ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ФРАНКА
ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ ТА ІНФОРМАТИКИ
КАФЕДРА ОБЧИСЛЮВАЛЬНОЇ МАТЕМАТИКИ

КУРСОВА РОБОТА

на тему:

# Аналіз атак на лінійні моделі машинного навчання

студента III курсу
групи ПМп-31
Середовича Віктора

Науковий керівник:
доцент Ю.А.Музичук

Завідуючий кафедри обчислювальної
математики
проф.

Львів — 2020

# Зміст

# Розділ 1

# Вступ

Алгоритми машинного навчання активно використовується у різних областях нашого життя та дуже добре демонструють себе у задачах класифікації. Однак, виявляється, що ці алгоритми класифікації є досить вразливими навіть до незначних, правильно вибраних, пертурбацій вхідної інформації. Такі модифіковані приклади, які навмисно змінюють для того щоб вони були хибно передбаченні називають змагальними прикладами або adversarial samples. В деяких випадках ці зміни можуть бути настільки малі, що для людини вони зовсім не будуть помітні, однак для класифікатор моделі буде робити хибне передбачення. Така вразливість може бути дуже небузпечною, коли алгоритми машинного навчання застосовуються в критичних для здоров'я людини середовищах. Саме тому ця область привертає до себе багато уваги і порбеює досліджень.

В межах теми цієї роботи будуть розглядатись різні методи атак на лінійні моделі машинного начання які будуть аналізуватись і порівнюватись між собою. Також будуть запропоновані методи захисту моделі від моделей.

## 1.1 Постановка задачі

*Мета* даної роботи полягає у тому, щоб дослідити ефективність атак на лінійні моделі машинного навчання, та визначити методи захисту від них.

Виходячи з мети, визначені завдання роботи:

- Реалізувати лінійну модель машинного навчання
- Дослідити різні методи генерування змагальних прикладів
- Проаналізувати ефективнівсть атак
- Визначити методів захисту

## 1.2 Основні поняття

В цій секції будуть розглянуті основні поняття, які будуть використовуватись в ході роботи.

**Змагальний приклад (Adversarial sample)**

Нехай існує класифікатор $f(x) : x \to y$, де $x \in X, y \in Y$, який передбачає значення $y$ для вхідного $x$. Метою змагального прикладу є знайти такий $x^*$, який знаходиться в околі $x$, але хибно визначається класифікатором. Зазвичай максимальний рівень шуму який може бути в змагальному прикладі може бути не більше за певну $L_p$ норму $\|x^* - x\|_p < \epsilon$, де $p = 1, 2, \infty$. В межах даної роботи для визначення рівня пертурбації буде виористовуватись саме $L_\infty$ норма.

**Націлені атаки (Untargeted Attacks)**

Націлені атаками є атаки метою яких є внести в приклад $x$, який правильно передбачається як $f(x) = y$ незначний шум так, щоб класифікатор зробив хибне передбачення $f(x^*) \neq y$.

**Ненацілені атаки (Targeted Attacks)**

Ненацілені атаками називають такі атаки, метою яких є внести у вхідний приклад $x$ такі пертурбації щоб класифікатор передбачив якийсь конкретний клас $f(x^*) = y^*$, де $y^*$ є заданою ціллю $y^* \neq y$.

**Атаки на закриту модель (White Box attacks)**

Атаки на закриту модель описують такий сценарій, коли в нападника є повний доступ до параметрів і градієнтів моделі.

**Атаки на відкриту модель (Black Box attacks)**

До атак на відкриту модель выдносять атаки, при яких в нападника нема доступу до параметрів моделі.

# Розділ 2

# Модель і датасет

## 2.1 Мультикласова логістична регресія

В якості лінійного методу машинного навчання, на котрий будуть сдійснюватись атаки буде використовуватись модифікований алгоритм логістичної регресії для мультикласової класифікації.

