КАЗАХСТАНСКИЙ ФИЛИАЛ МГУ ИМЕНИ М. В. ЛОМОНОСОВА
НАПРАВЛЕНИЕ 01.03.01 «ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА»"
КАФЕДРА ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Медведев Владимир Витальевич

Исследование эффективности атаки на извлечение модели искусственной нейронной сети с функцией активации ReLU

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
доцент, к.ф.-м.н.
Чижов Иван Владимирович

Астана, 2024

Оглавление

Введение

Мотивации соперника для атаки: кража, разведка. Атаки на извлечение модели классифицируются по двум целям: точность и достоверность. Два подхода к решению задачи извлечения модели: задача машинного обучения (функциональная аппроксимация) и математический подход (прямой анализ).

В данной работе рассматривается проблема извлечения всех весов и смещентй глубокой нейронной сети, которая запрашивается как чёрный ящик для получения выходов, соответствующих тщательно выбранным входам.

Вспомогательные определения:

Определение 0.1. Модель нейронной сети это параметризированная функция $f_\theta \colon \mathcal{X} \to \mathcal{Y}$ которая отображает входные данные $x \in \mathcal{X}$ на выходные данные $y \in \mathcal{Y}$.

Определение 0.2. Две модели f и g называются (ε, δ)-функционально эквивалентными на S, если $Pr_{x \in S}\left[|f(x) - g(x)| \leqslant \varepsilon\right] \geqslant 1 - \delta$

1. Предварительная подготовка

1.1. Основные определения и обозначения

Определение 1.1. Нейронная сеть $f_\theta(x)$ глубины r это функция, параметризированная по θ, которая берёт входы из входного пространства $\mathcal{X}$ и возвращает значения из выходного пространства $\mathcal{Y}$. Функция f составляется как последовательность функций, чередующихся между линейными слоями f_j и нелинейной функцией σ:

$$f = f_{r+1} \circ \sigma \circ \cdots \circ \sigma \circ f_2 \circ \sigma \circ f_1$$

.

Мы рассматриваем глубокие нейронные сети (DNN) на вещественных числах. Тогда, $\mathcal{X} = \mathbb{R}^{d_0}$ и $\mathcal{Y} = \mathbb{R}^{d_{r+1}}$, где d_0 и d_{r+1} положительные целые. Также, в работе рассматриваются только нейронные сети с функцией активации ReLU $\sigma\colon x \mapsto max\,(x,\ 0)$.

Определение 1.2. i-ый слой нейронной сети f_i задаётся как аффинное преобразование

$$f_i(x) = A^{(i)}x + b^{(i)}.$$

Веса $A^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$ это $d_i \times d_{i-1}$-матрица; смещение $b^{(i)} \in \mathbb{R}^{d_i}$ это d_i-мерный вектор.

Определение 1.3. Нейрон − это функция, определяемая соответствующей матрицей весов и функцией активации. В частности, j-ый нейрон слоя i − это функция η, заданная следующим образом

$$\eta\,(x) = \sigma\left(A_j^{(i)}x + b_j^{(i)}\right).$$

Всего имеется $N = \sum_{j=1}^{r} d_i$ нейронов.

Определение 1.4. Пусть $l = d_{i-1}$ и $A_j^{(i)}$ описывается как $(a_1, a_2, \ldots, a_l)$. Сигнатура j-ого нейрона слоя i это кортеж

$$\left(\frac{a_1}{a_1} = 1, \frac{a_2}{a_1}, \ldots, \frac{a_l}{a_1}\right). \tag{1.1}$$

Определение 1.5. Пусть $\mathcal{V}(\eta; x)$ обозначает вход нейрона η (до применения σ) при $x \in \mathcal{X}$. Нейрон η является критическим, когда $\mathcal{V}(\eta;x) = 0$. Критической точкой η, назовём входной сигнал x, обозначаемый $x \in \mathcal{W}(\eta)$. Если $\mathcal{V}(\eta;x) > 0$ тогда нейрон активен, иначе – неактивен. Состояние η на входе x (активное, неактивное или критическое) обозначается $\mathcal{S}(\eta; x)$.

Определение 1.6. Пусть $x \in \mathcal{X}$. Линейной окрестностью x назовём множество

$$\{u \in \mathcal{X} \mid \mathcal{S}(\eta; x) = \mathcal{S}(\eta; u) \text{ для всех нейронов } \eta \text{ в сети}\}.$$

Определение 1.7. Архитектура нейронной сети описывается структурой f: (a) числом слоёв и (b) размерностью каждого слоя $\{d_i\}_{i=0}^{k+1}$.

Пусть F_i обозначает функцию, которая вычисляет первые i слоёв DNN после функций ReLU, т.е. $F_i = \sigma \circ f_i \circ \sigma \circ \cdots \circ \sigma \circ f_1$. По определению, все нейроны остаются в одном и том же состоянии при вычислении DNN с входами из линейной окрестности для $x \in \mathcal{X}$. Для каждой такой точки x' имеем

$$\begin{aligned} F_i(x') &= I^{(i)}\left(A^{(i)} \ldots \left(I^{(2)}\left(A^{(2)}\left(I^{(1)}\left(A^{(1)}x' + b^{(1)}\right)\right) + b^{(2)}\right) \cdots + b^{(i)}\right) \\ &= I^{(i)}A^{(i)} \ldots I^{(2)}A^{(2)}I^{(1)}A^{(1)}x' + \beta \\ &= \Gamma x' + \beta, \end{aligned}$$

где $I^{(j)}$ это $0-1$ диагональная матрица с 0 на k-ом элементе диагонали, когда k-ый нейрон слоя j неактивен и 1 на k-ом элементе диагонали, когда нейрон активен. То есть, в линейной окрестности входного сигнала x мы можем "свернуть"действие различных смежных слоёв в аффинной преобразование. Если мы изменяем вход на Δ, то можем наблюдать соответствующие изменения в значении нейронов:

$$F_i(x + \Delta) - F_i(x) = \Gamma(x + \Delta) + \beta - (\Gamma(x) + \beta) = \Gamma\Delta.$$

Это Δ должно быть таким, чтобы $x + \Delta$ находилось в линейной окрестности x.

