ДОКЛАДЫ РОССИЙСКОЙ АКАДЕМИИ НАУК. МАТЕМАТИКА, ИНФОРМАТИКА, ПРОЦЕССЫ УПРАВЛЕНИЯ, 2023, том 514, № 2, с. 80–90

УДК 004.8

НОВЫЙ ВЫЧИСЛИТЕЛЬНО ПРОСТОЙ МЕТОД ДЛЯ РЕАЛИЗАЦИИ НЕЙРОННЫХ СЕТЕЙ С ЖЕСТКИМИ ОГРАНИЧЕНИЯМИ НА ВЫХОДНЫЕ ДАННЫЕ

© 2023 г. А. В. Константинов[1,*], Л. В. Уткин[1,]**

Представлено академиком РАН А.Л. Семеновым
Поступило 09.08.2023 г.
После доработки 25.09.2023 г.
Принято к публикации 15.10.2023 г.

Предлагается новый вычислительно простой метод построения нейронных сетей, строго удовлетворяющих ограничениям на выход. Ключевая идея метода заключается в отображении скрытого вектора сети в точку, которая гарантированно находится внутри допустимого множества, определяемого набором выпуклых ограничений. Отображение реализуется дополнительным слоем нейронной сети. Предлагаемый метод обобщается на случай, когда совместные ограничения накладываются на входные и выходные вектора. В рамках предлагаемого метода также реализуется модель проецирования в ограниченное выпуклое множество. Реализованы различные типы ограничений, в том числе линейные и квадратичные ограничения, ограничения равенства и динамические ограничения, а также возможность отображения на границу выпуклого множества. Важной особенностью метода является его вычислительная простота. Сложность прямого прохода предлагаемого слоя нейронной сети с линейными и квадратичными ограничениями равна $O(nm)$ и $O(n^2m)$, соответственно, где n – количество переменных, m – число ограничений. Численные эксперименты иллюстрируют метод путем решения задач оптимизации и классификации. Программный код, реализующий метод, находится в открытом доступе.

Ключевые слова: нейронные сети, жесткие ограничения, выпуклое множество, модель проекции, задача оптимизации, классификация

DOI: 10.31857/S2686954323601094, **EDN:** XPADWQ

1. ВВЕДЕНИЕ

Нейронные сети можно рассматривать как важный и эффективный инструмент для решения различных задач машинного обучения. Ряд прикладных задач, реализация которых осуществляется с использованием нейронных сетей, требуют, чтобы выход сети был ограничен, т.е. предсказания удовлетворяли заданным ограничениям. Примерами задач, требующих ограничений на значения выхода сети, являются задачи оптимизации с ограничениями, модели, генерирующие изображения или части изображений в заданной области, задачи классификации с ограничениями на вероятности классов и т.д. Наиболее распространенный подход для учета таких ограничений состоит в том, чтобы добавить некоторые дополнительные штрафные слагаемые к функции потерь, чтобы штрафовать нарушения ограничений. Однако это не гарантирует, что ограничения будут выполняться для всех обучающих и новых примеров, так как в этом случае нарушение ограничений только штрафуется, но не устраняется. Поэтому такой подход приводит к так называемым *мягким* ограничениям [1]. Другой подход заключается в модификации нейронной сети так, чтобы она строго предсказывала в пределах ограниченного выходного пространства. В этом случае ограничения являются *жесткими* в том смысле, что они выполняются для любого входного примера во время обучения и использования [2].

Хотя многие прикладные задачи требуют жестких ограничений, в настоящее время существует не так много моделей, которые их реализуют. Большинство моделей основано на применении мягких ограничений из-за их простой реализации [3]. Метод, учитывающий жесткие конические ограничения вида $Ax \le 0$, был предложен в работе [2]. Соответствующая модель “помещает” предсказания в допустимую область, используя

[1]*Высшая школа технологий искусственного интеллекта, Санкт-Петербургский политехнический университет Петра Великого, Санкт-Петербург, Россия*
*****E-mail: andrue.konst@gmail.com,*
******E-mail: lev.utkin@gmail.com*

определенный набор лучей. Серьезным ограничением этого метода является необходимость поиска соответствующих лучей. Если применять этот подход не только к коническим ограничениям, то нужно искать все вершины множества. Однако число вершин может быть чрезвычайно большим. Общий подход к решению задач оптимизации с ограничениями представлен в работе [4]. Он направлен на включение (потенциально невыпуклых) ограничений равенств и неравенств в алгоритмы оптимизации на основе глубокого обучения. Его производительность в значительной степени зависит от процесса обучения и выбранной архитектуры модели. Архитектура нейронной сети с жесткими ограничениями на выходные данные при помощи дополнительного выходного слоя предлагается в [5]. Однако метод рассматривает только линейные ограничения, но главное – это то, что для его реализации необходимо знать все вершины многогранника ограничений, число которых может быть огромным.