Нехай маємо набір тренувальних даних:

$$(x^{(1)}, y^{(1)}), \quad (x^{(2)}, y^{(2)}), \quad ... \quad , (x^{(m)}, y^{(m)})$$

$$X = \begin{bmatrix} \vdots & \vdots & & \vdots \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad x \in \begin{bmatrix} x_1 \\ \dots \\ x_n \end{bmatrix}$$

$$Y = \begin{bmatrix} \vdots & \vdots & & \vdots \\ y^{(1)} & y^{(2)} & \dots & y^{(m)} \\ \vdots & \vdots & & \vdots \end{bmatrix} \quad y \in \begin{bmatrix} y_1 \\ \dots \\ y_C \end{bmatrix} \quad y_1, ..., y_C$$

$X$ - матриця в якої кожен стовпець є набором характеристик $i-$ого прикладу, $i = 1, \dots, m$

$Y$ - матриця класів в якої кожен стовпець це масив розмірності $C$, з одиницею на місті справжнього класу

$m$ - кількість характеристик

$n$ - кількість характеристик в кожному прикладі

$C$ - кількість класів

Задача прикладу $x \in R^n$, знайти $\hat{y} = P(y = 1 \mid x), \quad 0 \leq \hat{y} \leq 1$

$$\hat{y} = softmax(\omega^T x + b), \quad \text{де } \omega \in R^n, b \in R \text{ - невідомі параметри}$$

Функція активації матиме вигляд:

$$softmax(z) = \frac{e^z}{\sum_{k=1}^{C} e_k^z} \tag{2.1}$$

Для кожного прикладу з тренувального датасету потрібно обчислити:

$$z^{(i)} = \omega^T x^{(i)} + b, \text{ де } y^{(i)} = softmax(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{C} e_k^z}, \text{ так щоб } \hat{y}^{(i)} \approx y^{(i)}$$

В якості функції втрати буде використовуватись функція кросс-ентропії:

$$\xi(y, x) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) \tag{2.2}$$

Задача полягає в тому щоб знайти параметри $w \in R_x^n, b \in R$ що мінімізують функцію $J(\omega, b)$. Для цього будемо використовувати алгоритм градієнтного спуску.

## 2.2 Датасет

Для аналізу атак на лінійні методи машинного навчання буде використовуватись MNIST база даних яка складаэться з 60тис. тренувальних зображень та 10 тис. тестувальних рукописних цифр.

Рис. 2.1: MNIST датасет рукописних цифр

На її основі буде здійснюватись тренування моделі та аналіз атак.

## 2.3 Тренування моделі

# Розділ 3

# Методи атак

В цьому розділі будуть розглянуті деякі методи генерування змагальних зразків, які були реалізовані і протестовані на лінійній моделі натренованій в розділі 2.1.

Далі, під час опису будуть використовуватись такі позначення:

- $\boldsymbol{X}$ - це зображення, зазвичай це 3-D тензор (ширина $\times$ висота $\times$ глибина). В нашому випадку має розміри $(28 \times 28 \times 1)$ пікселей, яке складається з додатніх цілих чисел в межах $[0, 255]$.
- $\boldsymbol{X}^*$ - згенерований змагальний приклад для зображення $\boldsymbol{X}$
- $y_{true}$ - привильний клас для зображення $\boldsymbol{X}$.
- $J(\boldsymbol{X}, y)$ - штрафна функція моделі. Ваги моделі $\omega$ навмисно не будемо вказувати, вважатимемо їх сталими, як значення які будуть отримані з моделі машинного навчання. В нашому випадку штрафною функцією є функція кросс-ентропії 2.2.
- $Clip_{\boldsymbol{X},\epsilon}\{\boldsymbol{X}^*\}$ - функція яка виконує по-піксельне обрізання зображення $X^*$ так, щоб результат був в $L_\infty$ $\epsilon$ - околі вхідного зображення $\boldsymbol{X}$. Рівняння для цього виглядає наступним чином:

$$Clip_{X,\epsilon}\{\boldsymbol{X}^*\}(x, y, z) = min\Big\{255, \boldsymbol{X}(x, y, z) + \epsilon, max\{0, \boldsymbol{X}(x, y, z) - \epsilon, \boldsymbol{X}^*(x, y, z)\}\Big\}, \tag{3.1}$$

де $\boldsymbol{X}(x, y, z)$ - це значення каналу $z$ зображення $\boldsymbol{X}$ з координатами $(x, y)$.

