# Adversarial learning Problem

EE6418: Game Theory for Engineering Application
Endterm Presentation

Course Instructor: Dr. Puduru Viswanadha Reddy
Presented By: Satwik Anand,Sujal Burad

November 29, 2024

1. **Adversarial attacks :** Adding perturbation or a small but effective change in the input data which degrades model performance, leading to potential harm in high-stakes application
2. **Real-World Examples of Adversarial Attacks:**
    1. **Images** - Slight manipulations (ex: adding noise to images) leads to mis-classification
    2. **Text** - Minor perturbations in text (e.g., replacing words with synonyms) causing incorrect outputs in sentiment analysis or translation tasks.
    3. **Safety Critical domains** - Autonomous vehicles, healthcare, and financial fraud detection
3. **Adversarial Learning Definition:** Focuses on building models that are resistant to adversarial manipulation (i.e., intentional perturbations of inputs to deceive a model).
4. **Need for Robustness:** In applications that require high accuracy and reliability, adversarial learning aims to ensure that models are resistant to these types of input manipulations.

Training samples $(x_1, y_1), \cdots, (x_n, y_n)$ with $x_i \in \mathbb{R}^p$ and $y_i = \pm 1$,

$$y = \text{sign}(w^\top x + b) \ ,$$

where $w \in \mathbb{R}^p$ and $b \in \mathbb{R}$ is the intercept. The algorithm learns $w$ and $b$ from the training samples. **MNIST dataset**

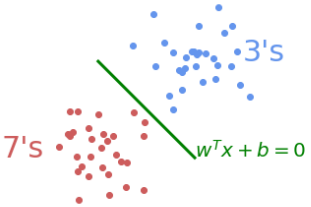



Figure: Linear model learns this

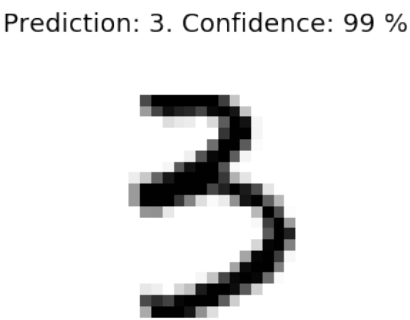
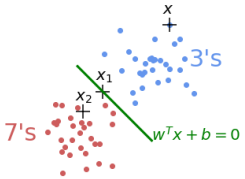



Figure: A training sample

The decision function is defined with a hyperplane $w^\top x + b$.




Prediction: 3. Confidence: 50 %

Figure: Moving the x in the hyperplane by doing:

$$x_1 = x - \frac{w^\top x + b}{w^\top w} w$$

Prediction: 7. Confidence: 99 %

Figure: Moving the x on other side of the hyperplane by doing:

$$x_2 = x - \alpha \frac{w^\top x + b}{w^\top w} w \ ,$$

where $\alpha > 1$:

This simple experiment shows that adversarial attacks are possible for one of the simplest classification model. Intuitively, increasing the model complexity, for instance by using a neural network, further worsen the problem.

In [22]:
```
classifiers
```

Out[22]:

| | Model | Precision | Recall | f1 score | Accuracy | ROC-AUC | Training time (sec) |
|---|---|---|---|---|---|---|---|
| 0 | XGBoost | 0.984398 | 0.989835 | 0.987075 | 0.989991 | 0.989835 | 10.97 |
| 1 | SVM (Linear) | 0.986752 | 0.980057 | 0.983343 | 0.987261 | 0.980057 | 0.15 |
| 2 | SVM (RBF) | 0.987905 | 0.978924 | 0.983305 | 0.987261 | 0.978924 | 10.24 |
| 3 | Logistic Regr. | 0.986108 | 0.964420 | 0.974630 | 0.980892 | 0.964420 | 2.02 |
| 4 | MultinomialNB | 0.985084 | 0.956294 | 0.969577 | 0.977252 | 0.956294 | 0.01 |
| 5 | Random Forest | 0.980821 | 0.948686 | 0.963358 | 0.972702 | 0.948686 | 3.64 |
| 6 | Gradient Boosting | 0.973134 | 0.946226 | 0.958675 | 0.969063 | 0.946226 | 19.83 |
| 7 | Decision Tree | 0.958695 | 0.945935 | 0.952076 | 0.963603 | 0.945935 | 1.63 |
| 8 | KNN | 0.968443 | 0.927610 | 0.945718 | 0.959964 | 0.927610 | 0.01 |

