to the maximizer also changes. The only exception is the linear classifier for two classes, because the decision boundary is a straight line. In this case, $\max_{j \neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \leq 0$ is simplified to $g_{i^*}(\boldsymbol{x}) - g_t(\boldsymbol{x}) \leq 0$ because for a two-class problem the index is either $i^*$ or $t$. For non-linear classifiers, or linear classifiers with more than two classes, such simplification generally does not hold except for some special cases.

## Geometry of the Constraints for Linear Classifiers

To gain insights about these constraints, we consider a $k$-class linear classifier. In this case, each discriminant function takes the form

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i,0}. \tag{3.9}$$

The decision boundary between the $i$-th class and the $t$-th class is therefore

$$g(\boldsymbol{x}) = (\boldsymbol{w}_i - \boldsymbol{w}_t)^T \boldsymbol{x} + w_{i,0} - w_{t,0} = 0. \tag{3.10}$$

Since we want the perturbed data $\boldsymbol{x}$ to live in $\mathcal{C}_t$, the constraint set becomes

$$\begin{bmatrix} \boldsymbol{w}_1^T - \boldsymbol{w}_t^T \\ \vdots \\ \boldsymbol{w}_{t-1}^T - \boldsymbol{w}_t^T \\ \boldsymbol{w}_{t+1}^T - \boldsymbol{w}_t^T \\ \vdots \\ \boldsymbol{w}_k^T - \boldsymbol{w}_t^T \end{bmatrix} \boldsymbol{x} + \begin{bmatrix} w_{1,0} - w_{t,0} \\ \vdots \\ w_{t-1,0} - w_{t,0} \\ w_{t+1,0} - w_{t,0} \\ \vdots \\ w_{k,0} - w_{t,0} \end{bmatrix} \leq \boldsymbol{0} \quad \Leftrightarrow \quad \boldsymbol{A}^T\boldsymbol{x} \leq \boldsymbol{b} \tag{3.11}$$

where $\boldsymbol{A} = [\boldsymbol{w}_1 - \boldsymbol{w}_t, \ldots, \boldsymbol{w}_k - \boldsymbol{w}_t] \in \mathbb{R}^{d\times(k-1)}$, and $\boldsymbol{b} = [w_{t,0} - w_{1,0}, \ldots, w_{t,0} - w_{k,0}]^T$. This is summarized in the following Lemma.

**Lemma 1** (Constraint Set of Linear Classifier)**.** *Consider a $k$-class linear classifier with discriminant function $g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}$ for $i = 1, \ldots, k$. Define*

$$\boldsymbol{A} = [\boldsymbol{w}_1 - \boldsymbol{w}_t, \ldots, \boldsymbol{w}_k - \boldsymbol{w}_t] \in \mathbb{R}^{d\times(k-1)}, \quad \textit{and} \quad \boldsymbol{b} = [w_{t,0} - w_{1,0}, \ldots, w_{t,0} - w_{k,0}]^T \in \mathbb{R}^{k-1},$$

*Then, the constraint set is*

$$\Omega = \{\boldsymbol{x} \mid \boldsymbol{A}^T\boldsymbol{x} \leq \boldsymbol{b}\}. \tag{3.12}$$

For linear classifiers, the constraint set $\Omega$ defines a $d$-dimensional polytope. Figure 3.4 shows one example. The normal vectors of each decision boundary is the column of the $\boldsymbol{A}$ matrix, and the $i$-th decision boundary is

$$\boldsymbol{a}_i^T\boldsymbol{x} = b_i. \tag{3.13}$$

Since any polytope is convex, the constraint set $\Omega$ is convex. This can also be proved easily by letting $\boldsymbol{x}_1 \in \Omega$ and $\boldsymbol{x}_2 \in \Omega$. For any $0 \leq \lambda \leq 1$, we have

$$\boldsymbol{A}^T(\lambda\boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2) = \lambda\boldsymbol{A}^T\boldsymbol{x}_1 + (1-\lambda)\boldsymbol{A}^T\boldsymbol{x}_2 \leq \lambda\boldsymbol{b} + (1-\lambda)\boldsymbol{b} = \boldsymbol{b}.$$

Thus, $\lambda\boldsymbol{x}_1 + (1-\lambda)\boldsymbol{x}_2 \in \Omega$ as well.

The set $\Omega$ is always non-empty if the classifier is a **linear machine**. A linear machine defines the decision based on Equation (3.1), i.e., decide class $\mathcal{C}_t$ if $g_t(\boldsymbol{x}) \geq g_j(\boldsymbol{x})$ for every $j \neq t$. Therefore, the entire input space is **partitioned** into $n$ decision boundaries, and each partition can contain up to $k-1$ decision boundaries. Since every partition is not empty by definition, $\Omega$ must also be non-empty.

An immediate corollary of the lemma is that for linear classifiers, the adversarial attack problem is essentially a quadratic minimization

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \quad \text{subject to} \quad \boldsymbol{A}^T\boldsymbol{x} \leq \boldsymbol{b}, \tag{3.14}$$

 

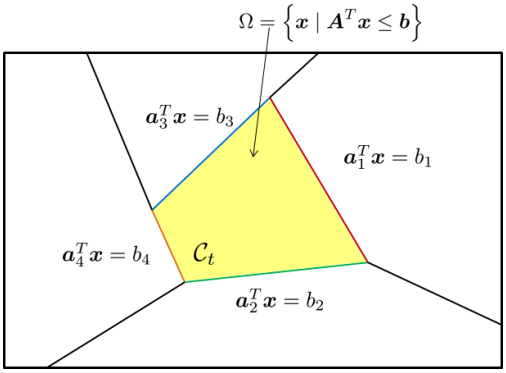


Figure 3.4: Geometry of the constraint set $\Omega$ for a linear classifier. The constraint set $\Omega$ is now a polygon with decision boundaries defined by $\boldsymbol{a}_i^T\boldsymbol{x} = b_i$, where $\boldsymbol{a}_i = \boldsymbol{w}_i - \boldsymbol{w}_t$ and $b_i = w_{i0} - w_{t0}$.

which can be solved easily using convex programming techniques. The following examples illustrates how to formulate an $\ell_1$-norm attack via linear programming.

**Example**. (Linear programming formulation for $\ell_1$-norm attack.) If we modify the $\ell_2$-norm objective to an $\ell_1$-norm, i.e.,

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|_1 \quad \text{subject to} \quad \boldsymbol{A}^T\boldsymbol{x} \leq \boldsymbol{b}, \tag{3.15}$$

then the problem can be formulated via a linear programming. To see this, we first rewrite Equation (3.15) by letting $\boldsymbol{r} = \boldsymbol{x} - \boldsymbol{x}_0$ so that we have

$$\underset{\boldsymbol{r}}{\text{minimize}} \quad \|\boldsymbol{r}\|_1 \quad \text{subject to} \quad \boldsymbol{A}^T\boldsymbol{r} \leq \widetilde{\boldsymbol{b}},$$

where $\widetilde{\boldsymbol{b}} = \boldsymbol{b} - \boldsymbol{A}^T\boldsymbol{x}_0$. Now, define $\boldsymbol{r}_+$ and $\boldsymbol{r}_-$ be the positive and negative parts of $\boldsymbol{r}$ such that $\boldsymbol{r} = \boldsymbol{r}_+ - \boldsymbol{r}_-$. Then, $\|\boldsymbol{r}\|_1 = \boldsymbol{r}_+ + \boldsymbol{r}_-$, and so the optimization becomes

$$\begin{aligned} &\underset{\boldsymbol{r}_+, \boldsymbol{r}_-}{\text{minimize}} \quad \boldsymbol{r}_+ + \boldsymbol{r}_- \\ &\text{subject to} \quad \begin{bmatrix} \boldsymbol{A}^T & -\boldsymbol{A}^T \end{bmatrix} \begin{bmatrix} \boldsymbol{r}_+ \\ \boldsymbol{r}_- \end{bmatrix} \leq \widetilde{\boldsymbol{b}}, \quad \boldsymbol{r}_+ \geq 0, \quad \text{and} \quad \boldsymbol{r}_- \geq 0. \end{aligned} \tag{3.16}$$

Note that this is a standard linear programming problem, which can be solved efficiently using the Simplex method.

**Caution**. The constraint set $\Omega$ could be empty if the classifier is not a linear machine. For example, if the decision rule is based on checking if $g_i(\boldsymbol{x}) > 0$, or if pairwise $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x})$, then it is possible to have unclassified regions. Linear machines do not have the issue because it checks $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x})$ for every $j \neq i$. See Duda-Hart-Stork Section 5.2.2 for discussion.

 

## 3.2 Geometry of the Attack

### A Projection Perspective

The geometry of adversarial attack is specified by two factors. First is the distance metric, which defines whether we like to use a ball ($\ell_2$-norm), a diamond ($\ell_1$-norm), or square ($\ell_\infty$-norm). Second is the feasible set $\Omega$ which specifies the decision boundary between the original class where $\boldsymbol{x}_0$ is positioned at, and the targeted class where $\boldsymbol{x}$ should be sent to. If the objective is $\ell_2$, then we can re-formulate the minimum-norm attack problem as a projection.