Интересный подход к решению задач квадратичной оптимизации с линейными ограничениями был предложен в [6]. Аналогичный подход, который встраивает слой оптимизации в нейронную сеть, предлагается в [7]. В отличие от [6], этот подход расширен на случай произвольной выпуклой функции потерь, включая ее параметры. Согласно [7], операция проекции выходной точки на ограниченную область может быть реализована с использованием дифференцируемого слоя оптимизации, который гарантирует, что выход нейронной сети удовлетворяет ограничениям. Однако эти подходы требуют решения задач выпуклой оптимизации для каждого прямого прохода нейронной сети, что значительно усложняет реализацию подходов. Еще один метод решения задачи оптимизации с линейными ограничениями представлен в [8]. Следует отметить, что метод может потребовать значительных вычислительных ресурсов и времени для решения сложных оптимизационных задач. Более того, он решает задачи оптимизации только с линейными ограничениями. Несколько подходов к решению задач условной оптимизации были предложены в работах [9–11]. Анализ этих подходов можно найти в обзорных статьях [12, 13].

Насколько нам известно, на данный момент нет подхода, позволяющего реализовывать и обучать нейронные сети, выход которых удовлетворяет линейным и квадратичным ограничениям, без решения задачи оптимизации при прямом проходе нейронной сети. Поэтому мы представляем новый вычислительно простой метод, который накладывает жесткие линейные и квадратичные ограничения на выходные значения нейронной сети. Основная идея метода состоит в том, чтобы отобразить вектор латентных параметров нейронной сети в точку, которая гарантированно находится внутри допустимого множества, определяемого набором ограничений. Отображение реализуется специальным дополнительным слоем нейронной сети. Предлагаемый метод просто обобщается на случай, когда ограничения накладываются не только на выходные данные сети, но и на входных данных. Еще одна особенность подхода заключается в том, что метод проекции точки на допустимое множество просто реализуется в рамках предложенного подхода. Важной особенностью предлагаемого метода является его вычислительная простота. Например, вычислительная сложность прямого прохода слоя нейронной сети, реализующего метод, в случае линейных ограничений составляет $O(nm)$, а в случае квадратичных ограничений – $O(n^2m)$, где n – количество переменных, m – количество ограничений.

Предлагаемый метод может применяться в различных прикладных задачах. Прежде всего, это решение задач оптимизации с произвольными дифференцируемыми функциями потерь и с линейными или квадратичными ограничениями. Метод может применяться для реализации генеративных моделей с ограничениями. Его можно использовать, когда ограничения накладываются на определенные подмножества точек. Есть много других прикладных задач, где входные и выходные данные нейронных сетей должны быть ограничены.

Вклад работы можно резюмировать следующим образом:

1. Предложен новый вычислительно простой метод, который накладывает жесткие линейные и квадратичные ограничения на выходные значения нейронной сети.

2. Рассмотрена реализация метода с различными типами ограничений, в том числе линейными и квадратичными ограничениями, ограничениями равенствами, ограничениями, накладываемыми на входные и выходные данные.

3. Исследуются различные модификации предложенного метода, в том числе модель получения решений на границах допустимого множества и проекционные модели.

4. Приведены численные эксперименты, иллюстрирующие предложенный метод. В частности, рассматриваются различные задачи оптимизации и задачи классификации.

Программное обеспечение, реализующее метод может быть найдено на сайте: https://github.com/andruekonst/ConstraiNet/.

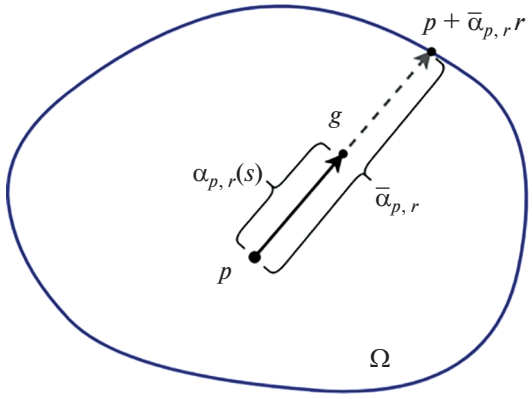

Рис. 1. Схема отображения $g_p(r,s)$.