Предположим, что нам полностью известны первые $i-1$ слоев, и в данный момент мы восстанавливаем слой i. Пусть F_{i-1} и G_{i+1} представляют собой, соответственно, полностью восстановленную и невостановленную части DNN, т.е

$$f = \underbrace{f_{k+1} \circ \sigma \circ \cdots \circ \sigma \circ f_{i+1}}_{G_{i+1}} \circ \sigma \circ f_i \circ \underbrace{f_{i-1} \circ \sigma \circ \cdots \circ \sigma \circ f_1}_{F_{i-1}}.$$

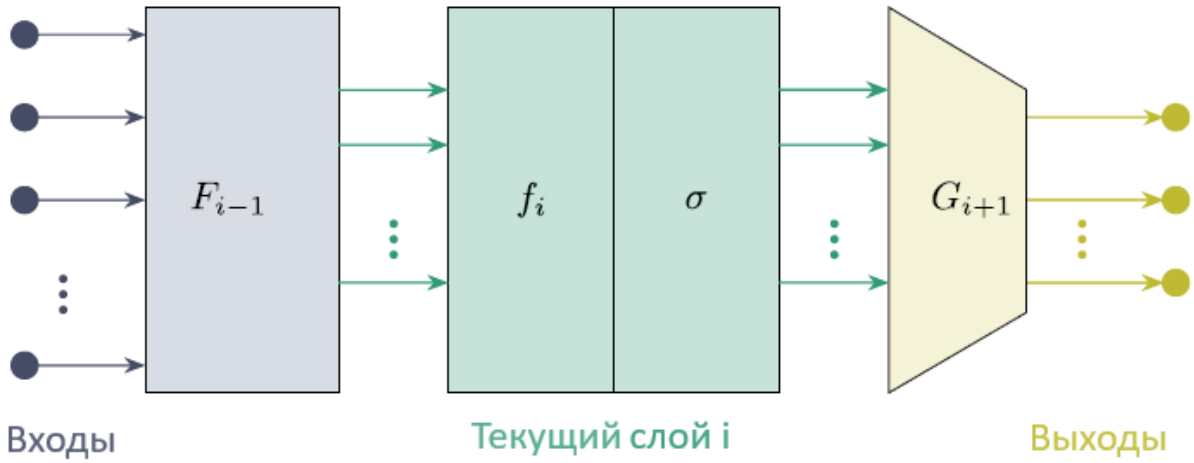

Рис. 1. Представление DNN в соответствии с восстановленной частью F_{i-1}, текущим целевым слоем i и неизвестной частью G_{i+1}.

Тогда, нейронную сеть можно изобразить так, как показано на рисунке **??**. Более того, если мы ограничим входы x' линейной окрестностью x, мы можем свернуть F_{i-1} и G_{i+1} как

$$F_{i-1}(x') = F_x^{(i-1)}x' + b_x^{(i-1)} \qquad \text{и} \qquad G_{i+1}(x') = G_x^{(i+1)}x' + b_x^{(i+1)},$$

соответственно. Подстрочный индекс x в свёрнутых матрицах и векторах смещения означает, что они определнены в линейной окрестности x.

Определение 1.8. j-ая строка $G_x^{(i+1)}$ есть выходные коэффициенты j-ого выхода DNN.

1.2. Постановка задачи и допущения

Определение 1.9. Атака с извлечением параметров модели получает оракульный доступ к параметризированной функции f_θ (в нашем случае нейронная сеть глубины k) и архитектуре f, и возвращает набор параметров $\widehat{\theta}$ с целью, чтобы $f_\theta(x)$ была как можно более похожа на $f_{\hat{\theta}}(x)$.

Восстановление параметров слой за слоем происходит в два этапа. На первом этапе находятся кратные параметры, в частности, сигнатуры нейронов **??**. Поскольку сигнатура состоит из соотношений пар весов, одновременное отрицание всех этих весов сохранит сигнатуру, но нелинейно изменит выходы

ReLU данного нейрона. Следовательно, прежде чем отделить слой нейронов, необходимо определить знак весов этого нейрона. На втором этапе эти знаки восстанавливаются для всех нейронов в текущем слое.

Сделаем некторые предположения относительно оракула и возможностей атакующего:

- **Знание архитектуры**. Нам требуется знать архитектуру нейронной сети.
- **Полнодоменные входы**. Мы подаём на вход произвольные сигналы из $\mathcal{X} = \mathbb{R}^{d_0}$.
- **Полные выходы**. Мы получаем выходные данные непосредственно от модели f без дополнительной оработки.
- **Точные вычисления**.Для задания и оценки f используется 64-разрядная арифметика с плавающей запятой.
- **Скалярные выходы**. Без потери общности считаем, что размерность выхода равнна 1, т.е. $\mathcal{Y} = \mathbb{R}$.
- **Полносвязная сеть и активация ReLU**. Сеть я вляется полностью соединённой, и все функции активации (σ) являются функциями ReLU.
- **Доступность и уникальность подписи**. Мы имеем доступ к сигнатуре каждого нейрона. Кроме того, мы предполагаем, что нет двух одинаковых сигнатур.

2. Обзор дифференциальной атаки.

Имея оракульный доступ к функции f_θ, мы можем оценить ∂f_θ с помощью конечных разностей по произвольным направлениям. Для простых линейных функций, определяемых как $f(x) = a \cdot x + b$, производная по направлению будет равна $\frac{\partial f}{\partial e_i} \equiv a_i$, где e_i - базисный вектор, a_i - это i-й элемент вектора a, что позволяет восстановить его веса путём запроса по этим хорошо подобранных входам.

В случае с нейронными сетями мы рассматриваем вторую частную производную по направлению. Нейронные сети ReLU это кусочно-линейные функции с $\frac{\partial^2 f}{\partial x^2} \equiv 0$ почти всюду.

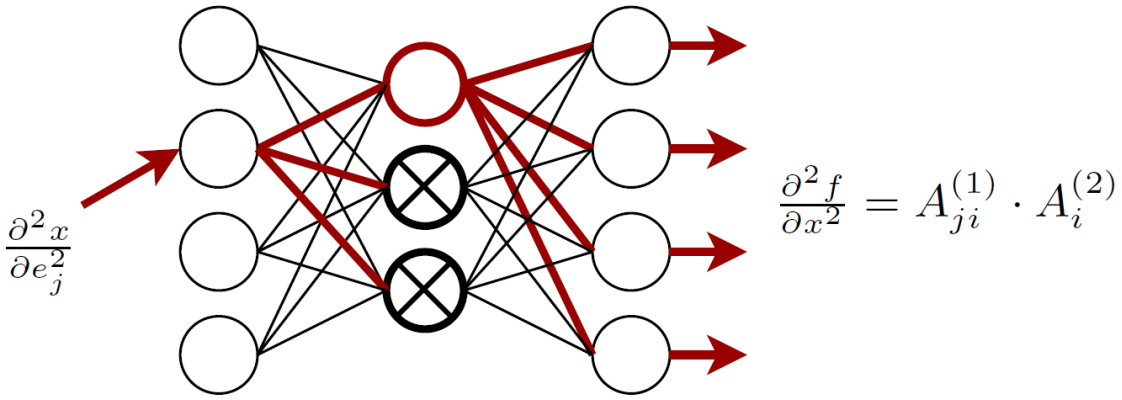

Рис. 2. Схема атаки на нейронную сеть глубиной 1.