**Theorem 1** (Minimum-Norm Attack as a Projection). *The adversarial attack formulated by Equation (3.3) is equivalent to the projection*

$$\begin{aligned} \boldsymbol{x}^* &= \underset{\boldsymbol{x}\in\Omega}{\text{argmin}} \ \|\boldsymbol{x}-\boldsymbol{x}_0\|^2, \quad \text{where} \quad \Omega = \{\boldsymbol{x} \mid \max_{j\neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \leq 0\}, \\ &= \mathcal{P}_\Omega(\boldsymbol{x}_0), \end{aligned} \tag{3.17}$$

*where $\mathcal{P}_\Omega(\cdot)$ denotes the projection onto the set $\Omega$.*

Equation (3.17) defines a projection of $\boldsymbol{x}_0$ onto the set $\Omega$. Figure 3.5 illustrates the idea. Suppose we are originally at $\boldsymbol{x}_0$ and we like to move to class $\mathcal{C}_t$. In this particular illustration, the class being considered is $\mathcal{C}_{i^*}$. By solving the minimization problem, we effectively project $\boldsymbol{x}_0$ onto the decision boundary between $\mathcal{C}_t$ and $\mathcal{C}_{i^*}$. The direction of the perturbation is the normal vector of the decision boundary at $\boldsymbol{x}$. The magnitude of the perturbation is the shortest distance between the input $\boldsymbol{x}_0$ and the target set $\mathcal{C}_t$. Note that because of the projection geometry, the optimal solution $\boldsymbol{x}^*$ has to stay on the decision boundary and cannot be inside the interior.

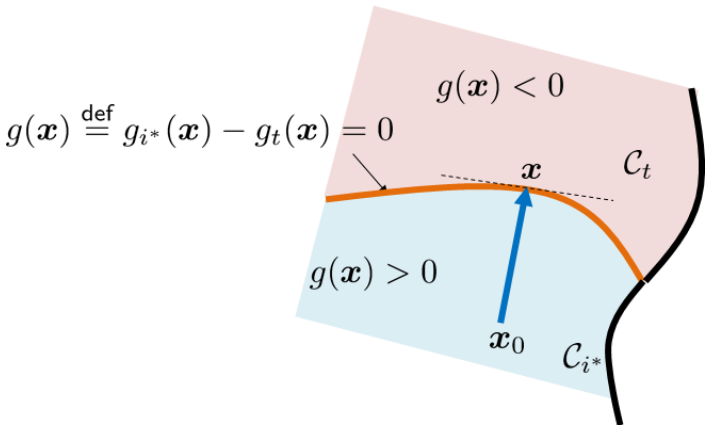


Figure 3.5: Geometry of Equation (3.3): Given an input data point $\boldsymbol{x}_0$, our goal is to send $\boldsymbol{x}_0$ to a targeted class $\mathcal{C}_t$ by minimizing the distance between $\boldsymbol{x}$ and $\boldsymbol{x}_0$. The decision boundary is characterized by $g(\boldsymbol{x}) = g_{i^*}(\boldsymbol{x}) - g_t(\boldsymbol{x})$. The optimal solution is the projection of $\boldsymbol{x}_0$ onto the decision boundary.

 

What if we insist of making a bigger step than needed so that $\boldsymbol{x}$ will enter the interior of the target class? In theory this is possible if we make

$$\boldsymbol{x} = \boldsymbol{x}_0 + \alpha(\mathcal{P}_\Omega(\boldsymbol{x}_0) - \boldsymbol{x}_0), \tag{3.18}$$

for some step size $\alpha > 0$, where the residue vector $\boldsymbol{r} = \mathcal{P}_\Omega(\boldsymbol{x}_0) - \boldsymbol{x}_0$ defines the direction of perturbation. Geometrically, the new data point $\boldsymbol{x}$ can be thought of moving the input $\boldsymbol{x}_0$ along the direction $\boldsymbol{r}$ with certain step size $\alpha$.

For complex geometry, attack based on Equation (3.18) could potentially cause problems if $\alpha$ is inappropriately chosen, e.g., landing on a class different from $\mathcal{C}_t$. Figure 3.6 illustrates two common examples. In the first example, the decision boundary is a hyperbola with decision regions $\mathcal{C}_i$ on both sides of the space. For excessively large $\alpha$, the perturbation will bypass the target class $\mathcal{C}_t$ and land on the original class $\mathcal{C}_i$ (though on the other side of the hyperbola). The second example shows a multi-class situation. Again, for excessively large $\alpha$, we may shoot over the target class and land on a third party class.

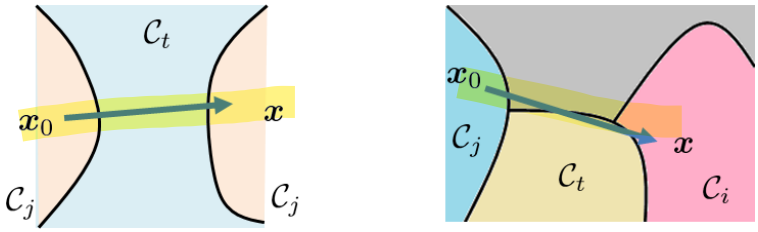


Figure 3.6: For inappropriately chosen step size $\alpha$, the data point $\boldsymbol{x}_0$ can be sent to a wrong class. [Left] The decision boundary is such that $\mathcal{C}_j$ occupies both sides of the space whereas $\mathcal{C}_t$ is in the middle. $\boldsymbol{x}$ is classified as the original class $\mathcal{C}_j$. [Right] In a multi-class decision, $\boldsymbol{x}_0$ is perturbed to $\mathcal{C}_i$. While this is not the targeted class $\mathcal{C}_t$, the new class $\mathcal{C}_i$ is still a wrong class.

## Targeted and Untargeted Attack

When considering adversarial attacks, there often two types of attack objectives: targeted or untargeted. By **targeted attack** we meant to move a data point $\boldsymbol{x}_0$ to the target class $\mathcal{C}_t$. This, however, does not require a specific class $\mathcal{C}_i$ for $\boldsymbol{x}_0$. Or in other words, as long as we can move $\boldsymbol{x}_0$ to $\mathcal{C}_t$, we have achieved the goal. We do not care about whether $\boldsymbol{x}_0$ is originally classified correctly, although in most cases we would wish $\boldsymbol{x}_0$ to be classified correctly.

Since targeted attack only concerns about the destination, the constraint is given by

$$\Omega = \{\boldsymbol{x} \mid \max_{j \neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \leq 0\}, \tag{3.19}$$

which is essentially the set of $\boldsymbol{x}$ living in $\mathcal{C}_t$.

An **untargeted attack** is opposite to targeted attack in the sense that we like to move $\boldsymbol{x}_0$ away from its current class. That is, if $\boldsymbol{x}_0 \in \mathcal{C}_i$, then we want $\boldsymbol{x} \notin \mathcal{C}_i$. Same as targeted

 

attack, whether $\boldsymbol{x}_0$ is classified correctly is not a concern, although we would wish this to happen. The constraint set of an untargeted attack is given by

$$\Omega = \{\boldsymbol{x} \mid g_i(\boldsymbol{x}) - \min_{j \neq i}\{g_j(\boldsymbol{x})\} \leq 0\}, \tag{3.20}$$

which says that $g_i(\boldsymbol{x}) \leq g_j(\boldsymbol{x})$ for every $j \neq i$, i.e., $\boldsymbol{x}$ cannot be classified as $\mathcal{C}_i$.

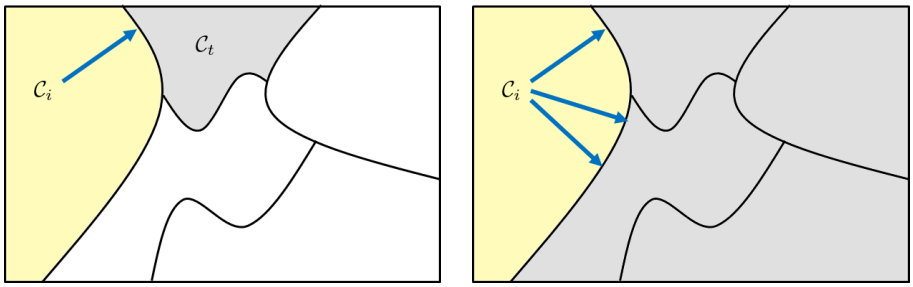


Figure 3.7: [Left] Targeted attack: The attack has to be specific from $\mathcal{C}_i$ to $\mathcal{C}_t$. [Right] Untargeted attack: The attack vector can point to anywhere outside $\mathcal{C}_i$.

## White Box and Black Box Attack

The majority of the attacks we see are **white box** attacks. These attacks assume complete knowledge about the classifier, i.e., we know exactly the discriminant functions $g_i(\boldsymbol{x})$ for every $i$. For linear classifiers, this requires knowing all the decision boundaries $\boldsymbol{w}_i$ and $w_{i0}$. For deep neural networks, this requires knowing all the weight values and network structure. Therefore, white box attack can be considered as the best possible attack we can ever design, or the worst scenario that a classifier should face.

The opposite to white box attacks is the **black box** attacks. However, if we know absolutely nothing about the classifier then it would be nonsense to ask for a meaningful attack except using random noise. Therefore, by black box we meant that we are able to probe the classifier for a fixed number of trials, say $M$ times. In each of the $m$-th trial, we input a sample $\boldsymbol{x}^{(m)}$ and record its classification result. In this case, the constraint set becomes

$$\Omega = \{\boldsymbol{x} \mid \max_{j \neq t}\{\widehat{g}_j(\boldsymbol{x})\} - \widehat{g}_t(\boldsymbol{x}) \leq 0\}, \tag{3.21}$$

where $\widehat{g}_i$ has been evaluated at $\{\widehat{g}_i(\boldsymbol{x}^{(1)}), \widehat{g}_i(\boldsymbol{x}^{(2)}), \ldots, \widehat{g}_i(\boldsymbol{x}^{(M)})\}$.