## 3.1 Атака шумом

Найпершим і одним з найпростіших методів атак є *атака шумом*. Цей тип атак можна характеризувати як *ненаціленну атаку на закриту модель*.

Основна ідея полягає в тому щоб згенерувати приклад який складається з випадково згенерованих значень і додати його до зображення на якому буде здійснюватись

класифікація. В випадку нашої моделі шумом буде деяке $\hat{r}$ розміром $784 \times 1$, зі значеннями в межах $[0, 255]$. При цьому $\hat{r}$ має задаватись так, щоб задовільняти деяку задану норму $\|\hat{r}\|_2 < \epsilon$

## 3.2 FGSM

Одним з найпоширініших методів для генерування змагальних зразків машинного навчання є Fast Gradient Sign Method. В перше він був представлений в роботі [1]. Ідея цього методу полягає в тому, щоб знайти такий adversarial приклад $x^*$ який максимізую функцію втрати $J(x^*, y)$ до певного $L_\infty$ обмеження. Цього можна досягти один раз використавши використавши зворотне поширення:

$$X^* = X + \epsilon \cdot sign(\Delta_x J(x, y)), \tag{3.2}$$

де $\epsilon$ - деякий гіперпараметер.

Для того щоб застосувати цей метод для логістичної регресії необхідно обчислити градієнт від функції кросс-ентропії 2.2. Для цього спочатку знайдемо похідну від $softmax$ функції:

$$z_i = \omega^T x^{(i)} + b; \quad \hat{y}_i = a_i = softmax(z_i); \quad softmax(z_i) = \frac{e^{z_i}}{\sum_{k=1}^{C} e_k^z};$$

Якщо $i = j$,

$$\frac{\partial \hat{y}_i}{\partial z_j} = \frac{\partial \frac{e^{z_i}}{\sum_{k=1}^{C} e_k^z}}{\partial z_j} = \frac{e^{z_i} \sum_{k=1}^{C} e^{z_k} - e^{z_j} e^{z_i}}{(\sum_{k=1}^{C} e^{z_k})^2} =$$
$$= \frac{e^{z_j}}{\sum_{k=1}^{C} e^{z_k}} \times \frac{(\sum_{k=1}^{C} e^{z_k} - e^{z_j})}{\sum_{k=1}^{C} e^{z_k}} = y_i(1 - y_j)$$

Якщо $i \neq j$,

$$\frac{\partial \hat{y}_i}{\partial z_j} = \frac{\partial \frac{e^{z_i}}{\sum_{k=1}^{C} e_k^z}}{\partial z_j} = \frac{0 - e^{z_j} e^{z_i}}{\sum_{k=1}^{C} e^{z_k}} \times \frac{e^{z_i}}{\sum_{k=1}^{C} e^{z_k}} = -y_i y_j$$

Далі обчислюємо похідну від функції кросс-ентропії:

$$\xi(y, x) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i)$$

$$\frac{\partial \xi}{\partial z_i} = -\sum_{j=1}^{C} \frac{\partial y_j \log(\hat{y}_j)}{\partial z_i} = -\sum_{j=1}^{C} y_j \frac{\partial \log(\hat{y}_j)}{\partial z_i} = -\sum_{j=1}^{C} y_j \frac{1}{\hat{y}_i} \cdot \frac{\partial \hat{y}_i}{\partial z_i} =$$

$$= -\frac{y_i}{\hat{y}_i}\frac{\partial \hat{y}_i}{\partial z_i} - \sum_{j \neq i} \frac{y_j}{\hat{y}_j}\frac{\partial \hat{y}_j}{\partial z_i} = -\frac{y_i}{\hat{y}_i}\hat{y}_i(1-\hat{y}_i) - \sum_{j \neq i} \frac{y_j}{\hat{y}_j}(-\hat{y}_i\hat{y}_j) =$$

$$= -y_i + y_i\hat{y}_i + \sum_{j \neq i} y_j\hat{y}_i = -y_i + \sum_{j=1}^{C} y_j\hat{y}_i = \hat{y}_i - y_i,$$