Figure: Normal Classifier space

**Algorithm 1: Label Flip Algorithm.**

1. set a threshold of 10% 15% 20% 25%
2. Make 4 train dataset with X% relabeled data both spam and ham
3. Train classifier with the train data
4. Evaluate the classifier with test data
5. Get performance metrics like , f1score , precision, recall, confusion matrix heatmap

Figure: Label Flipped Algorithm

Out[23]:

| | Threshold | Accuracy | Duration |
|---|---|---|---|
| 0 | 10 % | 0.961783 | 9.46 |
| 1 | 12 % | 0.940855 | 10.24 |
| 2 | 15 % | 0.873521 | 9.83 |
| 3 | 17 % | 0.829845 | 11.56 |
| 4 | 20 % | 0.725205 | 14.97 |

**Actual accuracy : 98.99 %**

Figure: Label Flipped Attack XGB

1. **Adversarial Interaction**:
   - **Learner (L)**: Trains a predictive model on data and aims to minimize classification error.
   - **Adversary (A)**: Modifies the data (e.g., adding noise) to mislead the learner's model.
2. **Game-Theoretic Framework**: This interaction can be modeled as a **game**:
   - **Learner's Goal**: Optimize model accuracy.
   - **Adversary's Goal**: Maximize the model's mis-classification.
3. **Traditional Assumption in Machine Learning**
   - **Assumption**: Training and test data come from **identical distributions**.
   - **Challenge**: In many real-world applications, this assumption is violated. The **adversary controls the distributions** of the test data, undermining the model performance.
4. **Key Question**: How do we formalize this interaction to predict or find the strategies for both the learner and the adversary

1. **Single-shot Game:** The learner controls the predictive model whereas the adversary exercises some control over the distribution of the input data. The optimal action for either player generally depends on both players' moves.
2. **Nash equilibrium:** For a game with $n$ players, the Nash Equilibrium condition is:

$$U_i(s_1, s_2, \ldots, s_i, \ldots, s_n) \geq U_i(s_1, s_2, \ldots, s_i', \ldots, s_n) \quad \forall s_i' \neq s_i$$

   Where:

   - $U_i$ is the utility or payoff function for player $i$,
   - $s_i$ is the strategy chosen by player $i$,
   - $s_i'$ is any alternative strategy for player $i$.

   **Interpretation:** A Nash equilibrium exists if, for all players, no player has an incentive to unilaterally deviate from their strategy when others are choosing their equilibrium strategies.
3. **Static Prediction Game:** The players make their decision based on predicting the action of others. Each player makes their decision simultaneously. It is a one time interaction.
4. **Infinite Game:** Continuous cost functions that leave players with infinitely many strategies to choose from.
5. **Minimax strategy:** Strategy which minimizes the cost under the worst possible movement of the opponent.
6. **Pure (deterministic) strategies:** A players makes a specific (deterministic) choice for every possible situation, without randomness or mixed options.

- Learner (v = +1), Adversary (v = -1).
- Players are constrained to select (i.e. deterministic) strategies.
- Both players access an input matrix $X$ with output $y$, drawn from a probability distribution:

$$q(X, y) = \prod_{i=1}^{n} q(x_i, y_i).$$

- **Learner**: Choose the parameter $a_{+1} \in A_{+1}$ of a linear model:

$$h_{a_{+1}}(x) = \mathbf{a_{+1}}^T x.$$

- **Adversary:** Chooses a transformation function $\phi_{a_{-1}}$ that maps any input matrix $X$ to an altered matrix $\phi_{a_{-1}}(X)$ . This transformation induces a transition from the input distribution $q$ to the test distribution $q_{\text{test}}$ with:

$$q(X, y) = q_{\text{test}}\left(\phi_{a_{-1}}(X), y\right) = \prod_{i=1}^{n} q_{\text{test}}\left(\phi_{a_{-1}}(X)_i, y_i\right).$$

- $$\phi_{a^{-1}}(X) = X + a_{-1}, \quad a_{-1} \in A_{-1} \subseteq \mathbb{R}^{m \times n}.$$

  The transformation adds a perturbation matrix $a_{-1}$ to $X$, i.e., the input pattern $x_i$ is subjected to a perturbation vector $a_{-1,i}$. If, for instance, inputs are word vectors, the perturbation matrix adds and deletes words.