2. ПОСТАНОВКА ЗАДАЧИ И ПРЕДЛАГАЕМЫЙ МЕТОД

Пусть $z \in \mathbb{R}^d$ – входной вектор нейронной сети, а $x \in \mathbb{R}^n$ – выходной вектор (предсказание) сети. Нейронная сеть рассматривается как функция $f_\theta : \mathbb{R}^d \to \mathbb{R}^n$ такая, что $x = f_\theta(z)$, где $\theta \in \Theta$ – вектор обучаемых параметров.

Пусть $\Omega \subset \mathbb{R}^n$ – выпуклое допустимое множество выходных данных, образованное множеством ограничений в виде m неравенств:

$$\Omega = \{x \,|\, h_i(x) \le 0,\ i = 1,\ldots,m\}, \qquad (1)$$

где каждое ограничение $h_i(x) \le 0$ является выпуклым.

Наша цель – построить нейронную сеть с ограничениями на выходные данные. Другими словами, необходимо построить модель $x = f_\theta(z) : \mathbb{R}^d \to \Omega$ и наложить ограничения на x так, что $x \in \Omega$ для любого $z \in \mathbb{R}^d$, т.е.

$$\forall z \in \mathbb{R}^d \,(f_\theta(z) \in \Omega).$$

2.1. Слой нейронной сети с ограничениями для выходных данных

Пусть задана фиксированная точка p внутри выпуклой области Ω, т.е. $p \in \Omega$. Любая точка x из множества Ω может быть представлена в виде:

$$x = p + \alpha \cdot r,$$

где $\alpha \ge 0$ – параметр масштаба; $r \in \mathbb{R}^n$ – вектор (луч из точки p).

С другой стороны, для любых p, r, существует верхняя граница $\bar{\alpha}_{p,r}$ параметра α, определяемая как

$$\bar{\alpha}_{p,r} = \max\{\alpha \ge 0 \,|\, p + \alpha \cdot r \in \Omega\}.$$

Причем отрезок $[p;\, p + \bar{\alpha}_{p,r} \cdot r]$ принадлежит Ω, так как Ω – выпуклое множество. Смысл определения границы $\bar{\alpha}_{p,r}$ в том, чтобы определить точку пересечения луча r с одним из ограничений.

Зададим слой нейронной сети, отображающий луч r и масштаб s в виде:

$$g_p(r,s) = p + \alpha_{p,r}(s) \cdot r,$$

где $\alpha_{p,r}(s)$ – функция параметра слоя s и $\bar{\alpha}_{p,r}$, определяемая как

$$\alpha_{p,r}(s) = \sigma(s) \cdot \bar{\alpha}_{p,r},$$

$\sigma(s) : \mathbb{R} \to [0,1]$ – сигмоидальная функция, т.е. гладкая монотонная функция.

Такой слой гарантированно выполняет ограничения:

$$\forall r \in \mathbb{R}^n, \quad s \in \mathbb{R}\,(g_p(r,s) \in \Omega),$$

так как отрезок $[p,\, p + \bar{\alpha}_{p,r} \cdot r]$ принадлежит Ω.

Схема отображения луча r и скаляра s в точку внутри области Ω представлена на рис. 1. По лучу r, выходящему из точки p, находится пересечение с границей области $p + \bar{\alpha}_{p,r} \cdot r$, и затем в результате масштабирования получается точка g.

Для системы ограничений достаточно находить верхнюю границу $\bar{\alpha}_{p,r}$, удовлетворяющую каждому из ограничений. Пусть $\bar{\alpha}^{(i)}_{p,r}$ – верхняя граница параметра α, соответствующая i-му ограничению $(h_i(x) \le 0)$ системы (1). Тогда верхняя граница всей системы задается так, чтобы $[p, p + \bar{\alpha}_{p,r} \cdot r] \subseteq$ $\subseteq [p, p + \bar{\alpha}^{(i)}_{p,r} \cdot r]$, т.е.

$$\bar{\alpha}_{p,r} = \min\{\bar{\alpha}^{(i)}_{p,r}\}_{i=1}^m. \qquad (2)$$

Схема поиска верхней границы $\bar{\alpha}_{p,r}$ при линейных ограничениях изображена на рис. 2.

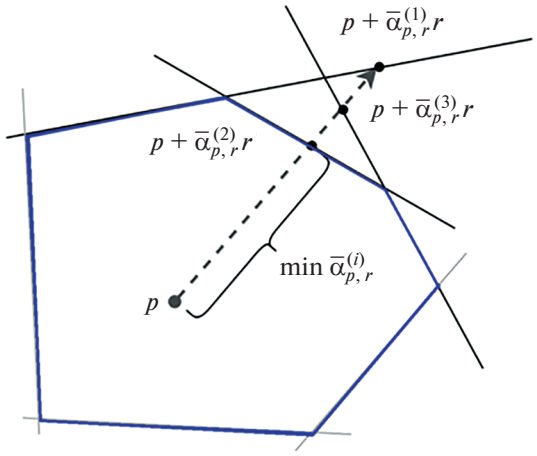

Рис. 2. Схема поиска верхней границы $\bar{\alpha}_{p,r}$ пересечения ограничений.