де i = 1, ..., C

Тепер можна обчислити градієнт від штрафної функції по $x$:

$$\frac{\partial \xi}{\partial x_i} = \frac{\partial \xi}{\partial z_i}\frac{\partial z_i}{\partial x_i} = (\hat{y}_i - y_i)\frac{\partial(\omega^T x_i + b)}{\partial x_i} = (\hat{y}_i - y_i)\omega^T = dz \cdot \omega^T$$

Маючи всі наведені обчислення, можна записати алгоритм знаходження змагального прикладу [1].

**Algorithm 1** *FGSM*

1: **Input:** Приклад $x$, клас $y_{true}$, класифікатор $f$, параметри: $\omega, b$.
2: **Input:** значення пертурбації $\epsilon$.
3: **Output:** Adversarial $x^*$ з нормою $\|x^* - x\|_\infty \leq \epsilon$;
4: $z = \omega^T x + b$;
5: $\hat{y} = softmax(z)$;
6: $\delta = (\hat{y} - y_{true}) \cdot \omega^T$;
7: $x^* = Clip_{X,\epsilon}\{x + \alpha \cdot sign(\delta)\}$;
8: **return** $x^* = x_T^*$.

## 3.3 I-FGSM

Даний метод є модифікацією попереднього, особливість якого полягає в тому, що для знахождення змагального прикладу буде використовуватись ітераційний підхід. Ми застосовуємо звичайний метод декілька разів з певним невеликим кроком $a$ і після кожного кроку використовуємо *clip* функцію, для того щоб переконатись що значення знаходяться в $\epsilon$-околі оригінального зображення.

Ітераційної алгоритм швидкого градієнтного спуску буже мати вигляд [2].

**Algorithm 2** $I - FGSM$

**Input:** Приклад $x$, значення класу $y_{true}$, класифікатор $f$
**Input:** значення пертурбації $\epsilon$, кількість ітерації $T$.
**Output:** Adversarial $x^*$ з нормою $\|x^* - x\|_\infty \leq \epsilon$;
$a = \epsilon/T$;
$x^* = x$;
**for** $t = 0$ *to* $T - 1$ **do**
    $x^* = Clip_{X,\epsilon}\{x^* + \alpha \cdot sign(\Delta_x J(x, y))\}$;
**end for**
**return** $x^* = x^*_T$.

## 3.4 TI-FGSM

TI-FGSM або як його ще часто називаюсь Projected Gradient Descent є націленою версією попереднього алгоритму, а отже він дає можливість обрати клас який має передбачити модель. Два попередні методи описаних вище намагалися максимізувати функцію втрати для правильного класу, не вказуючи яким буде неправильний класс. Цей метод - навпаки, мінімізує функцію втрати для деякого заданого неправильного класу. На сам алгоритм це вплине так, що на етапі додавання пертурбації у зображення знак буде змінений на протилежний.

**Algorithm 3** $TI - FGSM$

**Input:** Приклад $x$, значення класу $y_{true}$, класифікатор $f$
**Input:** значення пертурбації $\epsilon$, кількість ітерації $T$.
**Output:** Adversarial $x^*$ з нормою $\|x^* - x\|_\infty \leq \epsilon$;
$a = \epsilon/T$;
$x^* = x$;
**for** $t = 0$ *to* $T - 1$ **do**
    $x^* = Clip_{X,\epsilon}\{x^* - \alpha \cdot sign(\Delta_x J(x, y))\}$;
**end for**
**return** $x^* = x^*_T$.