- The possible moves $a = [a_{+1}, a_{-1}]$ constitute the joint action space $A = A_{+1} * A_{-1}$, which is assumed to be **nonempty, compact**, and **convex**. Action spaces $A_v$ are parameters of the game.
- Cost of action $a$ to players are $\theta_{+1}(a)$ and $\theta_{-1}(a)$ respectively.
- Each player has an individual loss function $l_v(y', y)$, where $y'$ is the value of decision function $h_{a_{+1}}$ and $y$ is the true label.
- **Antagonistic loss function:** $l_{+1} = -l_{-1}$.
- We will discuss non-antagonistic loss first.

**Player's cost function**

$$\theta_v(a_v, a_{-v}) = \sum_{i=1}^{n} l_v \left( h_{a_{+1}} \left( \phi_{a_{-1}}(X)_i \right), y_i \right) + \Omega_{a_v} \tag{1}$$

- The function is player-specific loss plus regularizer
- $\Omega_{a_{+1}}$ regularizes the learner's capacity of $h_{a_{+1}}$
- $\Omega_{a_{-1}}$ regularizes the adversary's amount of distortion that it may inflict on the data and thereby the extent to which an information payload has to be preserved.
- **Minimax solution:** $argmin_{a_v} max_{a_{-v}} \theta_v(a_v, a_{-v})$ minimizes the costs under the worst possible move of the opponent. (Optimal for a malicious opponent whose goal is to inflict maximally high costs on learner).

- When both players – learner, and adversary – behave rationally in minimizing their personal cost, we use Nash equilibrium to get an optimal choice of $a_v$.
- Nash equilibrium pair: $a^* = [a^*_{+1}, a^*_{-1}]$.
- For both players $v \in \{-1, +1\}$

**Catch:**

1. If the adversary behaves irrationally in the sense of inflicting high costs on the other player at the expense of incurring higher personal costs, then choosing an action according to the Nash equilibrium may result in higher costs than the minimax solution.
2. A game may not have an equilibrium point.
3. The game may possess multiple equilibria. If $a^* = [a^*_{+1}, a^*_{-1}]$, and $a' = [a'_{+1}, a'_{-1}]$ are distinct equilibria, and each player decides to act accordingly to one of them, then a combination $[a^*_v, a'_v]$ may be a poor joint strategy and may give rise to higher costs than a worst-case solution.

**A Unique Nash equilibrium is guaranteed to minimize the individual costs and be the optimal move**

**Requirement for the cost functions:**

- Must be **convex**, and **twice differentiable**.
- Regularizer $\Omega_{a_v}$ is strictly convex such as the $l_2$-norm regularizer.
- If both loss functions are monotonic in $y'$ with different monotonocities – one is monotonically increasing, and one is decreasing for any fixed $y$ – then the game has a unique Nash equilibrium that can be found efficiently.

**Theorem**

Let the cost functions be defined as in Equation (1) with **strictly convex regulairzer** $\Omega_{a_v}$, let action spaces $A_v$ be **nonempty, compact** and **convex subsets of finite-dimensional Euclidean spaces**. If for any fixed $y$, both loss functions $l_v(y', y)$ are monotonic in $y' \in \mathbb{R}$ with **distinct monotonicity, convex in** $y'$, and **twice differentiable** in $y'$, then a **Unique Nash Equilibrium exists**.