Нахождение критических точек. Для нахождения критических точек анализируется функция, вызываемая DNN, когда входной сигнал x_1 линейно преобразуется в другой входной сигнал x_2. Пусть $x_1, x_2 \in \mathbb{R}^{d_0}$ и функция $\mu\colon [0,1] \to \mathbb{R}^{d_0}$, определенная как

$$\mu\colon \lambda \mapsto x_1 + \lambda\,(x_2 - x_1)$$

есть линейное преобразование x_1 в x_2. Это приводит к функции выхода DNN,

$$f^*(\lambda) := f(\mu(\lambda)),$$

которая является кусочно-линейной функцией с разрывами первого порядка именно тогда, когда один и знейронов переключается между активным/неактивным состояниями.

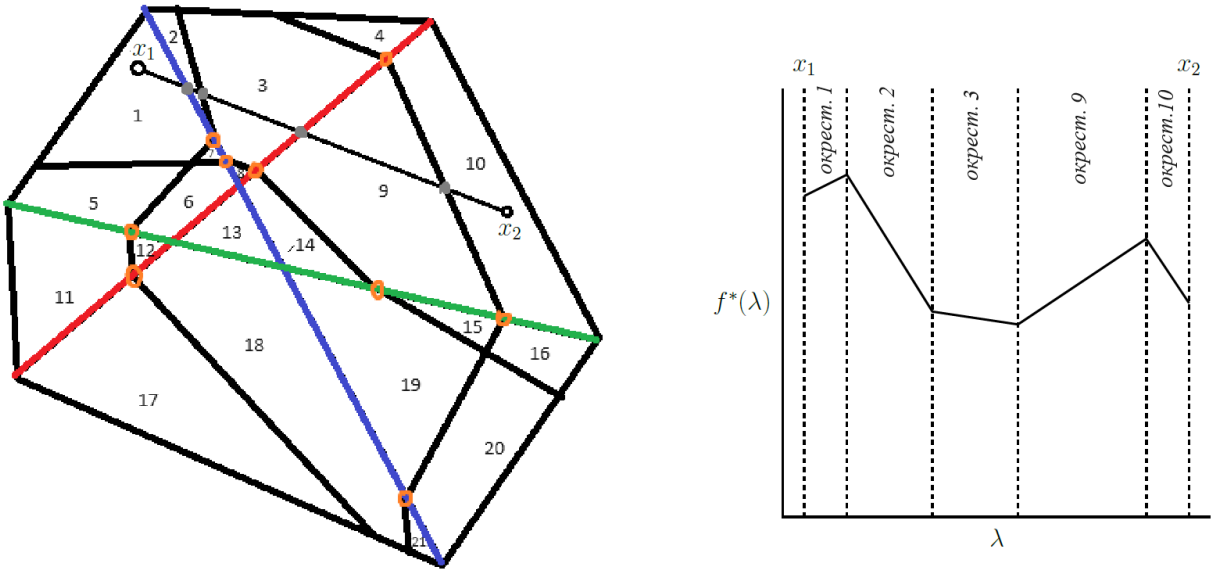

Рис. 3. Входное пространство может быть разбито на линейные окрестности, а выходная функция демонстрирует резкие изменения в поведении при пересечении их границ.

Как показано на рисунке **??**, мы можем определить разрывы первого пордка и вернуть отображение μ, чтобы восстановить точку, в которой линия от x_1 до x_2 пересекает границу между линейными окрестностями, что является критической точкой для некоторого нейрона.

На практике достаточно измерить наклон графика на рисунке **??** в разных точках и экстраполировать его, чтобы найти критическую точку, при этом проверяя, нет ли других резких изменений в поведени между ними.

На данном этапе мы ещё не знаем, к какому слою относится каждая критическая точка, но, выбрав достаточное количество пар x_1, x_2, в конечном итоге найдутся несколько критических точек для каждого нейрона в сети.

Поиск сигнатур. Входные данные в DNN – это также вход первого скрытого слоя, и мы имеем полный контроль над ним. Пусть η – нейрон с весами $(a_1, \ldots, a_l)$. Зная, что $x^* \in \mathbb{R}^{d_0}$ – критическая точка для η, мы можем запросить $\alpha_{i,-} = \frac{\partial f}{\partial e_i}(x^* - \varepsilon e_i)$ и $\alpha_{i,+} = \frac{\partial f}{\partial e_i}(x^* + \varepsilon e_i)$, где $\{e_i\}_{i=1}^{d_0}$ – это канонический базис $\mathbb{R}^{d_0}$, а ε – маленькое действительное число. Поскольку x^* является критической точкой, только если либо $\alpha_{i,-}$, либо $\alpha_{i,+}$ будут иметь η в активном состоянии, предполагая, что ε достаточно мал, чтобы никакой другой нейрон не переключился. Разность $\alpha_{i,-} - \alpha_{i,+}$ содержит градиентную информацию, перемещающуюся от входной координаты i через нейрон η к

выходу; градиентная информация через все другие нейроны аннулируется. Другими словами, эта разность кратна a_i, полученной от других слоёв DNN. Деление на другую координату устраняет мультипликативный фактор, т.е. $\left(\alpha_{i,-} - \alpha_{i,+}\right) / \left(\alpha_{k,-} - \alpha_{k,+}\right) = a_i / a_k$. Если мы зафиксируем $k = 1$, восстанавливается сигнатура (**??**). Обозначим через $\hat{A}_j^{(i)}$ сигнатуру j-ого нейрона слоя i, а через $\hat{A}^{(i)}$ – матрицу, чьей j-ой строкой является $\hat{A}_j^{(i)}$. Критические точки для одного и того же нейрона в целефом слое 1 будут давать одну и ту же сигнатуру, в то время как критические точки для нейронов в других слоях будут давать разные сигнатуры. Это позволяет нам узнать какие сигнатуры соответствуют нейрону 1-го слоя.