Вычислительная сложность прямого прохода описанного слоя нейронной сети прямо пропорциональна числу ограничений и вычислительной сложности пересечения с одним ограничением.

Утверждение 1. *С помощью слоя $g_p(r,s)$ может быть представлен любой вектор $x \in \Omega$. Для любого входа (r,s) выход слоя $g_p(r,s)$ принадлежит множеству Ω.*

Доказательство:

1. Любой выходной вектор $g_p(r,s)$ удовлетворяет ограничениям, т.е. $\forall r \in \mathbb{R}^n$, $s \in \mathbb{R}$ выполняется условие $g_p(r,s) \in \Omega$, так как $\alpha_{p,r}(s) \le \bar{\alpha}^{(i)}_{p,r}$, и $[p, p + \bar{\alpha}^{(i)}_{p,r} \cdot r] \subset \Omega$. Следовательно, $g_p(r, s) \in [p, p + \alpha_{p,r}(s)] \subset \Omega$.

2. Любая точка $x \in \Omega$ может быть представлена с помощью слоя $g_p(r,s)$. Действительно, пусть $r = x - p$, $s \to +\infty$. Тогда $x = g_p(x-p, +\infty) = p + 1 \cdot (x - p) = x$, что требовалось доказать.

Чтобы получить модель $f_\theta(z) : \mathbb{R}^d \to \Omega$, требуется подать на вход слоя $g_p(r,s)$ выход основной нейронной сети $r_\theta(z)$ и $s_\theta(z)$:

$$f_\theta(z) = g(r_\theta(z), s_\theta(z)).$$

Такая совокупная модель также образует нейронную сеть, которую можно обучать алгоритмом обратного распространения ошибки.

2.2. Линейные ограничения

В случае линейных ограничений $\bar{\alpha}_{p,r}$ определяется пересечением луча из точки p в направлении r с набором ограничений. Рассмотрим пересечение с одним линейным ограничением вида $a_i^T x \le b_i$. Верхняя граница параметра α определяется решением системы:

$$\begin{cases} x = p + \alpha \cdot r, \\ a_i^T x = b_i, \\ \alpha \ge 0. \end{cases} \tag{3}$$

Следовательно, если решение существует, то верно:

$$a_i^T p + \alpha \cdot a_i^T r = b_i,$$

$$\bar{\alpha}^{(i)}_{p,r} = \frac{b_i - a_i^T p}{a_i^T r}.$$

Если $a_i^T r = 0$ или $\bar{\alpha}_i(p,r) < 0$, то (3) не имеет решения и $\bar{\alpha}^{(i)}_{p,r}$ можно положить $+\infty$. Если задана система неравенств, то верхняя граница определяется с помощью (2).

В случае линейных ограничений вычислительная сложность прямого прохода слоя нейронной сети равна $O(nm)$.

2.3. Квадратичные ограничения

Пусть i-е квадратичное ограничение задано в виде:

$$\frac{1}{2} x^T P^{(i)} x + q_i^T x \le b_i,$$

где матрица $P^{(i)}$ – положительно полуопределенная. Тогда пересечение луча с ограничением задается уравнением:

$$\frac{1}{2}(p + \alpha \cdot r)^T P^{(i)} (p + \alpha \cdot r) + q_i^T (p + \alpha \cdot r) = b_i.$$

В зависимости от коэффициента при α^2, возможны два случая:

1. Если $r^T P^{(i)} r = 0$, то уравнение линейное и имеет решение:

$$\alpha = -\frac{q_i^T p + \frac{1}{2} p^T P^{(i)} p - b_i}{p^T P^{(i)} r + q_i^T r}.$$

2. Если $r^T P^{(i)} r > 0$, то решений два, но рассматриваем только большее положительное, соответствующее движению по направлению луча. Таким решением является:

$$\alpha = -\frac{-(p^T P^{(i)} r + q_i^T r) + \sqrt{D/4}}{r^T P^{(i)} r},$$

$$\frac{D}{4} = (p^T P^{(i)} r + q_i^T r)^2 -$$

$$- (r^T P^{(i)} r) \cdot (2 q_i^T p + p^T P^{(i)} p - 2 b_i),$$

так как знаменатель положительный.

Случай $r^T P^{(i)} r < 0$ невозможен, так как матрица положительно полуопределенная. В противном случае ограничение задавало бы невыпуклое множество.