 

## 3.3 Generating Attacks

### A. Minimum-Norm Attack

In order to gain insight of the adversarial attack, it is useful to consider a simple linear classifier with only two classes. A linear classifier has a discriminant function

$$g_i(\boldsymbol{x}) = \boldsymbol{w}_i^T \boldsymbol{x} + w_{i0}, \tag{3.22}$$

and therefore by defining $\boldsymbol{w} \overset{\text{def}}{=} \boldsymbol{w}_i - \boldsymbol{w}_t$ and $w_0 \overset{\text{def}}{=} w_{i0} - w_{t0}$, we can show that

$$g_i(\boldsymbol{x}) - g_t(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + w_0. \tag{3.23}$$

Recalling Equation (3.3), the adversarial attack can be formulated as

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{minimize}} && \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \\ &\text{subject to} && g_i(\boldsymbol{x}) - g_t(\boldsymbol{x}) = 0. \end{aligned} \tag{3.24}$$

In this optimization, we simplify $\max_{j \neq t} \{g_j(\boldsymbol{x})\}$ to $g_i(\boldsymbol{x})$ because there are only two classes in our problem. If the index is not $t$, then it has to be $i$. We also replaced the inequality $g_i(\boldsymbol{x}) - g_t(\boldsymbol{x}) \leq 0$ as an equality, because the projection theorem in Equation (3.17) suggests that the solution must be on the decision boundary.

**Theorem 2** (Minimum $\ell_2$ Norm Attack for Two-Class Linear Classifier)**.** *The adversarial attack to a two-class linear classifier is the solution of*

$$\underset{\boldsymbol{x}}{\textit{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 \quad \textit{subject to} \quad \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0, \tag{3.25}$$

*which is given by*

$$\boldsymbol{x}^* = \boldsymbol{x}_0 - \left( \frac{\boldsymbol{w}^T \boldsymbol{x}_0 + w_0}{\|\boldsymbol{w}\|_2} \right) \frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}. \tag{3.26}$$

*Proof.* The Lagrange multiplier of the constrained optimization is given by

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

The solution of the optimization is the saddle point $(\boldsymbol{x}^*, \lambda^*)$ such that $\nabla_{\boldsymbol{x}} \mathcal{L} = 0$ and $\nabla_\lambda \mathcal{L} = 0$.

Taking derivative with respect to $\boldsymbol{x}$ and $\lambda$ yields

$$\begin{aligned} \nabla_{\boldsymbol{x}} \mathcal{L} &= \boldsymbol{x} - \boldsymbol{x}_0 + \lambda \boldsymbol{w} = 0, \\ \nabla_\lambda \mathcal{L} &= \boldsymbol{w}^T \boldsymbol{x} + w_0 = 0. \end{aligned}$$

 

Multiplying the first equation by $\boldsymbol{w}^T$ yields

$$0 = \boldsymbol{w}^T\boldsymbol{x} - \boldsymbol{w}^T\boldsymbol{x}_0 + \lambda\|\boldsymbol{w}\|^2 = -w_0 - \boldsymbol{w}^T\boldsymbol{x}_0 + \lambda\|\boldsymbol{w}\|^2. \tag{3.27}$$

Thus, the optimal $\lambda$ is

$$\lambda^* = (\boldsymbol{w}^T\boldsymbol{x}_0 + w_0)/\|\boldsymbol{w}\|^2. \tag{3.28}$$

Correspondingly, the optimal $\boldsymbol{x}$ is

$$\boldsymbol{x}^* = \boldsymbol{x}_0 - \lambda^*\boldsymbol{w} = \boldsymbol{x}_0 - \left(\frac{\boldsymbol{w}^T\boldsymbol{x}_0 + w_0}{\|\boldsymbol{w}\|}\right)\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|_2}.$$

□

The result of this theorem provides many useful interpretation. First, the search direction is $-\boldsymbol{w}$ (or $-\boldsymbol{w}/\|\boldsymbol{w}\|_2$ for unit norm.) The negative sign is there because $\boldsymbol{w} = \boldsymbol{w}_i - \boldsymbol{w}_t$ is pointing from class $\mathcal{C}_t$ to $\mathcal{C}_i$, which is opposite to the desired search direction. The search step $\frac{\boldsymbol{w}^T\boldsymbol{x}_0+w_0}{\|\boldsymbol{w}\|_2}$ measures the distance from the input $\boldsymbol{x}_0$ to the target class $\mathcal{C}_t$. This is coherent to the projection perspective we presented earlier.

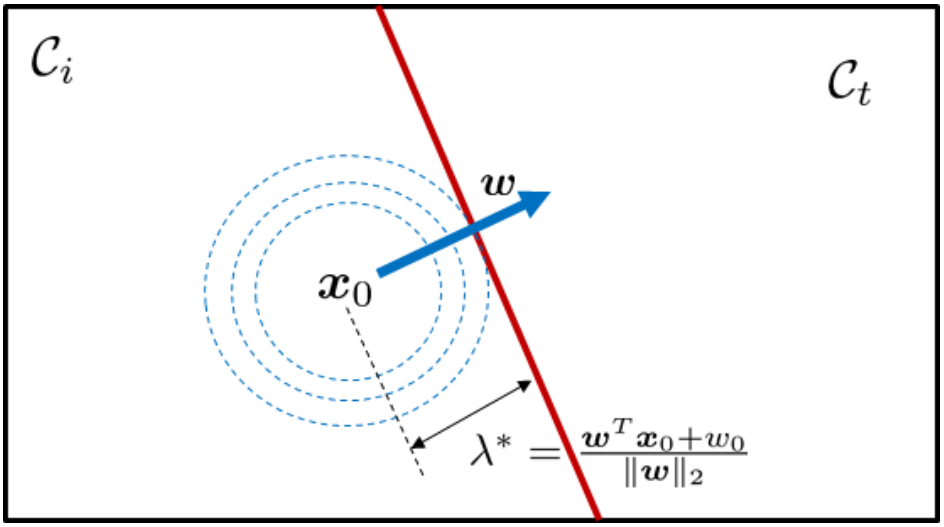


Figure 3.8: Geometry of minimum-norm attack for a two-class linear classifier with objective function $\|\boldsymbol{x} - \boldsymbol{x}_0\|^2$. The solution is a projection of the input $\boldsymbol{x}_0$ onto the separating hyperplane of the classifier.

## DeepFool Attack by Moosavi-Dezfooli et al. 2016

The DeepFool attack by Moosavi-Dezfooli, Fawzi and Frossard (CVPR 2016) is a generalization of the minimum $\ell_2$-norm attack under general decision boundaries. Instead of having an affine constraint $\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$, the constraint is given by a nonlinear function $g(\boldsymbol{x}) = 0$. For a two-class problem, DeepFool formulates the optimization as follows.

 

**Definition 6** (DeepFool Attack by Moosavi-Dezfooli et al. 2016). *The DeepFool attack for a two-class classification generates the attack by solving the optimization*

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x}-\boldsymbol{x}_0\|^2 \quad \text{subject to} \quad g(\boldsymbol{x})=0, \tag{3.29}$$

*where $g(\boldsymbol{x})=0$ is the nonlinear decision boundary separating the two classes.*

Due to the nonlinearity of $g(\boldsymbol{x})$, it is generally very difficult to derive a closed-form solution. The numerical procedure to compute the solution can be derived by iteratively updating $\boldsymbol{x}$, where each iteration minimizes $\boldsymbol{x}$ over the first order approximation of $g(\boldsymbol{x})$:

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

where $\boldsymbol{x}^{(k)}$ is the $k$-th iterate of the solution. In other words, we are defining a procedure

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}}{\text{argmin}} \quad \|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|^2 \quad \text{subject to} \quad g(\boldsymbol{x}^{(k)}) + \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}-\boldsymbol{x}^{(k)}) = 0. \tag{3.30}$$

By identifying $\boldsymbol{w}^{(k)} = \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})$ and $w_0^{(k)} = g(\boldsymbol{x}^{(k)}) - \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})^T \boldsymbol{x}^{(k)}$, the linearized problem Equation (3.30) becomes

$$\boldsymbol{x}^{(k+1)} = \underset{\boldsymbol{x}}{\text{argmin}} \quad \|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|^2 \quad \text{subject to} \quad (\boldsymbol{w}^{(k)})^T\boldsymbol{x} + w_0^{(k)} = 0$$

The solution can thus be found by using Equation (3.26), yielding

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \boldsymbol{x}^{(k)} - \left(\frac{(\boldsymbol{w}^{(k)})^T\boldsymbol{x}^{(k)} + w_0^{(k)}}{\|\boldsymbol{w}^{(k)}\|^2}\right)\boldsymbol{w}^{(k)} \\ &= \boldsymbol{x}^{(k)} - \left(\frac{g(\boldsymbol{x}^{(k)})}{\|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})\|^2}\right)\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)}).\end{aligned}$$

This result is summarized in the following corollary.

**Corollary 1** (DeepFool Algorithm for Two-Class Problem). *An iterative procedure to obtain the DeepFool attack solution is*

$$\begin{aligned}\boldsymbol{x}^{(k+1)} &= \underset{\boldsymbol{x}}{\textit{argmin}} \quad \|\boldsymbol{x}-\boldsymbol{x}^{(k)}\|^2 \quad \textit{subject to} \quad g(\boldsymbol{x}^{(k)}) + \nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})^T(\boldsymbol{x}-\boldsymbol{x}^{(k)}) = 0 \\ &= \boldsymbol{x}^{(k)} - \left(\frac{g(\boldsymbol{x}^{(k)})}{\|\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)})\|^2}\right)\nabla_{\boldsymbol{x}} g(\boldsymbol{x}^{(k)}).\end{aligned} \tag{3.31}$$

 

For multi-class problems, DeepFool has a simple solution if the classifier is **one-vs-all**. Here, by one-vs-all we meant that the classifier decides the class by considering $n$ two-class problems, where $n$ is the number of classes which is also the number of discriminant functions. Interested readers can consult the paper for details. However, we should remark that the one-vs-all approach does not apply to a **linear machine**, because one-vs-all essentially is handling a sequence of separating hyperplanes, whereas a linear machine considers a polytope.