**Antagonistic loss:** $l_{+1} = -l_{-1}$ . We are not making any assumptions about the adversary's regularizers. Because of the regularizers, the game is still **not a zero-sum game**. A unique Nash equilibrium cannot be guaranteed to exist because the adversary's cost function is not necessarily strictly convex. We identify the unique Nash equilibrium, whenever it exists.

$$\theta_{+1}(a_{+1}, a_{-1}) = \sum_{i=1}^{n} l_{+1}(h_{a_{+1}}(\phi_{a_{-1}}(X)_i), y_i) + \Omega_{a_{+1}} \tag{2}$$

$$\theta_{-1}(a_{-1}, a_{+1}) = -\sum_{i=1}^{n} l_{+1}(h_{a_{+1}}(\phi_{a_{-1}}(X)_i), y_i) + \Omega_{a_{-1}} \tag{3}$$

**Theorem**

If the game with cost functions $\theta_{+1}$ and $\theta_{-1}$ defined in equation (2) and (3) has a unique Nash equilibrium $a^*$, then this equilibrium also satisfies $a^* = argmin_{a_{+1}} max_{a_{-1}} \theta_0(a_{+1}, a_{-1})$, where

$$\theta_0(a_{+1}, a_{-1}) = \sum_{i=1}^{n} l_{+1}\left(h_{a_{+1}}\left(\phi_{a_{-1}}(X)_i\right), y_i\right) + \Omega_{a_{+1}} - \Omega_{a_{-1}} \tag{4}$$

# Things done so Far

1. We studied game in which each player's cost function consists of a data-dependent loss and regularizer. A learner produces a linear model while an adversary chooses a transformation matrix to be added to the data matrix.
2. A **Unique Nash Equilibrium** exists in the case of **regularized, non-antagonistic loss functions** that are **convex, twice differentiable, and have distinct monotonocity**.
3. For **Antagonistic loss function** with **arbitrary regularizer** a Unique Nash equilibrium **may or may not exist**.

1. The idea and game theory model discussed so far is simple in nature.
2. The classifiers discussed so far take particular forms (e.g., logistic regression, where the defender's choice is the set of weights) and identify conditions under which a unique pure Nash equilibrium exists.
3. An example of the setting used above is like defender choosing a logistic regression model and adjusting the weights depending on the attack.
4. The idea is to develop a broader scope where the classifiers are not restricted a priori and derive the set of classifiers used at equilibrium.
5. Thus,we move from notion of Pure Strategy Nash Equilibrium to Mix Strategy Nash Equilibrium.

1. We will analyse a new game theory classification model which captures the adaptive nature of attacker.
2. Similar to the previous model we will analyze a non-zero sum game which we will further transform into pseudo zero sum game for the computation of Nash Equilibrium.
3. We will be mainly concerned with two types of attacks -Poisoning attack and Evasion attack.
4. In poisoning attacks, the attacker can alter the training set while in evasion attacks, the classifier is fixed and the attacker simply attempts to reverse-engineer it to find an attack that suits his goal while being classified as non-attacker.

# The idea of attack vectors

- We consider a strategic situation between a defender and an agent that may either be an attacker with **probability p** or **a non-attacker with probability 1-p.**
- The attacker tends to exploit this uncertainty by attacking in a way so that it gets classified as a non attacker.
- Some Real life examples include in spam classification, the spammer (the attacker) might change the frequency of words included in an email to evade spam filters.
- To model this we use **attack vectors** that contains all features used by the defender for classification; e.g., average number of followers, number of retweets, and others (in social networks fraud),

**Some important terms**

- $p \in [0, 1]$: Probability that the agent is an attacker;
- $v$ is an attack vector in $\mathcal{V}$, where $\mathcal{V}$ is the set of all possible attack vectors.
- $c_d \in \mathbb{R}^+$: Cost of detection;
- $c_{fa} \in \mathbb{R}^+$: Cost of false alarm;
- $P_N : \mathcal{V} \to [0, 1]$: Probability measure that describes the non-attacker's distribution on $\mathcal{V}$;
- $R : \mathcal{V} \to \mathbb{R}^+$: The reward function.
- $\mathcal{C}$ is a classifier chosen by defender, where $\mathcal{C} \subseteq 2^{|\mathcal{V}|}$ is the set of all possible classifiers.