Если $\alpha \geq 0$, то $\bar{\alpha}_{p,r}^{(i)} = \alpha$. В противном случае, если луч не пересекает ограничение, то $\bar{\alpha}_{p,r}^{(i)} = +\infty$. Для системы квадратичных ограничений верхняя граница имеет вид (2).

В случае квадратичных ограничений вычислительная сложность прямого прохода слоя нейронной сети равна $O(n^2 m)$.

2.4. Ограничения типа равенств

Рассмотрим случай, когда область Ω задается системой линейных равенств и неравенств вида:

$$x \in \Omega \Leftrightarrow \begin{cases} Ax \leq b, \\ Qx = p. \end{cases} \quad (4)$$

В этом случае задача может быть сведена к (1), т.е. к системе неравенств. Для этого найдем и зафиксируем вектор u, удовлетворяющий системе $Qu = p$. Если система не имеет решений, то Ω пустая. Если существует только одно решение, то Ω состоит из одной точки. В противном случае решений бесконечное множество и достаточно выбрать любое из них, например, решая задачу наименьших квадратов:

$$\|Qu - p\|^2 \to \min.$$

Затем можно найти матрицу R, которая является базисом ядра Q, т.е. R удовлетворяет условию:

$$\forall w (QRw = 0).$$

Матрица R может быть получена при помощи SVD-разложения матрицы $Q \in \mathbb{R}^{\mu \times n}$:

$$USV = Q,$$

где $U \in \mathbb{R}^{\mu \times \mu}$ и $V \in \mathbb{R}^{n \times n}$ – ортогональные матрицы, $S \in \mathbb{R}^{\mu \times n}$ – матрица с ненулевыми значениями только на диагонали, отсортированными по убыванию.

Тогда матрица R задается как

$$R = (v_1, \ldots, v_\delta),$$

где δ – число ненулевых диагональных элементов матрицы S, $v_1, \ldots, v_\delta$ – столбцы матрицы V.

Отсюда

$$\forall w \in \mathbb{R}^\delta (Q(Rw + u) = p).$$

Новая система ограничений на вектор w определяется как

$$A(Rw + u) \leq b,$$

или в канонической форме

$$Bw \leq t, \quad (5)$$

где $B = AR$, $t = b - Au$.

Таким образом, w – вектор переменных для новой системы (4). Для любого w, вектор x, удовлетворяющий исходной системе, может быть восстановлен в виде $x = Rw + u$. В итоге модель решения будет задаваться как:

$$f_\theta(z) = R\tilde{f}_\theta(z) + u, \quad (6)$$

где $\tilde{f}_\theta(z)$ – модель для ограничений (5).

В более общем случае, если задано произвольное выпуклое множество Ω, как пересечение выпуклых ограничений типа неравенств (1), и одно дополнительное ограничение равенство:

$$x \in \Omega \Leftrightarrow \begin{cases} h_i(x) \leq 0, \\ Qx = p, \end{cases}$$

то можно применить замену переменных для получения новых (возможно нелинейных) ограничений:

$$x \in \Omega \Leftrightarrow \begin{cases} h_i(Rw + u) \leq 0, \\ x = Rw + u, \end{cases} \Leftrightarrow \begin{cases} \tilde{h}_i^{(R,u)}(w) \leq 0, \\ x = Rw + u. \end{cases}$$

Таким образом, модель может использоваться для генерации решений $\tilde{f}_\theta(z)$, удовлетворяющих нелинейным ограничениям $\tilde{h}_i^{(R,u)}(\tilde{f}_\theta(z))$, и затем решения для x получаются через (6).

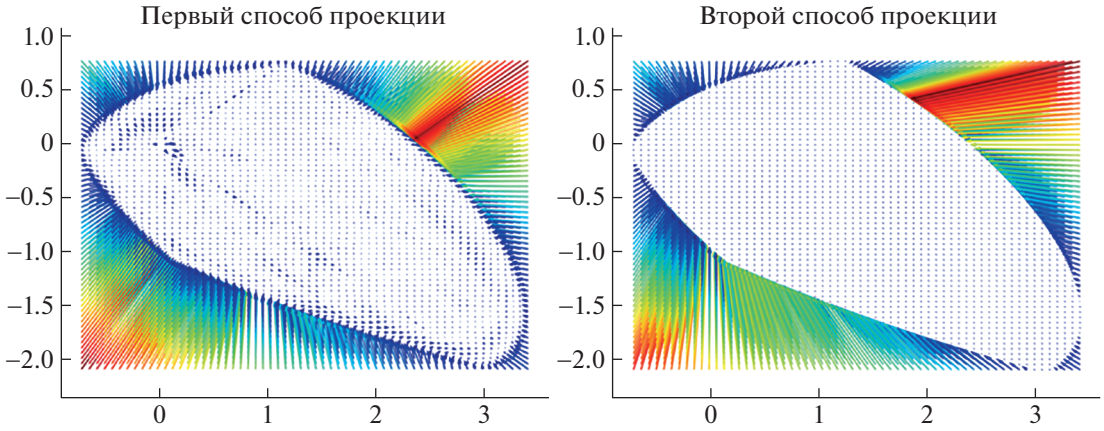

Рис. 3. Примеры моделей проекции для квадратичных ограничений.