## 3.5 MI-FGSM

Цей метод

**Algorithm 4** $MI - FGSM$

**Input:** Приклад $x$, значення цілі y, класифікатор $f$.;
**Input:** значення пертурбації $\epsilon$, значення цілі y, кількість ітерації $T$.;
**Output:** Adversarial $x^*$ з нормою $\|x^* - x\|_{inf} \leq \epsilon$;
$a = \epsilon/T$;
$g0 = 0$; $x_0^* = x$;
**for** $t = 0 \quad to \quad T - 1$ **do**
    $g_{t+1} = \mu \cdot g_t + \frac{\Delta_x J(x^*, y)}{\|\Delta_x J(x^*, y)\|_1}$;
    $x_{t+1}^* = x_t^* + \alpha \cdot sign(g_{t+1})$;
**end for**
**return** $x^* = x_T^*$.

## 3.6 DeepFool

DeepFool була вперше представлена в роботі Moosavi-Dezfooli, Fawzi та Frossard [2]. На відмінно від FGSM методу цей алгоритм не можна використовуватись для націлених атак, де є можливість вибрати клас який має передбачити модель. Отже, його можна використовувати тільки для генерування прикладів які будуть класифікуватись відмінно від правильно класу. Основною метою цього методу є знаходження мінімального необхідного значення пертурбація, для того щоб модель хибно передбачила клас прикладу.

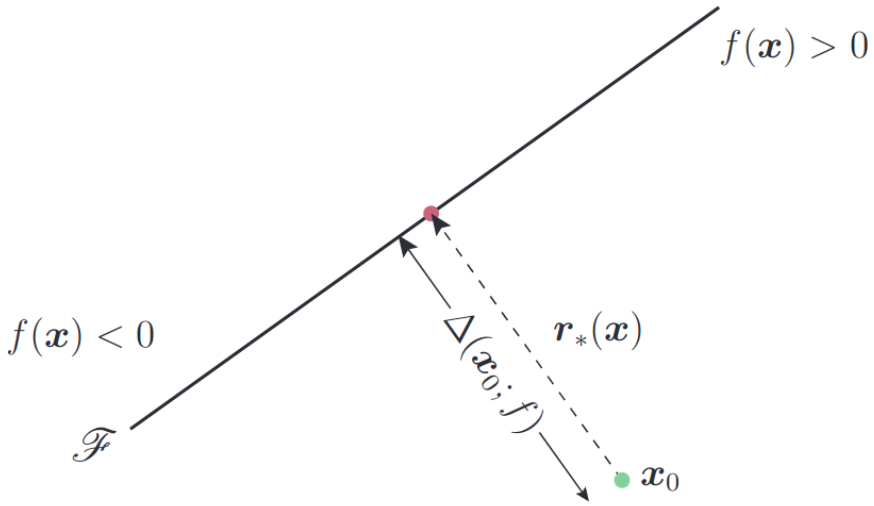


Рис. 3.1: Приклад для лінійного бінарного класифікатора [2]

Для випадку бінарної класифікації легко бачити, що надійність моделі $f$ в точці $x_0$, $\Delta(x_0; f)$, дорівнює відстанні від $x_0$ до площини гіперпараметра $\mathscr{F} = \{x : \omega^T x + b = 0\}$, яка розділяє два класи (Рис. 3.1). Таким чином, мінімальний зсув, необхідний для зміни рішення класифікатора відповідає ортогональній проекції $x_0$ на $\mathscr{F}$. Це можна

записати у вигляді формули:

$$r_*(x_0) := -\frac{f(x_0)}{\|\omega\|_2^2}\omega. \tag{3.3}$$

Для випалку $L2$ норими DeepFool алгоритм для мультикласового випадку буде мати наступний вигляд [5].