**Payoff function of attacker**

$$U_A(v, c) = R(v) - c_d \cdot \mathbf{1}_{c(v)=1},$$

where $R(v)$ is the reward obtained on conducting attack $v$, and $c_d$ is the cost incurred when the classifier $c$ detects it.

**Scaled Payoff function of defender**

$$U_D(v, c) = -U_A(v, c) - \sum_{v' \in \mathcal{V}} \frac{1-p}{p} \cdot c_{fa} \cdot \mathbf{1}_{c(v')=1} \cdot P_N(v')$$

where we are assuming that the cost incurred by attacker on detection is same as the benefit of defender when attacker detected and cost of detection is same regardless of attack vector.

**The classification game** $G = (\mathcal{V}, C, p, c_d, c_{fa}, P_N, R(\cdot))$ is a two-player game between the attacker and the defender, where:

- The strategy space of the attacker is $\mathcal{V}$.
- The strategy space of the defender is $C \subseteq 2^{\mathcal{V}}$.

The payoffs are given in the previous slide.

Now, we will view this as a normal form game, as the non-attacker's strategy is not strategic, and find the Mixed Strategy Nash Equilibrium (MSNE).

But before that, we focus on reducing the strategy space of the defender, which will, in turn, help reduce the strategy space of the attacker.

We will be interested in **mixed strategy equilibria,** where:

- The attacker randomizes across multiple attack vectors with a probability distribution $\alpha$ **on** $\mathcal{V}$**.**
- The defender randomizes across multiple classifiers with a distribution $\beta$ **on** $C$**.**

**The expected payoffs for the attacker and defender are then given by:**

$$U_A(\alpha, \beta) = \sum_{v \in \mathcal{V}} \sum_{c \in C} \alpha_v U_A(v, c) \beta_c \tag{3}$$

$$U_D(\alpha, \beta) = \sum_{v \in \mathcal{V}} \sum_{c \in C} \alpha_v U_D(v, c) \beta_c \tag{4}$$

The pair of probability measures $(\alpha, \beta)$ on $\mathcal{V}$ and $C$ respectively is a Nash equilibrium (NE) of game $G$ if each player's mixed strategy is a best response to the other player's mixed strategy, i.e.,

$$U_A(\alpha, \beta) \geq U_A(\hat{\alpha}, \beta) \quad \text{for every probability distribution } \hat{\alpha} \text{ over } \mathcal{V}, \tag{5}$$

and

$$U_D(\alpha, \beta) \geq U_D(\alpha, \hat{\beta}) \quad \text{for every probability distribution } \hat{\beta} \text{ over } C. \tag{6}$$

**Set of Threshold Classifiers:**

$$C_T = \left\{ c \in C : c(v) = \mathbf{1}_{R(v) \geq t},\ \forall v \in \mathcal{V} \text{ for some } t \in \mathbb{R} \right\}$$

**Probability of Detection Function:**
We define the probability of detection as the probability of class 1 classification (or detection) given the attack vector $v$ and the defender's strategy $\beta$.

$$\pi^d_\beta(v) = \sum_{c \in C} \beta_c \mathbf{1}_{c(v)=1}, \quad \forall v \in \mathcal{V}$$

Till now, we have seen that the defender's strategy space is the set of classifiers $\mathcal{C}$, whose size is $2^{|\mathcal{V}|}$.

We now propose a theorem that reduces the size of the strategy space to $|\mathcal{V}| + 1$.

**Theorem 1**

For any Nash equilibrium $(\alpha, \beta)$ of $G = (\mathcal{V}, \mathcal{C}, p, c_d, c_{fa}, P_N, R(\cdot))$, there exists a Nash equilibrium of

$$G^T = (\mathcal{V}, \mathcal{C}_T, p, c_d, c_{fa}, P_N, R(\cdot))$$

with the same $\alpha$ and equilibrium payoff pair, and the same $\pi_d$ in the support of the non-attacker's distribution.

We will be not exactly deriving the theorem word to word but will form the building blocks of the derivation.