2.5. Ограничения на входные и выходные данные

На практике может потребоваться задавать ограничения не только на выходной вектор $f_\theta(z)$, но и совместные ограничения, зависящие от некоторых входных векторов. Пусть задана выпуклая область ограничений на входной вектор z и выходной вектор $f_\theta(z)$:

$$\Lambda \subset \mathbb{R}^k \times \mathbb{R}^n,$$

т.е. для любого z, модель $f_\theta(z)$ должна удовлетворять условию:

$$y = \begin{bmatrix} f_\theta(z) \\ z \end{bmatrix} \in \Lambda.$$

Здесь y – конкатенация векторов $f_\theta(z)$ и z. Если область задана в виде пересечения выпуклых ограничений:

$$\Lambda = \{y | \Gamma_i(y) \le 0,\ i = 1,\ldots,m\},$$

то для фиксированного z путем подстановки может быть построена новая система ограничений исключительно на выходной вектор $f_\theta(z)$:

$$G(z) = \{z | \gamma_i(x;z) \le 0,\ i = 1,\ldots,m\},$$

где $\gamma_i(x;z)$ получены путем подстановки z в Γ_i.

Здесь γ_i зависит от z как от фиксированных параметров, а переменной является только x. Например, если Γ_i – линейная функция, то после подстановки параметров $\gamma_i(x;z) \le 0$ будет новым линейным ограничением на x, либо автоматически выполняться. Если Γ_i – квадратичная функция, то ограничение на x будет либо квадратичным, либо линейным, либо автоматически выполняться.

Новые *динамические* ограничения на выходные данные, зависящие от входного вектора z, удовлетворяют условию

$$f_\theta(z) \in G(z),$$

при условии, что входной вектор z из допустимого множества:

$$z \in \left\{ z | \exists x : \begin{bmatrix} x \\ z \end{bmatrix} \in \Lambda \right\}.$$

Такие ограничения могут изменяться с изменением z, и могут быть реализованы в нейронной сети при условии существования фиксированной точки p.

2.6. Модель проекции

Заметим, что с помощью предложенного слоя нейронной сети с ограничениями на выходные данные может быть построена модель *проекции*, отображающая точки в Ω и обладающая свойством идемпотентности, т.е.

$$\forall x \in \mathbb{R}^n \left(f_\theta(f_\theta(x)) = f_\theta(x) \right).$$

Такая модель может быть реализована двумя способами:

1. Первый способ – обучение модели $f_\theta(z) = g(r_\theta(z), s_\theta(s))$ тождественному преобразованию с помощью минимизации функционала, штрафующего расстояние между образом и прообразом, например, для ортогональной проекции в L_p-норме

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \| f_\theta(x_i) - x_i \|_p. \qquad (7)$$

Данный способ может найти применение при необходимости построения проецирующих моделей для сложных метрик, например, задаваемых нейронными сетями.

2. Второй способ – *центральная проекция*, которая может быть получена без оптимизации модели, за счет задания луча $r_\theta(z) \coloneqq z - p$. В этом

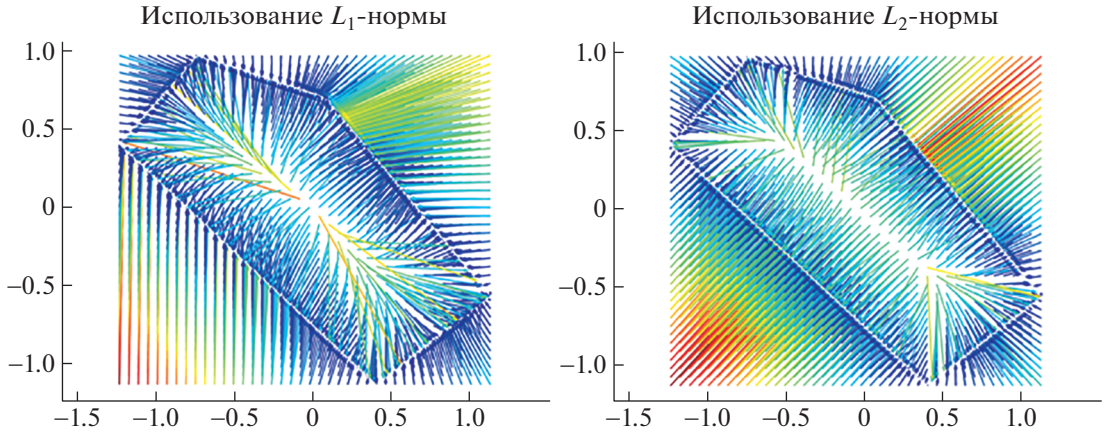

Рис. 4. Примеры аппроксимации проекции на границу с использованием L_1- и L_2-norm.