## B. Maximum-Allowable Attack

The minimum-norm attack using $\|\boldsymbol{x}-\boldsymbol{x}_0\|^2$ is a projection. What about maximum allowable attack? Before we answer this question, let us consider one more example of the minimum-norm attack but this time using $\|\boldsymbol{x}-\boldsymbol{x}_0\|_\infty$. That is, consider the optimization problem

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x}-\boldsymbol{x}_0\|_\infty \quad \text{subject to} \quad \boldsymbol{w}^T\boldsymbol{x}+w_0=0.$$

Let $\boldsymbol{r}=\boldsymbol{x}-\boldsymbol{x}_0$, we can rewrite the problem as

$$\underset{\boldsymbol{r}}{\text{minimize}} \quad \|\boldsymbol{r}\|_\infty \quad \text{subject to} \quad \boldsymbol{w}^T\boldsymbol{r}=b_0,$$

where $b_0=-(\boldsymbol{w}^T\boldsymbol{x}_0+w_0)$. What is the solution to this optimization? One very useful technique in handling minimax optimization is the Holder's inequality.

**Theorem 3** (Holder's Inequality)**.** *Let $\boldsymbol{x}\in\mathbb{R}^d$ and $\boldsymbol{y}\in\mathbb{R}^d$. Then,*

$$-\|\boldsymbol{x}\|_p\|\boldsymbol{y}\|_q \leq |\boldsymbol{x}^T\boldsymbol{y}| \leq \|\boldsymbol{x}\|_p\|\boldsymbol{y}\|_q \tag{3.32}$$

*for any $p$ and $q$ such that $\frac{1}{p}+\frac{1}{q}=1$, where $p\in[1,\infty]$.*

Holder's inequality can be considered as a generalization of the Cauchy-Schwarz inequality, which states that $|\boldsymbol{x}^T\boldsymbol{y}|\leq\|\boldsymbol{x}\|_2\|\boldsymbol{y}\|_2$, i.e., $p=q=1/2$.

If we let $p=1$ and $q=\infty$, then we can show that $|\boldsymbol{x}^T\boldsymbol{y}|\leq\|\boldsymbol{x}\|_1\|\boldsymbol{y}\|_\infty$. This then implies that

$$|b_0|=|\boldsymbol{w}^T\boldsymbol{r}|\leq\|\boldsymbol{w}\|_1\|\boldsymbol{r}\|_\infty, \quad\Longrightarrow\quad \|\boldsymbol{r}\|_\infty\geq\frac{|b_0|}{\|\boldsymbol{w}\|_1}.$$

In other words, the objective function $\|\boldsymbol{r}\|_\infty$ is lower bounded by $\frac{|b_0|}{\|\boldsymbol{w}\|_1}$, which is a constant independent of $\boldsymbol{r}$. If this lower bound can be attained, then $\|\boldsymbol{r}\|_\infty$ is minimized. So for what $\boldsymbol{r}$ can the lower bound be attained? We claim that the vector is $\boldsymbol{r}=\eta\cdot\text{sign}(\boldsymbol{w})$ for some constant $\eta$ to be determined. For this choice of $\boldsymbol{r}$, we can show that

$$\|\boldsymbol{r}\|_\infty=\max_i|\eta\cdot\text{sign}(w_i)|=|\eta|.$$

 

Therefore, if we let $\eta = b_0/\|\boldsymbol{w}\|_1$ (which is a constant independent of $\boldsymbol{r}$), then we will have $\|\boldsymbol{r}\|_\infty = |\eta| = \frac{|b_0|}{\|\boldsymbol{w}\|_1}$, i.e., the lower bound is attained. To summarize, we have the following theorem.

**Theorem 4** (Minimum $\ell_\infty$ Norm Attack for Two-Class Linear Classifier)**.** *The minimum $\ell_\infty$ norm attack for a two-class linear classifier, i.e.,*

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|_\infty \quad \text{subject to} \quad \boldsymbol{w}^T\boldsymbol{x} + w_0 = 0 \tag{3.33}$$

*is given by*

$$\boldsymbol{x} = \boldsymbol{x}_0 - \left(\frac{\boldsymbol{w}^T\boldsymbol{x}_0 + w_0}{\|\boldsymbol{w}\|_1}\right) \cdot \text{sign}(\boldsymbol{w}). \tag{3.34}$$

this direction is not perpendicular to line. it is in direction of vertex of square contours

vector of 1 or -1

Comparing the $\ell_2$ attack and the $\ell_\infty$ attack, $\ell_2$ attack gives the minimum distance attack as it is the projection of $\boldsymbol{x}_0$ onto the separating hyperplane. In contrast, the $\ell_\infty$ attack does not achieve the minimum distance but the attack direction is basically a vector of $+1$ and $-1$ (due to the sign operation). In fact, as shown in Figure 3.9, the attacked data $\boldsymbol{x}$ must be one of the vertices of the square norm ball defined by the $\ell_\infty$ norm, which by construction is vector containing only +1 or -1.

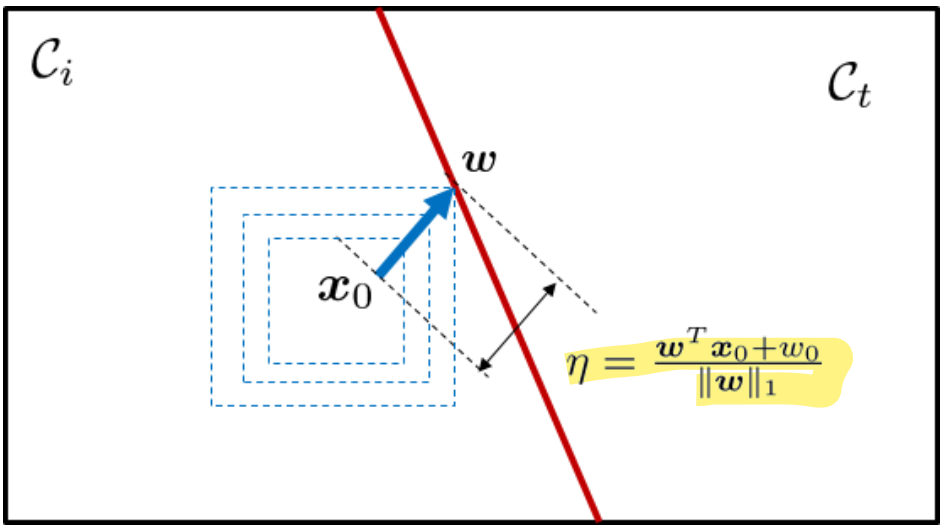


Figure 3.9: Geometry of minimum $\ell_\infty$ norm attack for a two-class linear classifier with objective function $\|\boldsymbol{x} - \boldsymbol{x}_0\|_\infty$. Because of the square geometry of the objective function, the perturbed data $\boldsymbol{x}$ must be the vertex of the square. Correspondingly, the attack direction is sign($\boldsymbol{w}$).

How is minimum $\ell_\infty$ norm attack related to the maximum allowable $\ell_\infty$ norm attack? Recall that the maximum allowable attack is given by

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \boldsymbol{w}^T\boldsymbol{x} + w_0 \quad \text{subject to} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|_\infty \leq \eta. \tag{3.35}$$

 

By defining $\boldsymbol{x} = \boldsymbol{x}_0 + \boldsymbol{r}$, we have $\boldsymbol{w}^T\boldsymbol{x} + w_0 = \boldsymbol{w}^T\boldsymbol{x}_0 + \boldsymbol{w}^T\boldsymbol{r} + w_0$. By defining $b_0 = -(\boldsymbol{w}^T\boldsymbol{x}_0 + w_0)$, the optimization can be rewritten as

$$\underset{\boldsymbol{r}}{\text{minimize}} \quad \boldsymbol{w}^T\boldsymbol{r} - b_0 \quad \text{subject to} \quad \|\boldsymbol{r}\|_\infty \leq \eta. \tag{3.36}$$

Using the Holder's inequality (the negative side), we can show that

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

The lower bound of $\boldsymbol{w}^T\boldsymbol{r}$ is attained when $\boldsymbol{r} = -\eta \cdot \text{sign}(\boldsymbol{w})$, because

$$\boldsymbol{w}^T\boldsymbol{r} = -\eta\boldsymbol{w}^T\text{sign}(\boldsymbol{w}) = -\eta\sum_{i=1}^{d} w_i\text{sign}(w_i) = -\eta\sum_{i=1}^{d}|w_i| = -\eta\|\boldsymbol{w}\|_1.$$

Therefore, the attacked data point $\boldsymbol{x}$ is $\boldsymbol{x} = \boldsymbol{x}_0 - \eta \cdot \text{sign}(\boldsymbol{w})$. This leads to the following theorem.

**Theorem 5** (Maximum Allowable $\ell_\infty$ Norm Attack of Two-Class Linear Classifier)**.** *The maximum allowable $\ell_\infty$ norm attack for a two-class linear classifier, i.e.,*

$$\underset{\boldsymbol{x}}{\textit{minimize}} \quad \boldsymbol{w}^T\boldsymbol{x} + w_0 \quad \textit{subject to} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|_\infty \leq \eta. \tag{3.37}$$

*is given by*

$$\boldsymbol{x} = \boldsymbol{x}_0 - \eta \cdot \textit{sign}(\boldsymbol{w}). \tag{3.38}$$

The result of Equation (3.38) is exactly the same as Equation (3.34), except that the allowable attack magnitude $\eta$ is unspecified. This should not be a surprise, as minimum norm attack does not involve internal parameters whereas maximum allowable attack requires $\eta$.