**Lemma 1**

For any strategy profile $(\alpha, \beta)$ of the game $G = (V, C, p, c_d, c_{fa}, PN, R(\cdot))$, the expected payoffs of the players depend on $\beta$ only through the probability of detection function $\pi_d^{\beta}(\cdot)$:

**Attacker's Expected Utility:**

$$U_A(\alpha, \beta) = \sum_{v \in V} \left[\alpha_v R(v) - c_d \alpha_v \pi_d^{\beta}(v)\right]$$

**Defender's Expected Utility:**

$$U_D(\alpha, \beta) = -U_A(\alpha, \beta) - \frac{1-p}{p} c_{fa} \sum_{v \in V} P_N(v) \pi_d^{\beta}(v)$$

**Lemma 2**

For any function $f : V \to [0, 1]$, there exists a probability measure $\beta$ over $C = 2^{|V|}$ such that

$$\pi_d^{\beta}(v) = f(v), \quad \forall v \in V.$$

**Intuition**:We can assume that vectors are arranged in non-decreasing order of probability of detection, i.e., $f(v_i) \leq f(v_j)$ for $i \leq j$. Then we can assign $\beta_c$ as follows:

- Assign $f(v_1)$ to the classifier $c_1$ such that $c_1(v) = 1$ for all $v \in V$.
- Assign $f(v_2) - f(v_1)$ to the classifier $c_2$ such that $c_2(v) = 1$ for all $v \in V \setminus \{v_1\}$.
- Similarly, assign $f(v_{|\mathcal{V}|}) - f(v_{|\mathcal{V}|-1})$ to the classifier $c_{|\mathcal{V}|}$ that classifies only $v_{|\mathcal{V}|}$ as an attacker.

If any remaining weight is left, it is assigned to the classifier that classifies everything as a non-attacker.

**Lemma 3**

If $(\alpha, \beta)$ is a NE of $G = (V, C, p, c_d, c_{fa}, PN, R(\cdot))$ that yields a probability of detection function $\pi_d$, then $\forall v \in V$ such that $PN(v) > 0$, one of the following three cases holds:

- **Case 0:** $\alpha_v = 0$ and $\pi_d(v) = 0$,
- **Case 1:** $\alpha_v > 0$ and $\pi_d(v) = 0$,
- **Case 2:** $\alpha_v > 0$ and $\pi_d(v) > 0$.

Furthermore, $R(v_0) \leq R(v_1) < R(v_2)$, for any strategies $v_0, v_1, v_2$ in Cases 0, 1, and 2, respectively.//
**Proof(Intuition):**This can be easily proved using Theorem 7.1 of Textbook 1 where it is proved that for a strategy profile to be a MSNE all the pure strategies in the support should have same utility function and greater than equal to the strategy that is not in support.

**Lemma 4**

Let $v_1, v_2 \in V$ be such that $P_N(v_1), P_N(v_2) > 0$. In any NE $(\alpha, \beta)$ of $G = (V, C, p, c_d, c_{fa}, P_N, R(\cdot))$ that yields a probability of detection function $\pi_d$, we have:

$$R(v_1) \leq R(v_2) \implies \pi_d(v_1) \leq \pi_d(v_2).$$

**Proof:**Same Theorem 7.1 can be used to prove

The first step in the proof of **Lemma 2** involved reindexing attack vectors to have non-decreasing detection probabilities. By **Lemma 4,** attack vectors ranked in non-decreasing reward already satisfy this property (if $PN(v) > 0, \forall v \in V$). Thus, we can skip the reindexing step and directly describe the classifiers $c \in C$ that are given positive weight.

- **Classifier** $c_1$ detects all attack vectors and acts as a threshold classifier with a threshold equal to the reward of the vector with the smallest reward (or smallest detection probability).
- **Classifier** $c_2$ detects all vectors except the one with the smallest reward and acts as a threshold classifier with a threshold equal to the second smallest attack reward.
- This process continues until **classifier** $c_{|V|}$, which detects only the attack vector with the highest reward (threshold equal to the highest reward).
- Any remaining weight is given to classifier $c_{|V|+1}$, which always classifies the agent as a non-attacker. This is equivalent to a threshold classifier with a threshold larger than the highest reward $R(v_{|V|})$.