После отслаивания слоя мы также мы также можем определить, соответствует ли сигнатура нейрону в текущем целевом слое, наблюдая за повторениями. Однако, начиная со 2-го слоя, мы уже не можем полностью контролировать вход слоя, и применение вышеописанного метода невозможно (мы не не можем изменять одну координату за раз). Чтобы это преодолеть, в слое $i > 1$, мы выбираем $d_i + 1$ направлений $\delta_k \sim \mathcal{N}\left(0, \varepsilon I_{d_0}\right) \in \mathcal{X}$, и пусть $\{y_k\} = \{\partial^2 f\left(x^*\right) / \partial\delta_1 \partial\delta_k\}_{k=1}^{d_i}$ и $h_k = F_{i-1}\left(x^* + \delta_k\right)$. Сигнатура тогда получется из вектора a так, что $\langle h_k, a\rangle = y_k$. Это, однако, даёт частичные сигнатуры (поскольку ReLU на предыдущем слое устанавливает отрицательные значения в ноль). Разные критические точки для одного и того же нейрона дают разные частичные сигнатуры. Каждая частичная сигнатура даёт свой набор координат, и при достаточном количестве частичных сигнатур мы можем восстановить полную сигнатуру.

Объединение частичных решений. Для заданной критической точки x^* скрытый вектор, полученный из $f_{1..i}\left(x^*\right)$, скорее всего, будет иметь несколько (в среднем, половину) отрицательных нейронов, и, следовательно, $F_i\left(x^*\right)$ и любой $F_i\left(x^* + \delta_j\right)$ будут иметь нейроны, тождественно равные нулю. Это делает невозможным восстановление полного весового вектора с помощью одного применения метода наименьших квадратов – можно вычислить веса только для тех входов, которые не являются нулевыми. Одним из решений может быть поиск критической точки x^*, такой, что по компонентам $f_{1..i}\left(x^*\right) \geqslant 0$; однако в общем случае это невозможно.

Вместо этого объединим несколько попыток извлечения весов при помощи процедуры объединения. Если x_1 и x_2 являются критическими точками для одного и того же нейрона, а частичный вектор $f_{1..i}\left(x_1\right)$ имеет входы

$t_1 \subset \{1, \ldots, d_i\}$ и частичный вектор $f_{1..i}(x_2)$ имеет записи $t_2 \subset \{1, \ldots, d_i\}$, то можно восстановить соотношения для всех входов $t_1 \bigcup t_2$ путём объединения двух частичных решений, если $t_1 \bigcap t_2$ непустое, как описано ниже.

Пусть r_1 обозначает извлечённый весовой вектор на критической точке x_1 с входами в локациях $t_1 \subset \{1, \ldots, d_i\}$ (соответственно, r_1 в x_2 с локациями в t_2). Поскольку два вектора соответствуют решениям для одной и той же строки матрицы весов $A_j^{(i)}$, вектора r_1 и r_2 должны быть согласованы на $t_1 \bigcap t_2$. Поэтому мы будем иметь, что $r_1 |t_1 \bigcap t_2| = c \cdot r_2 |t_1 \bigcap t_2|$ для скаляра $c \neq 0$. Пока $t_1 \bigcap t_2 \neq$ мы можем вычислить соответствующую константу c и затем восстановить весовой вектор $r_{1,2}$ со входами на позициях $t_1 \bigcup t_2$.

Обратим внимание, что эта процедура также позволяет *проверить*, являются ли x_1 и x_2 критическими точками нейрона η. Если $|t_1 \bigcap t_2| \geqslant \gamma$, то до тех пор, пока не существует двух строк $A_{(i)}$, в которых $\gamma + 1$ входа скалярно кратны друг другу, будет существовать единственное решение, объединяющее два частных решения вместе. Если процедура объединения проваливается − потому что не существует ни одного скаляра c, чтобы $r_1 |t_1 \bigcap t_2| = c \cdot r_2 |t_1 \bigcap t_2|$ − тогда x_1 и x_2 не являются критическими точками одного и того же нейрона.

Восстановление знаков с помощью метода замораживания. Рассмотрим теперь задачу нахождения знаков текущего слоя. Если сеть не расширяющаяся, эта задача для первого скрытого слоя не представляет сложности. Действительно, для целевого нейрона k в слое i, мы можем найти колебание Δ_k во вхоном пространстве, которое производит колебание $\pm e_k$ в первом скрытом слое, где $e_k \in \mathbb{R}^{d_i}$ это базисный вектор в k-ом направлении, путём решения d_1 уравнений в d_0 переменных, заданных матрицей весов первого слоя. Для любых x из входного пространства, выходы нейронной сети в x и в $x + \Delta_k$ будут равны, если k-ый нейрон не активен в линейной окрестности x, и будут разными иначе. Эта простая техника восстановления знаков − "Замораживание".

Есть несколько сценариев, когда этот метод может быть применён к более глубоким слоям. Заметим, что в линейной окрестности любого входа x, отображение из d_0-мерного пространства входов в пространство значений, поступающих на слой i, является аффинным отображением, поскольку не существует ReLU, переходящих из активного состояния в неактивное или наоборот. Ранг этого отображения определяет, в скольких линейно незави-

симых направлениях мы можем слегка возмущать входы слоя i, когда мы рассматриваем произвольное возмущение входа x. Если ранг дотаточно высок, мы можем ожидать, что найдём предизображения для каждого базисного вектора и сможем применить метод "Замораживания".

В целом, однако, наши возможности по изменению входов в глубокий скрытый слой сильно ограничены, поскольку около половины нейронов в каждом слое будут подавлены, и, таким образом, ранг аффинного отображения будет уменьшаться по мере продвижения вглубь сети. Метод "Замораживания"может быть применён только в сетях, которые являются "достаточно сократимыми"(то есть размер слоя должен уменьшаться примерно в 2 раза в каждом слое, чтобы компенсировать потерю ранга).

3. Техники восстановления знаков

Уменьшение контроля над входами является существенной проблемой, которая затрудняет восстановление знаков в более глубоких слоях. Опишем эту проблему более подробно.

Пусть $x \in \mathbb{R}^{d_0}$ я вляется входом для DNN. $F_x^{(i-1)}$ и $G_x^{(i+1)}$ это свёрнутые матрицы для x, соответствующие уже восстановленной и ещё не известной части DNN, соответственно.

Определение 3.1. Пространство контроля для слоя i вокруг входа x, обозначаемое $V_x^{(i-1)}$, это диапозон линейной трансформации $F_x^{(i-1)}$. Размерностью этого пространства называется число степеней свободы для слоя i со входом x и обозначается $d_x^{(i-1)}$.