## Fast Gradient Sign Method (FGSM)

The result of Equation (3.38) can also be considered as a special case of the **Fast Gradient Sign Method** (FGSM) by Goodfellow et al. (NIPS 2014). The FGSM was originally designed for attacking deep neural networks. The idea is to maximize certain loss function $J(\boldsymbol{x}; \boldsymbol{w})$, subject to an upper bound on the perturbation, e.g., $\|\boldsymbol{x} - \boldsymbol{x}_0\|_\infty \leq \eta$, in the original FGSM paper by Goodfellow et al. The loss function $J(\boldsymbol{x}; \boldsymbol{w})$ can be treated as the loss incurred by evaluating a data point $\boldsymbol{x}$ to a classifier parameterized by $\boldsymbol{w}$. Therefore, by maximizing the loss we aim to make the classifier perform as worse as possible. The definition of the FGSM for perturbing a data point $\boldsymbol{x}_0$ with respect to a loss function $J(\boldsymbol{x}; \boldsymbol{w})$ is stated as follows.

 

**Definition 7** (Fast Gradient Sign Method (FGSM) by Goodfellow et al 2014)**.** *Given a loss function $J(\boldsymbol{x}; \boldsymbol{w})$, the FGSM creates an attack $\boldsymbol{x}$ by*

$$\boldsymbol{x} = \boldsymbol{x}_0 + \eta \cdot sign(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0; \boldsymbol{w})). \tag{3.39}$$

Note that in this definition, the gradient is taken with respect to the input $\boldsymbol{x}$, not the parameter $\boldsymbol{w}$. Therefore, $\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0; \boldsymbol{w})$ should be interpreted as the gradient of $J$ with respect to $\boldsymbol{x}$ evaluated at $\boldsymbol{x}_0$. The definition of FGSM also explains its name: It is the gradient of the loss function, and because of the $\ell_\infty$ bound on the perturbation magnitude, the perturbation direction is the sign of the gradient. It is a fast method, because the gradient is typically not difficult to compute even for deep neural networks (by means of back propagation, which is available in most programming platforms.)

How do we specify the loss function? For most of the modern classifiers the loss function is also the training loss. But for simple toy problems, it is possible to explicitly derive the expression. Below is an example for the two-class linear classifier.

**Example**. (FGSM Loss Function for a Two-Class Linear Classifier). Recall that the objective function in the maximum allowable attack is

$$\varphi(\boldsymbol{x}) = \boldsymbol{w}^T\boldsymbol{x} + w_0 = (\boldsymbol{w}_i^T\boldsymbol{x} + w_{i0}) - (\boldsymbol{w}_t^T\boldsymbol{x} + w_{t0}),$$

which is a measure of how far is $\mathcal{C}_i$ from $\mathcal{C}_t$. If we can make $(\boldsymbol{w}_i^T\boldsymbol{x}+w_{i0})-(\boldsymbol{w}_t^T\boldsymbol{x}+w_{t0}) \leq 0$, then we are guaranteed that $\boldsymbol{x}$ is misclassified as $\mathcal{C}_t$. Putting this in another way, we can also define a loss function $J(\boldsymbol{x}) = (\boldsymbol{w}_t^T\boldsymbol{x} + w_{t0}) - (\boldsymbol{w}_i^T\boldsymbol{x} + w_{i0})$. If $J(\boldsymbol{x}) \geq 0$, then $\boldsymbol{x}$ is misclassified and so we have a loss. By maximizing $J(\boldsymbol{x})$, we are guaranteed to make the classifier perform worse. Therefore, the loss function is

$$J(\boldsymbol{x}; \boldsymbol{w}) = -(\boldsymbol{w}^T\boldsymbol{x} + w_0). \tag{3.40}$$

Given a loss function $J(\boldsymbol{x}; \boldsymbol{w})$, is there an optimization that will result the FGSM attack? If we can answer this question, then perhaps we can generalize FGSM to a broader class of attacks. To this end, we notice that for general (possibly nonlinear) loss functions, FGSM can be interpreted by applying a first order approximation:

$$J(\boldsymbol{x}; \boldsymbol{w}) = J(\boldsymbol{x}_0 + \boldsymbol{r}; \boldsymbol{w}) \approx J(\boldsymbol{x}_0; \boldsymbol{w}) + \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0; \boldsymbol{w})^T\boldsymbol{r}. \tag{3.41}$$

Therefore, finding $\boldsymbol{r}$ to maximize $J(\boldsymbol{x}_0+\boldsymbol{r}; \boldsymbol{w})$ is approximately equivalent to finding $\boldsymbol{r}$ which maximizes $J(\boldsymbol{x}_0; \boldsymbol{w}) + \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0; \boldsymbol{w})^T\boldsymbol{r}$. Hence, FGSM is the solution to

$$\underset{\boldsymbol{r}}{\text{maximize}} \quad J(\boldsymbol{x}_0; \boldsymbol{w}) + \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0; \boldsymbol{w})^T\boldsymbol{r} \quad \text{subject to} \quad \|\boldsymbol{r}\|_\infty \leq \eta,$$

 

which can be simplified to

$$\underset{\boldsymbol{r}}{\text{minimize}} \quad \underbrace{-\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})^T \boldsymbol{r}}_{\boldsymbol{w}^T \boldsymbol{r}} - \underbrace{J(\boldsymbol{x}_0;\, \boldsymbol{w})}_{w_0} \quad \text{subject to} \quad \|\boldsymbol{r}\|_\infty \leq \eta,$$

where we flipped the maximization to minimization by putting a negative sign to the gradient. Interestingly, this optimization is exactly the same as Equation (3.36) with $\boldsymbol{w}$ and $w_0$ identified as above. Therefore, the solution is

$$\boldsymbol{r} = -\eta \cdot \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})),$$

and hence the perturbed data is $\boldsymbol{x} = \boldsymbol{x}_0 + \eta \cdot \text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w}))$. In other words, FGSM is the solution of the following maximum allowable attack problem.

**Corollary 2** (FGSM as a Maximum Allowable Attack Problem)**.** *The FGSM attack can be formulated as the optimization with $J(\boldsymbol{x};\, \boldsymbol{w})$ being the loss function:*

$$\underset{\boldsymbol{r}}{\textit{maximize}} \quad \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})^T \boldsymbol{r} + J(\boldsymbol{x}_0;\, \boldsymbol{w}) \quad \textit{subject to} \quad \|\boldsymbol{r}\|_\infty \leq \eta,$$

*of which the solution is given by*

$$\boldsymbol{x} = \boldsymbol{x}_0 + \eta \cdot \textit{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})). \tag{3.42}$$

Knowing that FGSM corresponds to a maximum allowable attack with $\ell_\infty$ norm, we can easily generalize the attack to other $\ell_p$ norms.

**Corollary 3** (FGSM with $\ell_2$ Norm)**.** *The FGSM attack with $\ell_2$-norm bound on the perturbation magnitude is*

$$\underset{\boldsymbol{r}}{\textit{maximize}} \quad \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})^T \boldsymbol{r} + J(\boldsymbol{x}_0;\, \boldsymbol{w}) \quad \textit{subject to} \quad \|\boldsymbol{r}\|_2 \leq \eta,$$

*of which the solution is given by*

$$\boldsymbol{x} = \boldsymbol{x}_0 + \eta \cdot \frac{\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})}{\|\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})\|_2}. \tag{3.43}$$

*Proof.* The optimization problem can be rewritten as

$$\underset{\boldsymbol{r}}{\text{minimize}} \quad \boldsymbol{w}^T \boldsymbol{r} + w_0 \quad \text{subject to} \quad \|\boldsymbol{r}\|_2 \leq \eta,$$

where $\boldsymbol{w} = -\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})$ and $w_0 = -J(\boldsymbol{x}_0;\, \boldsymbol{w})$. By Cauchy inequality, we have that $\boldsymbol{w}^T \boldsymbol{r} \geq -\|\boldsymbol{w}\|_2 \|\boldsymbol{r}\|_2 \geq -\eta \|\boldsymbol{w}\|_2$, where the second inequality holds because $\|\boldsymbol{r}\|_2 \leq \eta$. The lower bound of $\boldsymbol{w}^T \boldsymbol{r} \geq -\eta \|\boldsymbol{w}\|_2$ is attained when $\boldsymbol{r} = -\eta \boldsymbol{w} / \|\boldsymbol{w}\|_2$. Therefore, the perturbation is $\boldsymbol{x} = \boldsymbol{x}_0 + \eta \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w}) / \|\nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})\|_2$. □

 

## Iterative Gradient Sign Method by Kurakin et al. 2017

One obvious problem of the FGSM (in fact, all other attack strategies) is the potential unboundedness of the perturbed data $\boldsymbol{x}$. For many image classification problems, the unboundedness of $\boldsymbol{x}$ is physically prohibited, e.g., a pixel can only take values between [0, 255] or [0, 1] for normalized intensity. The **Iterative Fast Gradient Sign Method** (I-FGSM) by Kurakin, Goodfellow and Bengio (ICLR 2017) addresses this problem by posing an bound constraint on $\boldsymbol{x}$. Taking FGSM as an example, this changes the optimization to

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{maximize}} \;\; \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})^T \boldsymbol{x} - \nabla_{\boldsymbol{x}} J(\boldsymbol{x}_0;\, \boldsymbol{w})^T \boldsymbol{x}_0 + J(\boldsymbol{x}_0;\, \boldsymbol{w}) \\ &\text{subject to} \;\; \|\boldsymbol{x} - \boldsymbol{x}_0\|_\infty \leq \eta, \;\; 0 \leq \boldsymbol{x} \leq 1. \end{aligned}$$

In addition to the bound constraint, I-FGSM uses an iterative linearization rather than the one-shot linearization in FGSM. Therefore, the algorithm of computing the attack becomes

$$\begin{aligned} \boldsymbol{x}^{(k+1)} = &\underset{\boldsymbol{x}}{\text{argmax}} \;\; \nabla_{\boldsymbol{x}} J(\boldsymbol{x}^{(k)};\, \boldsymbol{w})^T \boldsymbol{x} - \nabla_{\boldsymbol{x}} J(\boldsymbol{x}^{(k)};\, \boldsymbol{w})^T \boldsymbol{x}^{(k)} + J(\boldsymbol{x}^{(k)};\, \boldsymbol{w}) \\ &\text{subject to} \;\; \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|_\infty \leq \eta, \;\; 0 \leq \boldsymbol{x} \leq 1 \end{aligned}$$

By dropping the terms that are independent of $\boldsymbol{x}$, the optimization can be simplified to

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

where $\mathcal{P}_{[0,1]}(\boldsymbol{x})$ is a projection operator that elementwisely projects out of bound values to the bound $0 \leq \boldsymbol{x} \leq 1$.