**Algorithm 5** DeepFool: мультикласовий випадок

**Input:** Приклад $x$, значення цілі y, класифікатор $f$.;
**Input:** значення цілі y, кількість ітерації $T$.;
**Output:** Adversarial $x^*$ з нормою $\|x^* - x\|_2 \leq \epsilon$;
$x_0 = x$; $i = 0$;
**while** $\hat{f}(x_i) = \hat{f}(x_0)$ **do**
    **for** $k \neq \hat{f}(x_0)$ **do**
        $w'_k = \Delta f_k(x_i) - \Delta f_{\hat{k}(x_0)}(x_i)$;
        $f'_k = \Delta f_k(x_i) - \Delta f_{\hat{k}(x_0)}(x_i)$;
    **end for**
    $\hat{l} = \arg\min_{k \neq \hat{f}(x_0)} \frac{|f'_k|}{\|w'_k\|_2}$;
    $r_i = \frac{|f'_{\hat{l}}|}{\|w'_{\hat{l}}\|_2^2} w'_{\hat{l}}$;
    $x_{i+1} = x_i + r_i$;
    $i = i + 1$;
**end while**
**return** $\hat{r} = \sum_i r_i$.

# Розділ 4

# Аналіз атак

# Розділ 5

# Методи захисту

3.2. Defenses Due to the threat of adversarial examples, extensive research has been conducted on building robust models to defend against adversarial attacks. In this paper, we roughly classify the defense techniques into five categories, including robust training, input transformation, randomization, model ensemble, and certified defenses. Note that these defense categories are not exclusive, i.e., a defense can belong to many categories. Below we introduce each category

Robust Training: The basic principle of robust training is to make the classifier robust against small perturbations internally. One line of work is based on adversarial training [20, 56, 37, 26, 68], which augments the training data by adversarially generated examples. Another line of work trains robust models by regularizations, including those on the Lipschitz constant [12], input gradients [24, 47], or perturbation norm [66].

Input Transformation: Several defenses transform the inputs before feeding them to the classifier, including JPEG compression [18], bit-depth reduction [65], total variance minimization [21], autoencoder-based denoising [33], and projecting adversarial examples onto the data distribution through generative models [49, 52]. However, these defenses can cause shattered gradients or vanishing/exploding gradients [1], which can be evaded by adaptive attacks.

Randomization: The classifiers can be made random to mitigate adversarial effects. The randomness can be added to either the input [62] or the model [14, 35]. The randomness can also be modeled by Bayesian neural networks [36]. These methods partially rely on random gradients to prevent adversarial attacks, and can be defeated by attacks that take the expectation over the random gradients [23, 1].

Model Ensemble: An effective defense strategy in practice is to construct an ensemble of individual models [31]. Besides aggregating the output of each model in the ensemble, some different ensemble strategies have been proposed. Random self-ensemble [35] averages the predictions over random noises injected to the model, which is equivalent to ensemble an infinite number of noisy models. Pang et al. [40] propose to promote the diversity among the predictions of different models, and introduce an adaptive diversity promoting regularizer to achieve this

Certified Defenses: There are a lot of works [44, 51, 58, 59, 45, 61] on training certified defenses, which are provably guaranteed to be robust against adversarial perturbations under some threat models. Recently, certified defenses [69, 13] can apply to ImageNet [48], showing the scalability of this type of defenses.

# Розділ 6

# Висновок

# Бібліографія


[1] Ian J. Goodfellow, Jonathon Shlens та Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. 2014. arXiv: 1412.6572 [stat.ML].

[2] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi та Pascal Frossard. *DeepFool: a simple and accurate method to fool deep neural networks*. 2015. arXiv: 1511.04599 [cs.LG].

[3] Alexey Kurakin, Ian Goodfellow та Samy Bengio. *Adversarial examples in the physical world*. 2016. arXiv: 1607.02533 [cs.CV].

[4] Yinpeng Dong та ін. *Boosting Adversarial Attacks with Momentum*. 2017. arXiv: 1710.06081 [cs.LG].

[5] Xiaoyong Yuan та ін. *Adversarial Examples: Attacks and Defenses for Deep Learning*. 2017. arXiv: 1712.07107 [cs.LG].

[6] Alexey Kurakin та ін. *Adversarial Attacks and Defences Competition*. 2018. arXiv: 1804.00097 [cs.CV].

[7] Yinpeng Dong та ін. *Benchmarking Adversarial Robustness*. 2019. arXiv: 1912.11852 [cs.CV].