Пространство контроля это векторное пространство, содержащее все возможные малые измененния на входе в слой i, и, по определению, $d_x^{(i-1)} = \text{rank}\left(F_x^{(i-1)}\right)$. Если в слоях с 1-ого по $i-1$ ReLU делает нейроны неактивными, то для фиксированного числа x число степеней свободы останется равным или уменьшается с ростом i. Рассмотрим вход x, делающий половину от d нейронов в 1-ом слое активными. Матрица $F_x^{(1)}$ тогда будет иметь ранг $d/2$, и, в частности, строки, соответствующие неактивным нейронам, являются нулевыми векторами. Если во втором слое $d' < d/2$ активных нейронов (для того же входа x), матрица $F_x^{(2)}$ имеет ранг d', и, следовательно, число степеней свободы также равно d'. Теперь, если $d' \geqslant d/2$, то $F_x^{(2)}$ имеет ранг $d/2$ (поскольку ранг $F_x^{(1)}$ не может увеличиться при умножении на другую матрицу). Для последующих слоёв ситуация та же: ранг нельзя увеличить, если он уменьшился. По факту, число степеней свободы на слое i обычно определяется минимальным числом активных нейронов на слой, среди всех слоёв от 1 до $i-1$, но в некоторых случаях, оно может быть строго меньше.

Расмотрим DNN с входной размерностью d и одинаковой шириной d во всех скрытых слоях. Предположим далее, что каждый нейрон имеет вероятность $1/2$ быть активным. Изначально число степеней свободы равно размерности d входа DNN. Затем в первом скрытом слое будет активна в среднем половина нейронов, что уменьшит число степеней свободы на входе слоя 2 до $d/2$. Можно считать, что ReLU проецирует пространство контроля на пространство, определяемое активными нейронами. Можно подумать,

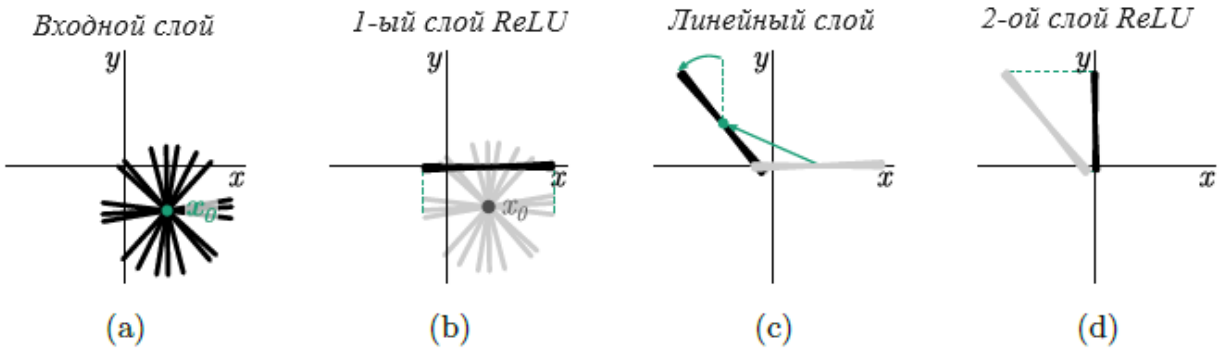

Рис. 4. Интуиция для пространства контроля. Рассмотрим сеть с двумерным входом и двумя скрытыми слоями с двумя нейронами в каждом. (a) Для первого скрытого слоя пространство контроля – это полное двумерное пространство. Мы можем перемещаться по входу x_0 в любом направлении. (b) После первого скрытого слоя, если один ReLU положительный, а другой отрицательный, мы теряем одно измерение контроля. (c) Линейное преобразование во втором слое будет вращать, перемещать и масштабировать пространство контроля. (d) Таким образом, после ReLU во стором слое(если опять же один положительный, а другой отрицательный) у нас остаётся одномерное пространство контроля. При отсутствии вращения пространство контроля сворачивается в точку после этих двух скрытых слоёв.

что половина из этих $d/2$ степеней свободы снова будет потеряна во втором слое из-за того, чо половина его ReLU будет неактивна, и в итоге получится $d/4$ степеней свободы. Однако, прежде чем перейти к ReLU на этом слое, пространство контроля для слоя 2 обычно поворачивается за счёт $A^{(2)}$. благодаря этому повороту многие координаты могут пережить проекцию ReLU на второй слой. Мы всё ещё можем потерять несколько степеней свободы, но не более половины из них на каждом последующем слое. Благодаря этому эффекту линейного преобразования число степеней свободы обычно стабилизируется после нескольких первых слоёв. На рисунке **??** это явление изображено в двумерном случае.

В контексте атаки при восстановлении знаков слоя i мы полностью знаем F_{i-1}, знаем f_i с точностью до знака на нейрон, и G_{i+1} полностью неизвестно. Для простоты записи опустим подстрочный индекс x

3.1. Восстановление знаков SOE

Рассмотрим случай, когда число степеней свободы достаточно велико. Название метод SOE получил, так как он основан на решении системы уравнений (**S**ystem **O**f **E**quations). Этот метод внешне похож на метод "Замораживания но использует другие уравнения и другой набор переменных (в случае "Замораживания"переменные обозначали направление, по которому мы должны двигаться во входжном пространстве, чтобы "заморозить"все нейроны кроме целевого, в то время как в SOE переменные обозначают коэффициенты выходной функции G_{i+1} в некоторой случайно выбранной точке x).

Как и прежде, пусть $I_x^{(i)}$ − матрица, представляющая ReLU на лое i для входа x. Уравнение для изменения выхода при произвольной последовательности изменений входа Δ_k, может быть записано как

$$f\left(x+\Delta_k\right)-f\left(x\right)=G_x^{(i+1)}I_x^{(i)}A^{(i)}F_x^{(i-1)}\Delta_k.$$

Пусть, $y_k=A^{(i)}F_x^{(i-1)}\Delta_k$ и $c=G_x^{(i+1)}I_x^{(i)}$. Если соблюдается левосторонность наблюдается в полученни значений z_k через прямые запросы, уравнение может быть переписано в виде

$$c\cdot y_k=z_k,$$

которое можно рассматривать как систему уравнений, где c − вектор переменных. В силу ReLU, если нейрон j неактивен на входе x, то $c_j=0$, поэтому получив d_i уравнений, мы можем решить систему и определить, какие нейроны неактивны в окрестности x, и, следовательно, выбрать подходящие знаки для текущего слоя.

Замечание 3.1. В контексте атаки мы можем восстановить y_k только до глобального масштабирования каждого входа (включая, возможно, перестановку знаков), но это приводит к системе уравнений, в которой решение имеет тот же набор исчезающих переменных.