**Corollary 4** (I-FGSM Algorithm as Projected FGSM)**.** *The Iterative FGSM by Kurakin et al. generates the attack by iteratively solving*

$$\boldsymbol{x}^{(k+1)} = \underset{0 \leq \boldsymbol{x} \leq 1}{\textit{argmax}} \;\; \nabla_{\boldsymbol{x}} J(\boldsymbol{x}^{(k)};\, \boldsymbol{w})^T \boldsymbol{x} \;\; \textit{subject to} \;\; \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|_\infty \leq \eta, \tag{3.44}$$

*of which the per-iteration solution is given by*

$$\boldsymbol{x}^{(k+1)} = \mathcal{P}_{[0,1]} \left\{ \boldsymbol{x}^{(k)} + \eta \cdot \textit{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{x}^{(k)};\, \boldsymbol{w})) \right\}. \tag{3.45}$$

**Remark**. (Choice of $\eta$ for I-FGSM.) Practically, when generating attack using I-FGSM, the magnitude upper bound $\eta$ is usually reduced to a smaller constant so that the number of active attack pixels is limited.

 

## C. Regularization-based Attack

In both minimum norm attack and maximum allowable attack, the optimizations are constrained. For certain simple cases, e.g., binary linear classifiers, closed-form solutions can be derived. However, for advanced classifiers such as deep neural networks, solving an optimization involving constraints are typically very difficult. The regularization-based attack alleviates the problem by considering

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x}-\boldsymbol{x}_0\|^2 + \lambda\left(\max_{j\neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})\right).$$

For two-class linear classifier, this is simplified to

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x}-\boldsymbol{x}_0\|^2 + \lambda(\boldsymbol{w}^T\boldsymbol{x}+w_0).$$

Since the regularization-based attack is an unconstrained version of the minimum norm attack and the maximum allowable attack, one should expect to have a very similar solution.

**Theorem 6** (Regularization-based Attack for Two-Class Linear Classifier)**.** *The regularization-based attack for a two-class linear classifier generates the attack by solving*

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|^2 + \lambda(\boldsymbol{w}^T\boldsymbol{x}+w_0), \tag{3.46}$$

*of which the solution is given by*

$$\boldsymbol{x} = \boldsymbol{x}_0 - \lambda\boldsymbol{w}. \tag{3.47}$$

*Proof.* Let $\varphi(\boldsymbol{x}) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{x}_0\|^2 + \lambda(\boldsymbol{w}^T\boldsymbol{x}+w_0)$. Taking the first order derivative and sending to zero yields

$$0 = \nabla\varphi(\boldsymbol{x}) = (\boldsymbol{x}-\boldsymbol{x}_0) + \lambda\boldsymbol{w}. \tag{3.48}$$

Therefore, we have $\boldsymbol{x} = \boldsymbol{x}_0 - \lambda\boldsymbol{w}$. □

In a regularization based attack, the attack magnitude $\lambda$ is specified by the user. It is not difficult to see that for linear classifiers, the optimal $\lambda$ and $\eta$ are both $\lambda = \eta = \frac{\boldsymbol{w}^T\boldsymbol{x}_0+w_0}{\|\boldsymbol{w}\|^2}$. However, in minimum norm attack, this magnitude is automatically guaranteed by the algorithm. Maximum allowable attack is slightly more complicated, but it is still more interpretable than regularization based attack as defining $\eta$ is equivalent to controlling the maximum perturbation.

Another drawback of the regularization based attack is that the objective value could potentially be unbounded. See the example below.

 

**Example**. (Unboundedness of Regularization-based Attack). Consider an $\ell_1$-norm attack:

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x}-\boldsymbol{x}_0\|_1 + \lambda(\boldsymbol{w}^T\boldsymbol{x}+w_0),$$

which is equivalent to

$$\underset{\boldsymbol{r}}{\text{minimize}} \quad \|\boldsymbol{r}\|_1 + \lambda\boldsymbol{w}^T\boldsymbol{r}.$$

The optimality condition is stated by the subgradient:

$$0 \in \partial\|\boldsymbol{r}\|_1 + \lambda\boldsymbol{w}, \quad \Longrightarrow \quad 0 \in \text{sign}(\boldsymbol{r}) + \lambda\boldsymbol{w},$$

where $\text{sign}(r_i) = [-1,1]$ if $r_i = 0$, $\text{sign}(r_i) = \{+1\}$ if $r_i > 0$, and $\text{sign}(r_i) = \{-1\}$ if $r_i < 0$. Therefore, the optimality condition holds only when $\lambda w_i \in [-1,1]$, i.e., $|\lambda\boldsymbol{w}| \leq \mathbf{1}$. When this happens, we must have $\boldsymbol{r} = \mathbf{0}$ and so the optimal value is 0. However, if $|\lambda\boldsymbol{w}| > \mathbf{1}$, then it is impossible to have $\mathbf{0} \in \text{sign}(\boldsymbol{r}) + \lambda\boldsymbol{w}$ regardless of how we choose $\boldsymbol{r}$. This means that the objective function is unbounded below. In fact, one can show that the 1D function $f(x) = |x| + \alpha x \to -\infty$ as $x \to -\infty$ if $\alpha > 1$, or as $x \to \infty$ if $\alpha < -1$.

To summarize, while regularization based attack leads to an unconstrained optimization which is attractive to many off-the-shelf optimization packages, the parameter tuning of $\lambda$ and the potential unboundedness of the problem are its major drawbacks. In addition, for problems involving nonlinear regularization terms, the optimization may be difficult as well.

## Carlini-Wagner Attack

The Carlini-Wagner attack (2016) is a regularization-based attack with some critical modifications which can resolve the unboundedness issue. The main idea is to realize that for general classifiers, the optimization can be written as

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x}-\boldsymbol{x}_0\|^2 + \iota_\Omega(\boldsymbol{x}), \tag{3.49}$$

where

$$\iota_\Omega(\boldsymbol{x}) = \begin{cases} 0, & \text{if} \quad \max_{j\neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \leq 0, \\ +\infty, & \text{otherwise.} \end{cases} \tag{3.50}$$

That is, when the constraint $\max_{j\neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x}) \leq 0$ is satisfied, the optimization is simplified to $\min\|\boldsymbol{x}-\boldsymbol{x}_0\|^2$ which has a trivial solution $\boldsymbol{x} = \boldsymbol{x}_0$. In other words, when $\boldsymbol{x}_0$ is already in the targeted class $\mathcal{C}_t$ (so that $\max_{j\neq t}\{g_j(\boldsymbol{x}_0)\} \leq g_t(\boldsymbol{x}_0)$), then there is no need to perturb the data point. Of course, in most of the cases $\boldsymbol{x}_0 \notin \mathcal{C}_t$, and so we need to iteratively check whether the current iterate satisfies the constraint. Yet, the introduction

 

of $\iota_\Omega(\boldsymbol{x})$ allows us to rewrite the constrained problem (i.e., the minimum-norm attack) into an unconstrained problem.

Unfortunately, the unconstrained problem in Equation (3.49) remains uneasy to solve, for $\iota_\Omega$ is either 0 or $+\infty$. Carlini-Wagner attack relaxes $\iota_\Omega$ by considering a rectifier function

$$\zeta(x) = \max(x, 0). \tag{3.51}$$

Putting into the optimization, we have

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 + \lambda \cdot \zeta(\boldsymbol{x}), \tag{3.52}$$

for some constant $\lambda > 0$. Intuitively, $\zeta(\cdot)$ maps any positive $x$ to itself, and any negative $x$ to 0. It is a useful relaxation because when the constraint is satisfied, $\zeta(x)$ will be 0. When the constraint is not satisfied, $\eta(x)$ penalizes the objective linearly.

Using the rectifier function, the attack can now be written as

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 + \lambda \cdot \max\left\{\left(\max_{j \neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})\right), 0\right\}, \tag{3.53}$$

which is known as the Carlini-Wagner attack. In the original paper by Carlini and Wagner, it was shown that besides the rectifier function there are many other choices one can consider. Readers interested in the details can consult their paper.

The rectifier function is particularly useful for two reasons. The first reason is the computation, because the derivative of a rectifier function is a unit step function which is computationally very easy to implement. The second reason is its effect in handling the non-boundedness issue. Below is an example.

**Example**. (Carlini-Wagner Attack for $\ell_1$ Problem.) Let us revisit the $\ell_1$ minimization but using the rectifier function this time. Thus, the optimization becomes

$$\underset{\boldsymbol{x}}{\text{minimize}} \quad \|\boldsymbol{x} - \boldsymbol{x}_0\|_1 + \lambda \max\left(\boldsymbol{w}^T\boldsymbol{x} + w_0, 0\right).$$

The objective function is always non-negative, as $\|\boldsymbol{x} - \boldsymbol{x}_0\|_1 \geq 0$ and $\max\left(\boldsymbol{w}^T\boldsymbol{x} + w_0, 0\right) \geq 0$. Therefore, the optimization is always bounded below by 0.