This leads to the following corollary and building block of proof of Theorem 1:

**Corollary 2:** For any NE $(\alpha, \beta)$ of $G = (V, C, p, c_d, c_{fa}, PN, R(\cdot))$ that results in a non-decreasing probability of detection $\pi_d^{\beta}$, there exists a NE $(\alpha, \hat{\beta})$ of $G$, where $\hat{\beta}_c = 0, \forall c \in C \setminus C_T$, and $\pi_d^{\beta}(v) = \pi_d^{\hat{\beta}}(v), \forall v \in V$.

Now that we have successfully reduced the strategy space of the classifier from $\mathcal{C}$ to $\mathcal{C}_{\mathcal{T}}$, we can similarly reduce the strategy space of the attacker.But before let us look at equilibrium strategy of attacker.The significance of the lemma will be that attacker mixes an attack vector v proportionally with non-attack distribution.

**Lemma 5**

If $(\alpha, \beta)$ is a NE of $G = (\mathcal{V}, \mathcal{C}, p, c_d, c_{fa}, P_N, R(\cdot))$, then for all $v \in \mathcal{V}$ such that $0 < \pi_d(v) < 1$,

$$\alpha_v = \frac{1-p}{p} \cdot \frac{c_{fa}}{c_d} \cdot P_N(v).$$

**Proof**

Consider $v_i \in \mathcal{V}$ with $\pi_d(v_i) \in (0,1)$. By definition, there exists $c_i, c^* \in \mathcal{C}$ such that $c_i(v_i) = 1$, $\beta_{c_i} > 0$, and $c^*(v_i) = 0$, $\beta_{c^*} > 0$. Without loss of generality, assume $c_i(v) = c^*(v)$ for all $v \in \mathcal{V} \setminus \{v_i\}$.
At equilibrium, the defender's payoffs for $c_i$ and $c^*$ are equal:

$$-c_d \alpha_{v_i} + \frac{1-p}{p} c_{fa} P_N(v_i) = 0.$$

Solving for $\alpha_{v_i}$, we obtain:

$$\alpha_{v_i} = \frac{1-p}{p} \cdot \frac{c_{fa}}{c_d} \cdot P_N(v_i).$$

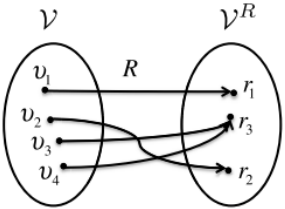



Figure: Attacker's reduced strategy space

We observe that since the defender uses only threshold classifiers, the attack vector space $\mathcal{V}$ can be transformed into the reduced space $\mathcal{V}^{\mathcal{R}}$, where $\mathcal{V}^{\mathcal{R}} = \{r \in \mathbb{R}^+ : r = R(v), \text{ for some } v \in \mathcal{V}\}$.

Similarly, the non-attacker's probability measure can be reduced to a probability measure on $\mathcal{V}^{\mathcal{R}}$, defined as:

$$P_N^R(r) = \sum_{v \in \mathcal{V}} P_N(v) \mathbf{1}_{R(v)=r}.$$

## Reduced Game theoretic model

The reduced game $\mathcal{G}^{\mathcal{V}^{\mathcal{R}},\mathcal{C}_{\mathcal{T}}} = (\mathcal{V}^{\mathcal{R}}, \mathcal{C}_{\mathcal{T}}, p, c_d, c_{fa}, P_N^R)$ is defined as the game between the attacker choosing $r \in \mathcal{V}^{\mathcal{R}}$ and the defender choosing $c \in \mathcal{C}_{\mathcal{T}}$, where the utilities are adapted is as follows:

$$U_A(r, c) = r - c_d \mathbf{1}_{c(r)=1},$$

$$U_D(r, c) = -U_A(r, c) - \frac{1-p}{p} c_{fa} \sum_{r \in \mathcal{V}^{\mathcal{R}}} P_N^R(r) \mathbf{1}_{c(r)=1}.$$

and the following holds true If $(\alpha, \beta)$ is a NE of $\mathcal{G}^T = (\mathcal{V}, \mathcal{C}_{\mathcal{T}}, p, c_d, c_{fa}, P_N, R(\cdot))$, then $(\alpha^*, \beta)$ is a NE of $\mathcal{G}^{\mathcal{V}^{\mathcal{R}},\mathcal{C}_{\mathcal{T}}} = (\mathcal{V}^{\mathcal{R}}, \mathcal{C}_{\mathcal{T}}, p, c_d, c_{fa}, P_N^R)$ with the same equilibrium payoff pair, where

$$\alpha^*_{r_i} = \sum_{v_j \in \mathcal{V}, R(v_j)=r_i} \alpha_{v_j}, \quad \forall r_i \in \mathcal{V}^{\mathcal{R}}.$$