Вызовы Оракула/Время. Данный метод оптимален с точки зрения количества запросов, так как для решения слоя размера d_i требуется только d_i+1 запрос (а именно, нужно запросить $f\left(x\right)$ и $f\left(x+\Delta_k\right)$ для d_i значений k).

После выполнения запросов временная сложность атаки заключается в решении системы уравнений, что можно сделать за $O\left(d_i^3\right)$ с помощью стандартных методов.

Ограничения. Каждый выбор Δ_k приводит к новому уравнени, поэтому мы пытаемся собрать больше уравнений, пока система не станет однозначно разрешимой. Однако не все полученные уавнения могут быть линейно независимыми. На самом деле, поскольку все y_k лежат в $A^{(i)}\left(V_x^{(i-1)}\right)$, мы получим достаточно линейно независимых уравнений, только если $d_x^{(i-1)} \geqslant d_i$. Вероятно, так и будет, если размер слоя будет постоянно уменьшаться в 2 раза, но есть важное исключение для первых двух скрытых слоёв. Действительно, пока сеть не расширяется, линейная карта, соответствующая первому скрытому слою может быть инвертирована. Это позволяет легко найти x, при котором все нейроны первого слоя активны, и следовательно, $d_x^{(1)} = d_1 \geqslant d_2$. Таким образом, метод легко применим для первых двух скрытых слоёв при единственном условии − отсутствии расширеи, а в последующих слоях он может быть успешен только в том случае, если каждый из них уменьшится на коэффициент, близкий к 2.

3.2. Восстановление знака колебанием нейронов

Данный метод не имеет таких жестких ограничений на архитектуру сети, как описанный выше, поскольку его производительность постепенно снижается по мере уменьшения числа степеней свободы, в то время как в SOE система уравнений резко переходит от разрешимой к неразрешимой.

Определение 3.2. Колебание на i-ом слое это вектор $\delta \in \mathbb{R}^{d_{i-1}}$ разностей значений нейронов в этом слое.

Вспомним, что восстановленная матрица весов для слоя i это $\hat{A}^{(i)} \in \mathbb{R}^{d_i \times d_{i-1}}$ и пусть её k-ая строка это $\hat{A}_k^{(i)}$. Для колебания δ в слое i, нейрон $k \in \{1, \ldots, d_i\}$ в этом слое изменяет своё значение на $e_k = \left\langle \hat{A}_k^{(i)}, \delta \right\rangle$. Рассмотрим один выход DNN, и пусть $\left(c_1, \ldots, c_{d_i}\right)$ − это соответствующий ему вектор выходных коэффициентов (т.е. соответствующий вектор-строка в $G^{(i+1)}$). Если мы "протолкнём" разность e_k через оставшиеся слои, то разность на выходе

будет

$$\sum_{k \in I} c_k e_k, \tag{3.1}$$

где I содержит индексы всех активных нейронов слоя i. Мы хотим восстановить знак нейрона j. Если этот нейрон активен, выходная разность содержит вклад e_j. Если нейрон неактивен, ReLU "блокирует" e_j, и он не вносит свой вклад. Когда $c_j e_j$ достаточно велико, мы можем определить, присутствует ли он в **??**, и использовать эту информацию для фосстановления знака нейрона. Лучше всего это получается, когда δ – это колебание, которе максимизирует изменение значения для данного нейрона, т.е. $\left\|\hat{A}^{(i)}\delta\right\|_\infty = \left|e_j\right|$. Важнейшее свойство, которое мы здесь получаем, состоит в том, что максимизация размера колебания, производимого линейным выражением, не требует знания его знака – если мы отрицаем выражение, то получаем колебание того же размера, но в противоположном направлении. Теперь покажем, как вычислить такое максимальное колебание и как восстановить знак целевого нейрона.

Вычисление колебания целевого нейрона. Пусть $\delta \in \mathbb{R}^{d_{i-1}}$ параллельно $\hat{A}_j^{(i)}$ (т.е. все координаты δ имеют тот же, либо противоположный знак к соответствующей координате $\hat{A}_j^{(i)}$). Тогда все слагаемые в скалярном произведении $\left\langle \hat{A}_j^{(i)}, \delta \right\rangle$ имеют одинаковый знак. Если ни одна строка в $\hat{A}^{(i)}$ не кратна $\hat{A}_j^{(i)}$, тогда с высокой вероятностью $\left|\left\langle \hat{A}_j^{(i)}, \delta \right\rangle\right| > \left|\left\langle \hat{A}_k^{(i)}, \delta \right\rangle\right|$, для всех $k \neq j$. Это значит, что $\left\|\hat{A}^{(i)}\delta\right\|_\infty = \left|\left\langle \hat{A}_j^{(i)}, \delta \right\rangle\right|$. Следовательно, изменение значения для нейрона j может быть максимальным, если колебание параллельно $\hat{A}_j^{(i)}$.

$V^{(i-1)}$ – это пространство контроля для слоя i при входном сигнале x. Мы проецируем $\hat{A}_j^{(i)}$ на $V^{(i-1)}$ и получаем δ, масштабируя эту проекцию так, чтобы она имела достаточно малую норму $\varepsilon^{(i-1)}$. Наконец, мы получаем входную разность $\Delta \in \mathbb{R}^{d_0}$, которая порождает δ, найдя прообраз δ под $F^{(i-1)}$.

Восстановление знака целевого нейрона. Вы хотим восстановить знак для η_j – j-ого нейрона в слое i. Пусть x^* – критическая точка для η_j и пусть $\Delta \in \mathbb{R}^{d_0}$ генерирует колубание δ (на слое i), которое максимизирует изменение значеня для этого нейрона. Предположим, что знак η_j положительный. Тогда знаки восстановленных весов $\hat{A}_j^{(i)}$ такие же как и у настоящих весов $A_j^{(i)}$. Отсюда следует, что и координаты δ имеют тот же знак, что и координаты $A_j^{(i)}$ и $e_k = \left\langle A_k^{(i)}, \delta \right\rangle$ имеет положительное значение. Поскольку x^* является критической точкой для η_j, вычисление DNN в точке $x^* + \Delta$ делает нейрон η_j

активным, таким образом

$$f(x^* + \Delta) - f(x^*) = c_j e_j + \sum_{k \in I \setminus \{j\}} c_k e_k,$$

где I – индексы всех активных нейронов слоя i. Необходимо, чтобы Δ изменял состояние только нейрона η_j. При вычислении в точке $x^* - \Delta$ колебание δ имеет противоположные знаки, чем $A_j^{(i)}$. Тогда, все разности e_k также имеют противоположные знаки(в сравнении со значениями в $x^*+\Delta$). В данном случае, η_j становится неактивным и мы имеем что

$$f(x^* - \Delta) - f(x^*) = -\sum_{k \in I \setminus \{j\}} c_k e_k.$$

Теперь предположим, что η_j имеет отрицательный знак. Тогда колебания δ будут иметь противоположные знаки, чем $A_j^{(i)}$, и, следуя аналогичному анализу, мы получим, что

$$f(x^* + \Delta) - f(x^*) = -\sum_{k \in I \setminus \{j\}} c_k e_k$$

и

$$f(x^* - \Delta) - f(x^*) = c_j e_j + \sum_{k \in I \setminus \{j\}} c_k e_k.$$

Таким образом, чтобы определить знак, нам нужно определить, вносит ли $c_j e_j$ вклад в выходную разность с $x^* - \Delta$ или $x^* + \Delta$.