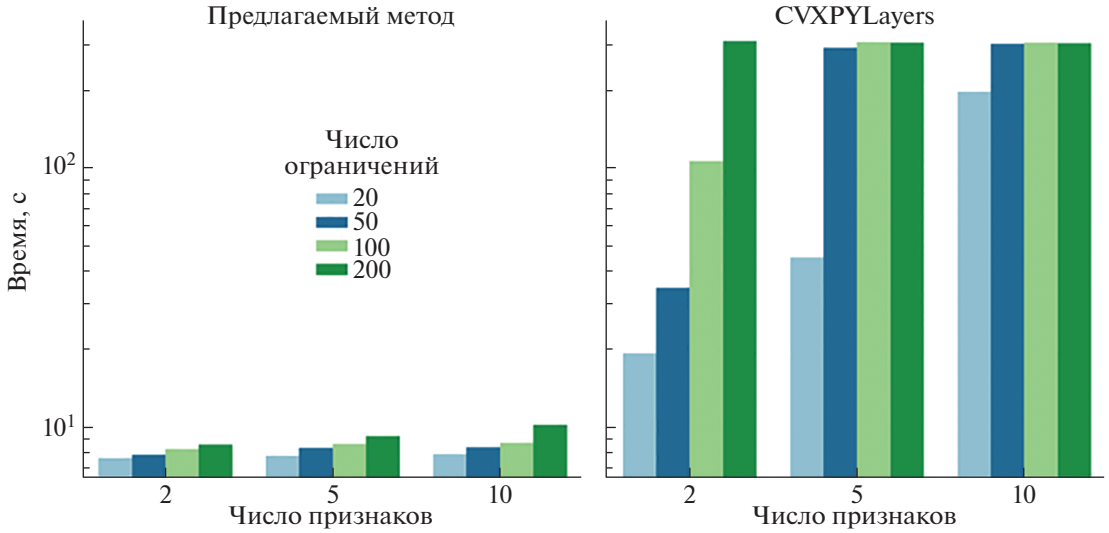

Рис. 5. Сравнение времени работы предлагаемой нейронной сети и проецирования с помощью CVXPYLayers.

случае масштаб должен быть задан без сигмоиды в явном виде, как: $\alpha_{p,r}(s) = \min\{1, \bar{\alpha}_{p,r}\}$. Тогда

$$\tilde{g}_p(r(x)) = p + \min\{1, \bar{\alpha}_{p,(x-p)}\} \cdot (x - p) = \\ = \begin{cases} x, & \bar{\alpha}_{p,(x-p)} \geq 1, \\ (1 - \bar{\alpha}_{p,(x-p)})\, p + \bar{\alpha}_{p,(x-p)} x, & \text{иначе.} \end{cases}$$

Примеры реализации модели проекции для квадратичных ограничений приведены на рис. 3. Для каждого из вариантов приведено векторное поле, где начало каждой стрелки соответствует прообразу, а конец – его проекции в множество из 5 ограничений. Слева приведен результат нейронной сети, параметры которой минимизируют (7), где в Ω имеются артефакты, соответствующие областям с большими ошибками аппроксимации. Справа приведен результат нейронной сети без обучаемых параметров, реализующей центральную проекцию, где нет ошибок.

2.7. Построение решений на границе

Разработанный метод может быть также применен для построения решений в невыпуклом граничном множестве, обозначенном $\partial\Omega$. Для этого положим $\sigma(s) = 1$. Тогда

$$g(r) = p + \bar{\alpha}(p, r) \cdot r \in \partial\Omega.$$

Отсюда следует, что $g(r)$ находится на границе при $r \neq 0$.