Note that the lower bound is achieved when $\boldsymbol{x} = \boldsymbol{x}_0$ and $\boldsymbol{w}^T\boldsymbol{x}_0 + w_0 = 0$. This happens only when the attack solution is $\boldsymbol{x} = \boldsymbol{x}_0$ and $\boldsymbol{x}_0$ is right on the decision boundary. Clearly, this is unlikely to happen, and if this happens the optimal objective value is 0. Therefore, besides this trivial solution, the optimization is guarantee to attain a finite minimum objective.

 

**Remark**. (Convexity of CW attack.) Let $\varphi(\boldsymbol{x}) = \max_{j\neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})$. It is not difficult to show that the function $h(\boldsymbol{x}) = \max(\varphi(\boldsymbol{x}), 0)$ is convex if $\varphi$ is convex:

$$\begin{aligned} h(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}) &= \max\left(\varphi(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{y}), 0\right) \leq \max\left(\alpha\varphi(\boldsymbol{x}) + (1-\alpha)\varphi(\boldsymbol{y}), 0\right) \\ &\leq \alpha\max\left(\varphi(\boldsymbol{x}), 0\right) + (1-\alpha)\max(\varphi(\boldsymbol{y}), 0) = \alpha h(\boldsymbol{x}) + (1-\alpha)h(\boldsymbol{y}). \end{aligned}$$

In particular, if $g_i$ is linear, then $\varphi$ is linear and so $h$ is convex.

CW attack is appeal because it allows us to use simple algorithms to compute the attack. To do so, we first note that the gradient of $\zeta(\cdot)$ is

$$\frac{d}{ds}\zeta(s) = \mathbb{I}\{s > 0\} \stackrel{\text{def}}{=} \begin{cases} 1, & \text{if} \quad s > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Let $i^*(\boldsymbol{x})$ be the index of the maximum response: $i^*(\boldsymbol{x}) = \underset{j\neq t}{\text{argmax}}\{g_j(\boldsymbol{x})\}$. If, for the time being, we assume that the index $i^*$ is independent of $\boldsymbol{x}$, then the gradient for the rectifier function applied to the constraint is

$$\begin{aligned} \nabla_{\boldsymbol{x}}\zeta\left(\max_{j\neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})\right) &= \nabla_{\boldsymbol{x}}\zeta\left(\{g_{i^*}(\boldsymbol{x})\} - g_t(\boldsymbol{x})\right) \\ &= \begin{cases} \nabla_{\boldsymbol{x}} g_{i^*}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} g_t(\boldsymbol{x}), & \text{if} \quad g_{i^*}(\boldsymbol{x}) - g_t(\boldsymbol{x}) > 0, \\ 0, & \text{otherwise.} \end{cases} \\ &= \mathbb{I}\{g_{i^*}(\boldsymbol{x}) - g_t(\boldsymbol{x}) > 0\} \cdot (\nabla_{\boldsymbol{x}} g_{i^*}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} g_j(\boldsymbol{x})) \end{aligned}$$

Letting $\varphi(\boldsymbol{x})$ be the overall objective function, i.e.,

$$\varphi(\boldsymbol{x}) = \|\boldsymbol{x} - \boldsymbol{x}_0\|^2 + \lambda \cdot \max\left\{\left(\max_{j\neq t}\{g_j(\boldsymbol{x})\} - g_t(\boldsymbol{x})\right), 0\right\}, \tag{3.54}$$

it follows that the gradient is

$$\nabla\varphi(\boldsymbol{x};\ i^*) = 2(\boldsymbol{x} - \boldsymbol{x}_0) + \lambda \cdot \mathbb{I}\{g_{i^*}(\boldsymbol{x}) - g_t(\boldsymbol{x}) > 0\} \cdot (\nabla g_{i^*}(\boldsymbol{x}) - \nabla g_j(\boldsymbol{x})). \tag{3.55}$$

Here, we denote the gradient as $\nabla\varphi(\boldsymbol{x};\ i^*)$ by parametrizing using $i^*$, as the gradient depends on how $i^*$ is defined.

Applying gradient descent, the solution can be found by iteratively updating the estimate.

**Algorithm**. (CW Attack Gradient Descent.) The Gradient Descent algorithm for generating CW attack is given by the following iteration for $k = 1, 2, \ldots$

$$\begin{aligned} i^* &= \underset{j\neq t}{\text{argmax}}\{g_j(\boldsymbol{x}^k)\} \\ \boldsymbol{x}^{k+1} &= \boldsymbol{x}^k - \alpha\nabla\varphi(\boldsymbol{x}^k;\ i^*). \end{aligned} \tag{3.56}$$

 

As we can see in the above algorithm, the two parameters we need to tune are:

- $\alpha$: Gradient descent step size, which controls the rate of convergence. For non-convex problems, it is advised to use smaller $\alpha$ so that a local minimum is not missed.
- $\lambda$: Regularization parameter, which controls the relative strength between the distance term and the constraint term. The general rule of thumb is either by trial-and-error or cross-validation if training set is available.

## 3.4 Can Random Noise Attack?

Adversarial attack works because the perturbation is carefully designed. What if we perturb the data by pure i.i.d. Gaussian noise?

Recall that when launching attacks for a linear classifier, the perturbation always takes the form of $\boldsymbol{x} = \boldsymbol{x}_0 + \lambda \boldsymbol{w}$. That is, we want to move $\boldsymbol{x}_0$ along $\boldsymbol{w}$ by an amount $\lambda$ so that $\boldsymbol{x}$ is misclassified. Now, if we do not move along $\boldsymbol{w}$ but along a random vector $\boldsymbol{r}$ such that

$$\boldsymbol{x} = \boldsymbol{x}_0 + \sigma_r \boldsymbol{r}, \tag{3.57}$$

where $\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, then we can ask if we can still misclassify the data point $\boldsymbol{x}$. Clearly, this requires us to check whether $\boldsymbol{w}^T \boldsymbol{r} > 0$. If $\boldsymbol{w}^T \boldsymbol{r} > 0$, then $\boldsymbol{r}$ and $\boldsymbol{w}$ will form an acute angle and so for sufficient step size we will be able to move $\boldsymbol{x}_0$ to another class. If $\boldsymbol{w}^T \boldsymbol{r} < 0$, then $\boldsymbol{w}$ and $\boldsymbol{r}$ are form an obtuse angle and so $\boldsymbol{r}$ will move $\boldsymbol{x}_0$ to an opposite direction.

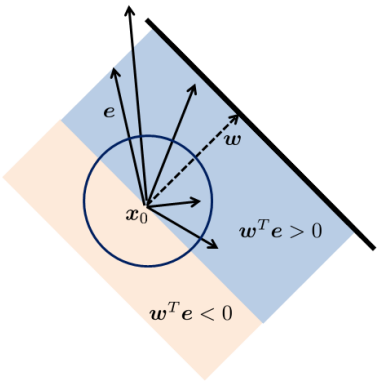


Figure 3.10: Attacking the linear classifier with i.i.d. noise is equivalent to putting an uncertainty circle around $\boldsymbol{x}_0$ with radius $\sigma_r$. The possible attack directions are those that form acute angle with the normal vector $\boldsymbol{w}$. Therefore, among all the possible $\boldsymbol{r}$'s, we are only interested in those such that $\boldsymbol{w}^T \boldsymbol{r} > 0$, which occupy half of the space in 2D.

Figure 3.10 provides a pictorial illustration of the noise attack. From this figure, we are tempted to think that i.i.d. noise can achieve 50% attack rate because $\boldsymbol{w}^T \boldsymbol{r} > 0$ occupies half of the space. The problem of such argument is that in high dimension, the probability of $\boldsymbol{w}^T \boldsymbol{r} > 0$ is diminishing very quickly as the dimensionality of $\boldsymbol{r}$ grows. This is a well-known

 

phenomenon called **curse of dimensionality**. To illustrate the idea, let us evaluate the probability of $\boldsymbol{w}^T\boldsymbol{r} \geq \epsilon$ for some $\epsilon > 0$. To this end, let us consider

$$\mathbb{P}\left[\frac{1}{d}\boldsymbol{w}^T\boldsymbol{r} \geq \epsilon\right] = \mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \epsilon\right],$$

where $d$ is the dimensionality of $\boldsymbol{w}$, i.e., $\boldsymbol{w} \in \mathbb{R}^d$. The tolerance level $\epsilon$ is a small positive constant that stays away from 0.

**Example**. Before we discuss the theoretical results, it will be useful to do a quick numerical experiment. Consider a special case where $\boldsymbol{w} = \mathbf{1}_{d\times 1}$, i.e., a $d$-dimensional all-one vector, and $\boldsymbol{r} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. In this case, we define the average as

$$Y \overset{\text{def}}{=} \frac{1}{d}\sum_{j=1}^{d} r_j. \tag{3.58}$$

It follows that $Y$ is a Gaussian random variable because linear combination of Gaussian remains a Gaussian. The mean and variance are

$$\mathbb{E}[Y] = 0, \quad \text{and} \quad \text{Var}[Y] = \frac{1}{d}. \tag{3.59}$$

Therefore, the probability of the event $\{Y > \epsilon\}$ is

$$\begin{aligned}\mathbb{P}[Y > \epsilon] &= \int_{\epsilon}^{\infty} \frac{1}{\sqrt{2\pi/d}} \exp\left\{-\frac{t^2}{2/d}\right\} dt \\ &= \int_{\epsilon\sqrt{\frac{d}{2}}}^{\infty} \frac{1}{\sqrt{\pi}} \exp\left\{-t^2\right\} dt \\ &= \frac{1}{2}\text{erfc}\left(\epsilon\sqrt{d/2}\right),\end{aligned} \tag{3.60}$$

where erfc is the complementary error function defined as $\text{erfc}(z) \overset{\text{def}}{=} \frac{2}{\sqrt{\pi}}\int_z^{\infty} e^{-t^2} dt$. As we can see in Figure 3.11, the probability drops rapidly as $d$ increases.