Пусть $L = f(x^* - \Delta) - f(x^*)$ и $R = f(x^* + \Delta) - f(x^*)$ обозначают, соответственно, выходную разность слева и справа для x^*. Мы решаем, что $c_j e_j$ появляется слева, т.е. знак нейрона равен -1, если $|L| > |R|$. Иначе, мы решаем, что знак равен $+1$. Так как $c_j e_j \neq 0$, то невозможно, что $|L| = |R|$.

Если $c_j e_j$ и $\sum_k c_k e_k$ имеют одинаковый знак, тогда $\left|c_j e_j + \sum_k c_k e_k\right| > \left|-\sum_k c_k e_k\right|$ всегда выполняется, и решение о знаке всегда верно. Если $c_j e_j$ и $\sum_k c_k e_k$ имеют противоположные знаки, это может привести к неправельному решению. Это неправильное решение случается, когда $\left|-\sum_k c_k e_k\right| > \left|c_j e_j + \sum_k c_k e_k\right|$. Тогда для правильного решения необходимо, чтобы $\left|c_j e_j\right| > 2\left|\sum_k c_k e_k\right|$.

Вспомним, что заданный вход DNN определяет частичную матрицу $G^{(i+1)}$. То есть, разные входы определяют разные коэффициенты на выходе DNN. Этот факт называется *рандомизацией выхода* и используется он для преодоления проблемы принятия неверного решения о знаке. Мы находим

s различных критических точек для нейрона η_j; каждая точка определяет различные выходные коэффициенты. Мы ожидаем, что большинство этих точек определяют такие коэффициенты, что $c_j e_j$ и $\sum_k c_k e_k$ выполняют условия для принятия правильного решения. Для каждой критической точки, мы вычисляем колебание δ, соответствующее входной разности Δ, и делаем выбор знака. Пусть s_- и s_+ обозначают, соответственно, число критических точек, для которых знак был выбран -1 и $+1$. Также, пусть *уровень доверия* α для -1 равен s_-/s и s_+/s для $+1$. Затем принимаем решение о знаке -1, если $s_- > s_+$, и уровень доверия больше, чем порог α_0. Если $s_+ > s_-$ и уровень доверия больше, чем α_0, то принимаем решение о $+1$. Иначе, решение о знаке не делается. В последнем случае, мы можем попробовать воостановить знак при помощи дополнительных критических точек.

С увеличением числа нейронов техника станет ещё лучше, поскольку в более высоких измерениях векторы, как правило, более ортогональны друг другу, и, следовательно, соотношение между размерами колебанй в целевом нейроне и в других нейронах должно увеличиться.

До сих пор мы предполагали, что у DNN один выход. Когда у неё несколько выходов, мы можем использовать евклидову норму над вектором выходов для сравнения L и R, что выгодно для данного метода. Это связано с тем, что каждая критическая точка рандомизирует коэффициенты нескольких выходов, и вероятность того, что несколько выходов будут иметь одновременно "плохие"коэффициенты, ниже.

Вызов оракула/Время. Чтобы восстановить знак одного нейрона для s критических точек, мы вычисляем Δ и сравниваем L с R. Вычислене Δ не требует запросов к оракулу: для этого достаточно выполнить операции линейной алгебры, чтобы найти критическую точку x^*, спроецировать $A_j^{(i)}$ на $V^{(i)}$ и найти Δ. В частности, умножения и инверсии матриц. Размер задействованных матриц определяется количеством входов d_0 и числом нейронов d_i. Таким образом, временная сложность составляет $O\left(d^3\right)$, где $d = \max(d_0, d_i)$. Сравнение L с R требует запросов к оракулу на §*, §$^* - \Delta$ и §$^* + \Delta$. Вычисление $L = \|f\left(x^* - \Delta\right) - f\left(x^*\right)\|$ и $R = \|f\left(x^* + \Delta\right) - f\left(x^*\right)\|$ требует $O(d_{r+1})$ операций, где d_{r+1} это число выходов DNN. Таким образом, для одного нейрона требуется $3s$ запросов и $\left(sd^3\right)$ операций. В общей сложности нам потребуется $3sd_i$ запросов и $O\left(sd_i d^3\right)$ операций для восстановления знака всех нейронов в слое i.

Ограничения. Мы можем рассматривать $c_j e_j$ как сигнал, а $\sum_k c_k e_k$ как шум. Мы видели, что когда знаки сигнала и шума различаются, мы можем ошибочно определить знак нейрона. Это происходит, когда сигнал недостаточно велик по сравнению с шумом. В частности, если количество нейронов в слое i слишком велико по сравнению со степенями свободы (для конкретного входа x), сигнал может быть очень слабым по отношению к шум. Это означает, что данная техника может не работать с DNN с большим расширением по сравнению с наименьшим скрытым слоем или количеством входов.

Кроме того, этот метод может не подойти для DNN с небольшим количеством нейронов в целевом слое. Вероятность того, что два случайных вектора будут перпендикулярны, уменьшается в низкоразмерных пространствах. Поэтому колебание для целевого нейрона может привести к ощутимо большим изменениям и для других нейронов (с весовыми векторами, несколько параллельными весовым векторам целевого нейрона). В этой ситуации вклад других нейронов может противодействовать вкладу целевого нейрона.

Наконец, этот метод использует рандомизацию вывода. При восстановлении последнего скрытого слоя рандомизация выходов отсутствует. Если в этом слое есть нейроны с плохими (постоянными коэффициентами, то соответсвтующий им знак будет восстановлен неверно.