Since our Reduced strategy game is a finite strategy game,it has a mixed strategy Nash Equilibrium but finding that has high computational complexity.We will look at an approach to bring that down.We define $\tilde{\mathbf{\Lambda}}$ to be the cost matrix of the attacker, with

$$\tilde{\Lambda}(i,j) = c_d \cdot \mathbf{1}_{r_i \geq r_j} - r_i, \quad i \in \{1,\ldots,|\mathcal{V}^{\mathcal{R}}|\},\ j \in \{1,\ldots,|\mathcal{V}^{\mathcal{R}}|+1\}.$$

Certain computations are simplified by using a matrix with only positive entries. Therefore, we define

$$\Lambda = \tilde{\Lambda} + (r_{|\mathcal{V}^R|} + \epsilon) \cdot \mathbf{1}_{|\mathcal{V}^R| \times (|\mathcal{V}^R|+1)},$$

where $\epsilon > 0$. Adding a constant to the players' payoff does not affect their best responses, and hence does not change the equilibrium strategies. In the following, the pair $(\alpha, \beta)$ will denote the probability measures of the attacker and defender on $\mathcal{V}^R$ and $\mathcal{C}_T$, respectively. The attacker's expected cost is given by $\alpha^\top \Lambda \beta$. The defender's expected payoff is given by $\alpha^\top \Lambda_{\text{eq}} \beta$, with

$$\Lambda_{\text{eq}} = \Lambda - \mathbf{1}_{|\mathcal{V}^R|} \cdot \boldsymbol{\mu}^\top,$$

where $\boldsymbol{\mu}$ represents the false alarm penalty vector for the defender, with elements $\mu_i = \frac{1-p}{p} c_{\text{fa}} \sum_{k \geq i} P_N^R(r_k)$.

**Proof:** The defender's expected utility in $G^{R,T}$ is given by:

$$U_D(\alpha, \beta) = \alpha^\top \Lambda_{\text{eq}} \beta,$$

where

$$\Lambda_{\text{eq}} = \Lambda - \mathbf{1}_{|\mathcal{V}^R|} \cdot \boldsymbol{\mu}^\top, \quad \mu_i = \frac{1-p}{p} c_{fa} \sum_{k \geq i} P_N^R(r_k).$$

The attacker's expected cost is given by:

$$U_A(\alpha, \beta) = \alpha^\top \Lambda \beta.$$

Rewriting $U_D(\alpha, \beta)$ in terms of $U_A(\alpha, \beta)$:

$$U_D(\alpha, \beta) = -U_A(\alpha, \beta) - \alpha^\top (\mathbf{1}_{|\mathcal{V}^R|} \cdot \boldsymbol{\mu}^\top) \beta.$$

The term

$$-\alpha^\top (\mathbf{1}_{|\mathcal{V}^R|} \cdot \boldsymbol{\mu}^\top) \beta$$

is independent of the attacker's strategy $\alpha$.

**Conclusion:** The game is best-response equivalent to a zero-sum game, where:

- The defender minimizes $U_A(\alpha, \beta)$,
- The attacker maximizes $U_A(\alpha, \beta)$.
- Mixed Strategy Nash Equilibrium of a Zero-Sum game can be easily found using LP in a reduced time complexity using min max strategy.

1. Nash Equilibrium of Static Prediction Game
2. A Game theoretical Framework for Adversarial Learning
3. A Game-Theoretic Analysis of Adversarial Classification
4. Attacking a linear model
5. Adversarial attack on Image Classification model
6. Spam filtering Adversarial Attack