Let us discuss the general case. One classical result to derive the probability is the tail bound on Gaussian by Feller 1968. [1] If $Y$ is a Gaussian random variable with $Y \sim \mathcal{N}(\mu, \sigma)$, then

$$\mathbb{P}[Y \geq \mu + \sigma\epsilon] \leq \frac{1}{\epsilon}\frac{e^{-\epsilon^2/2}}{\sqrt{2\pi}}. \tag{3.61}$$

[1] `http://www.stat.yale.edu/~pollard/Books/Mini/Basic.pdf`

 

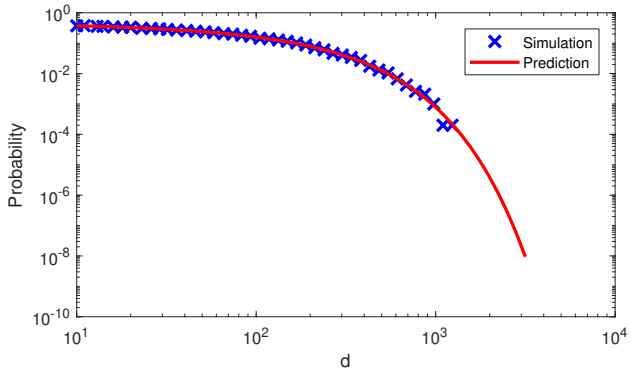


Figure 3.11: Empirical probability and the theoretically predicted value. In this experiment, we let $Y = \frac{1}{d}\sum_{j=1}^{d} r_j$ where $r_j \sim \mathcal{N}(0,1)$ is an iid Gaussian random variable. We plot the probability $\mathbb{P}[Y > \epsilon]$ as a function of $d$. Note the tightness of the prediction, and the rapid decaying behavior of the probability. The result shows that when $d$ grows, the value $Y$ is concentrated at 0.

Now, let $Y = \frac{1}{d}\sum_{j=1}^{d} w_j r_j$. Since a linear combination of Gaussian remains a Gaussian, it holds that $Y$ is Gaussian and

$$\mu = \mathbb{E}[Y] = 0, \quad \text{and} \quad \sigma^2 = \text{Var}[Y] = \frac{1}{d^2}\sum_{j=1}^{d} w_j^2 = \frac{\|\boldsymbol{w}\|^2}{d^2}. \tag{3.62}$$

Therefore, by substituting $\varepsilon = \sigma\epsilon$ we can show that

$$\mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \varepsilon\right] = \mathbb{P}[Y \geq \varepsilon] \leq \frac{\sigma}{\varepsilon}\frac{e^{-\frac{\varepsilon^2}{2\sigma^2}}}{\sqrt{2\pi}} = \frac{\|\boldsymbol{w}\|}{\varepsilon d\sqrt{2\pi}} \exp\left\{-d^2\frac{\varepsilon^2}{2\|\boldsymbol{w}\|^2}\right\}. \tag{3.63}$$

As $d \to \infty$, it holds that $\mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \varepsilon\right] \to 0$. That means, the probability of getting a "good attack direction" is diminishing to zero exponentially. Putting everything together we have the following theorem.

**Theorem 7.** *Let $\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$ be the decision boundary of a linear classifier, and let $\boldsymbol{x}_0 \in \mathbb{R}^d$ be an input data point. Suppose we attack the classifier by adding i.i.d. Gaussian noise $\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ to $\boldsymbol{x}_0$. The probability of a successful attack at a tolerance level $\varepsilon$ is $\mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \varepsilon\right]$, and such probability is upper bounded by*

$$\mathbb{P}\left[\frac{1}{d}\sum_{j=1}^{d} w_j r_j \geq \varepsilon\right] \leq \frac{\|\boldsymbol{w}\|}{\varepsilon d\sqrt{2\pi}} \exp\left\{-d^2\frac{\varepsilon^2}{2\|\boldsymbol{w}\|^2}\right\}. \tag{3.64}$$

*Therefore, as $d \to \infty$ it becomes increasingly more difficult for i.i.d. Gaussian noise to succeed in attacking.*

 

Why does the probability goes to zero so quickly? If we look at the inner product, we realize that

$$\boldsymbol{w}^T\boldsymbol{r} = \sum_{j=1}^{d} w_j r_j \tag{3.65}$$

follows Central Limit Theorem to a Gaussian. As $d$ increases, the random estimate $\sum_{j=1}^{d} w_j r_j$ approaches its expectation (which is zero in this case because $\mathbb{E}[r_i] = 0$ for all $i$) exponentially according to the Gaussian tail probability. This can be seen from the fact that there are positive and negative values of $r_i$'s in the sum. The more $r_i$'s we have, the more likely that the terms will cancel out each other.

The concentration inequality also suggests a surprising picture in high-dimensional space. Instead of having a half space, the event $\left|\frac{1}{d}\sum_{j=1}^{d} w_j r_j\right| < \epsilon$ is actually concentrated at the equator of a high-dimensional sphere. See Figure 3.12 for an illustration. Therefore, as $d$ increases, there is less and less probability that we can ever find a point NOT on the equator.

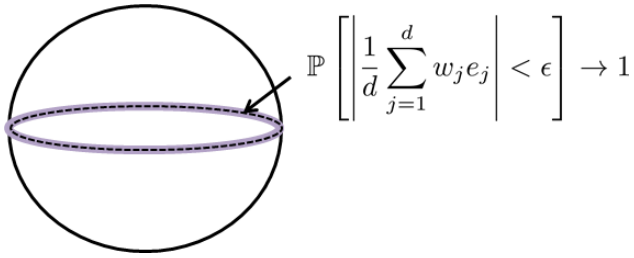


Figure 3.12: The magnitude of i.i.d. noise attack can be determined by the cosine angle, $\cos\theta = \boldsymbol{w}^T\boldsymbol{r}/\|\boldsymbol{w}\|_2\|\boldsymbol{r}\|_2$, which is equivalent to $\lambda^*/\sigma_e$.

What about the attack magnitude $\sigma_e$? Let $\cos\theta$ be the angle between $\boldsymbol{w}$ and $\boldsymbol{r}$. That is,

$$\frac{\boldsymbol{w}^T\boldsymbol{r}}{\|\boldsymbol{w}\|_2\|\boldsymbol{r}\|_2} = \cos\theta. \tag{3.66}$$

Since the shortest distance between $\boldsymbol{x}_0$ and the decision boundary $\boldsymbol{w}^T\boldsymbol{x} + w_0 = 0$ is $\lambda^*$, it follows that the ratio between $\lambda^*$ and $\sigma_e$, the distance from $\boldsymbol{x}_0$ to the decision boundary along the direction $\boldsymbol{r}$, is given by

$$\cos\theta = \frac{\lambda^*}{\sigma_e}. \tag{3.67}$$

Equating the results yields the following corollary.

 

**Corollary 5.** *Let $\boldsymbol{r} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$ be a random attack direction, and $\boldsymbol{w}$ be the linear discriminant function. The search step $\sigma_r$ to move an input $\boldsymbol{x}_0$ to the decision boundary along the direction $\boldsymbol{r}$ is*

$$\sigma_r = \frac{\lambda^* \|\boldsymbol{w}\|_2 \|\boldsymbol{r}\|_2}{\boldsymbol{w}^T \boldsymbol{r}}, \tag{3.68}$$

*where $\lambda^*$ defined in Equation (3.28) is the minimum distance from $\boldsymbol{x}_0$ to the decision boundary.*

Intuitively, Equation (3.68) provides a scaling to the shortest distance $\lambda^*$. The scaling constant $\frac{\|\boldsymbol{w}\|_2\|\boldsymbol{r}\|_2}{\boldsymbol{w}^T\boldsymbol{r}}$ is the reciprocal of the cosine angle. By Cauchy inequality we can show a lower bound on the ratio: $\frac{\|\boldsymbol{w}\|_2\|\boldsymbol{r}\|_2}{\boldsymbol{w}^T\boldsymbol{r}} \geq 1$. Therefore, if $\boldsymbol{r}$ aligns well with $\boldsymbol{w}$ then $\frac{\|\boldsymbol{w}\|_2\|\boldsymbol{r}\|_2}{\boldsymbol{w}^T\boldsymbol{r}}$ will be close to 1. In contrast, if $\boldsymbol{r}$ is almost perpendicular to $\boldsymbol{w}$, then $\frac{\|\boldsymbol{w}\|_2\|\boldsymbol{r}\|_2}{\boldsymbol{w}^T\boldsymbol{r}}$ can become big.

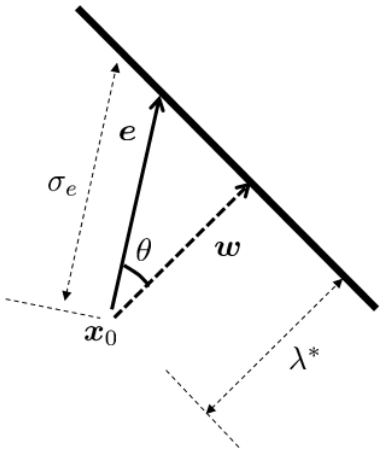


Figure 3.13: The magnitude of i.i.d. noise attack can be determined by the cosine angle, $\cos\theta = \boldsymbol{w}^T\boldsymbol{r}/\|\boldsymbol{w}\|_2\|\boldsymbol{r}\|_2$, which is equivalent to $\lambda^*/\sigma_r$.

The concentration inequality predicts that for large $d$, the probability for $\boldsymbol{w}^T\boldsymbol{r}$ to stay away from 0 is exponentially vanishing. This suggests that as $d \to \infty$, $\cos\theta \propto \boldsymbol{w}^T\boldsymbol{r} \to 0$ exponentially. Therefore, in order to use i.i.d. noise as an attack, the magnitude of the attack $\sigma_r$ has to grow exponentially with $d$.

 