3.3. Восстановление знака последнего скрытого слоя

Данный метод сосредоточен на восстановлении знака нейронов в последнем скрытом слое. Выход DNN представляется как аффинное преобразование f_{r+1} без последующих вызовов ReLU. Поэтому, в слое r (последний скрытый слой), матрица $G^{(r+1)}$ одинакова для всех входов x. Это означает, что все входы определяют одинаковые выходные коэффициенты; эквивалентно, для этого слоя не существует рандомизации выхода. Мы используем этот факт, чтобы восстановить знаки нейронов. Выходные коэффициенты восстанавливаются через вторые производные, таким образом, этот метод напоминает дифференциальный криптоанализ второго порядка.

Пусть $c_1, \ldots, c_r$ – выходные коэффициенты. Также, пусть x – входной сигнал DNN и $y^{(i)}$ есть выход i-ого слоя после ReLU, т.е., $y^{(i)} = F_i(x)$. Выходной сигнал DNN определяется как

$$f(x) = c_1 y_1^{(r)} + \cdots + c_{d_r} y_{d_r}^{(r)} + b^{(r+1)}, \tag{3.2}$$

где $y_k^{(r)}$ – k-ая координата $y^{(r)}$ и $b^{(r+1)}$ – смещение выходного слоя.

С помощью восстановленной матрицы $\hat{A}^{(r)}$, мы можем вычислять значение, которое примет k-ый нейрон в слое r перед ReLU, то есть $e_k = \left\langle \hat{A}^{(r)}, y^{(r-1)} \right\rangle$, но мы не знаем его знак s_k. Рассмотрим оба варианта. Только один e_k или $-e_k$ положительный, а другой – отрицательный; последний будет заблокирован ReLU. Таким образом, $\sigma(e_k, -e_k)$ это либо $(e_k, 0)$, либо $(0, e_k)$, в зависимости от того $e_k > 0$ или $e_k < 0$, соответственно. Мы знаем значение e_k, поэтому мы можем вычислить $\sigma(e_k, -e_k)$. Запишем $\left(\hat{y}_{k+}^{(r)}, \hat{y}_{k-}^{(r)}\right) = \sigma(e_k, -e_k)$. Теперь, нахождение s_k эквивалентно решению вещественное значение $y_k^{(r)}$ нейрона после ReLU равно $\hat{y}_{k+}^{(r)}$ или $\hat{y}_{k-}^{(r)}$. Вклад $f(x)$ в уравнении **??** это $c_k \hat{y}_{k+}^{(r)}$ когда $s_k = +1$, иначе это $c_k \hat{y}_{k-}^{(r)}$. Тогда это уравнение может быть переписано как

$$f(x) = \sum_{k=1}^{d_r} c_k \left(s_k \hat{y}_{k+}^{(r)} + (1 - s_1)\, \hat{y}_{k-}^{(r)} \right) + b^{(r+1)}.$$

Итак, мы берём случайные входы x, строит систему линейных уравнений и решаем для неизвестных s_k и $b^{(r+1)}$. Нам нужно выбрать не менее $d_r + 1$ случайных входов, чтобы система имела единственное решение.

Выходные коэффициенты c_k могут быть получены через вторую производную, которая представляет собой изменение наклона функции. Вторая производная для функции ReLU:

$$c = \frac{\sigma(y + \varepsilon) - 2\sigma(y) + \sigma(y - \varepsilon)}{\varepsilon^2}$$

Пусть x^* – критическая точка k-го нейрона слоя i и Δ – вектор малой норм в линейной окрестности x^*. Тогда,

$$f\,(x^* + \Delta) - 2f(x^*) + f\,(x^* - \Delta) = \pm \left\langle F_k^{(i)}, \Delta \right\rangle c_k,$$

где $F_k^{(i)}$ – это k-ая строка матрицы $F^{(i)}$ определяемая x^* при коллапсе слоёв с 1-ого по i-ый. Знак выше "+" когда $x + \Delta$ активирует нейрон и "−" иначе. Если Δ параллелен $F_k^{(i)}$, $\left\langle F_k^{(i)}, \Delta \right\rangle > 0$ и нейрон активирован с $x + \Delta$. Тогда, разделив полученное выше выражение на $\left\langle F_k^{(i)}, \Delta \right\rangle$ получим коэффициент. При восстановлении знака мы не имеем доступ к настоящей $F^{(i)}$, но мы знаем $\hat{F}^{(i)} = \hat{A}^{(i)} F^{(i-1)}$ Чтобы получить коэффициент k-ого нейрона мы выбираем Δ как $\hat{F}_k^{(i)}$ масштабированную так, чтобы норма была достаточно малой. Получение выходного коэффициента нейрона k может быть выполненно для

любого скрытого слоя. ТОлько в последнем из них коэффициенты остоются постоянными для разных точек x^*.

Вызовы Оракула/Время. Сначала для каждого нейрона мы находим критическую точку и вычисляем выходной коэффициент. Это требует 3-ёх запросов к оракулу и операций линейной алгебры со сложностью по времени $O\left(d^3\right)$, где $d = max\left(d_0, d_r\right)$. То есть для всех d_r нейронов требуется $3d_r$ запросов и временная сложность $O\left(d_r d^3\right)$. Тогда, построение системы уравнений также требует $O\left(d^3\right)$ операций (чтобы вычислить значения $\hat{y}_{k-}^{(r)}$ и $\hat{y}_{k+}^{(r)}$ для $d_r + 1$ точки) и $d_r + 1$ запросов. Наконец, решение системы уравнений требует $O_r\left(d^3\right)$ операций и не требует запросов.Всего нам требуется $4d_r + 1$ запросов, а временная сложность составляет $O\left(d_r d^3\right)$ операций.

Ограничения. Эта техника требует, чтобы выходные коэффициенты были постоянными. Это происходит только в последнем скрытом слое. Поэтому данный метод применим только к этому слою.

4. Заключение

Новые методы восстановления знака работают за полиномиальное количество запросов и полиномиальное количество времени. Это существенно лучше предыдущего метода "Замораживания" который работал за полиномиальное количество запросов, но за полиномиальное количество времени.

5. Список литературы

1) Isaac A. Canales-Martínez and Jorge Chavez-Saab and Anna Hambitzer and Francisco Rodríguez-Henríquez and Nitin Satpute and Adi Shamir, Polynomial Time Cryptanalytic Extraction of Neural Network Models, Cryptology ePrint Archive, Paper 2023/1526

2) Matthew Jagielski and Nicholas Carlini and David Berthelot and Alex Kurakin and Nicolas Papernot, High Accuracy and High Fidelity Extraction of Neural Networks, 2020

3) Nicholas Carlini and Matthew Jagielski and Ilya Mironov, Cryptanalytic Extraction of Neural Network Models, 2020