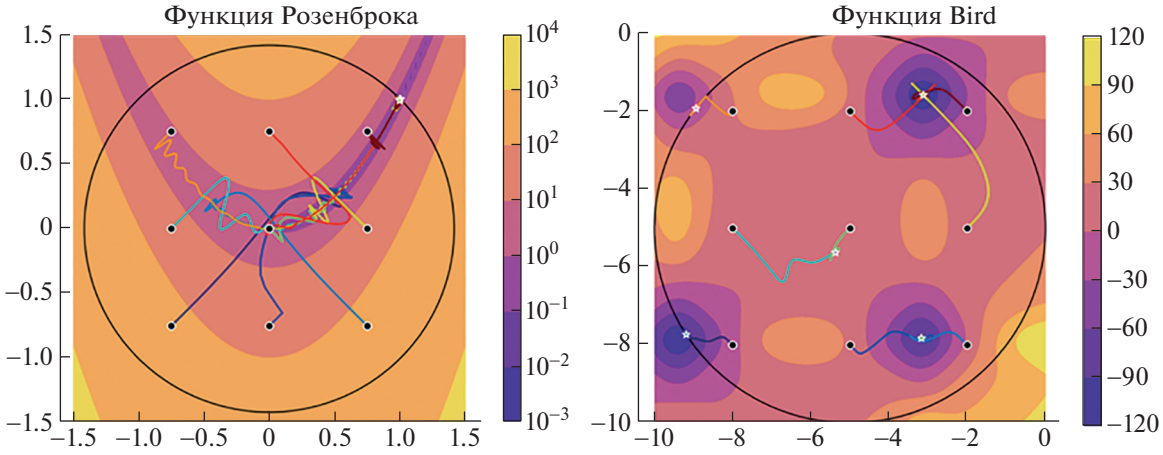

Рис. 6. Траектории оптимизации для функций Розенброка [15] и Bird [16].

Примечательно, что такой подход позволяет строить отображение на невыпуклое связное объединение выпуклых областей. С другой стороны, любой способ, опирающийся на выпуклую комбинацию базисных векторов, где веса находятся с помощью операции *softmax*, позволяет строить точки только внутри области, но не на границе. В качестве примера рассмотрим задачу проецирования точек на границу выпуклого множества:

$$\begin{gathered}\min\|z - f_\theta(z)\|_p \\ \text{при ограничениях } f_\theta(z) \in \partial\Omega.\end{gathered} \tag{8}$$

Примеры проецирования на область, заданную набором линейных ограничений, для L_1- и L_2-норм приведены на рис. 4. Для решения каждой из задач обучена нейронная сеть, содержащая 5 слоев размера 100, минимизирующая (8), множество значений которой задано как $\partial\Omega$. Из рис. 6 видно, что генерируемые точки находятся на границе ограничений.

3. ЭКСПЕРИМЕНТЫ

3.1. Задачи оптимизации

Одно из применений предлагаемого подхода – решение задач оптимизации с ограничениями. Для каждой частной задачи оптимизируется вектор входных параметров z_i так, чтобы минимизировать функцию потерь $l_i(f_\theta(z_i))$. Для тестирования были сгенерированы по 50 наборов задач и ограничений для выходной размерности 2, 5 и 10, с 50, 100 и 200 ограничениями отдельно для случаев линейных и квадратичных ограничений, так, чтобы область Ω была ограничена.

Для линейных ограничений генерировались m векторов $a_1,\ldots,a_m \sim \mathcal{N}(0, I)$, задающих точки, принадлежащие гиперплоскостям, и нормали к этим гиперплоскостям. Тогда $b_i = a_i^T a_i$, и $\Omega = \{x \,|\, a_i^T x \le b_i,\ i = 1,\ldots,m\}$. Для квадратичных ограничений генерировались положительно полу-

Таблица 1. *RE* для задач с различными функциями потерь и ограничениями

m	n	$LL{-}LC$	$LL{-}QC$	$QL{-}LC$	$QL{-}QC$
50	2	3.6×10^{-5}	1.3×10^{-4}	3.9×10^{-5}	7.1×10^{-6}
	5	2.4×10^{-3}	7.7×10^{-5}	1.2×10^{-3}	4.9×10^{-4}
	10	1.4×10^{-2}	4.3×10^{-6}	6.9×10^{-3}	3.5×10^{-3}
100	2	2.6×10^{-5}	1.6×10^{-4}	5.4×10^{-5}	6.0×10^{-6}
	5	2.1×10^{-3}	8.1×10^{-4}	1.6×10^{-3}	8.9×10^{-4}
	10	1.1×10^{-2}	1.3×10^{-5}	8.0×10^{-3}	4.3×10^{-3}
200	2	1.5×10^{-5}	3.1×10^{-4}	5.0×10^{-5}	1.2×10^{-5}
	5	2.6×10^{-3}	1.1×10^{-3}	1.3×10^{-3}	5.9×10^{-4}
	10	1.2×10^{-2}	2.8×10^{-4}	7.7×10^{-3}	4.9×10^{-3}

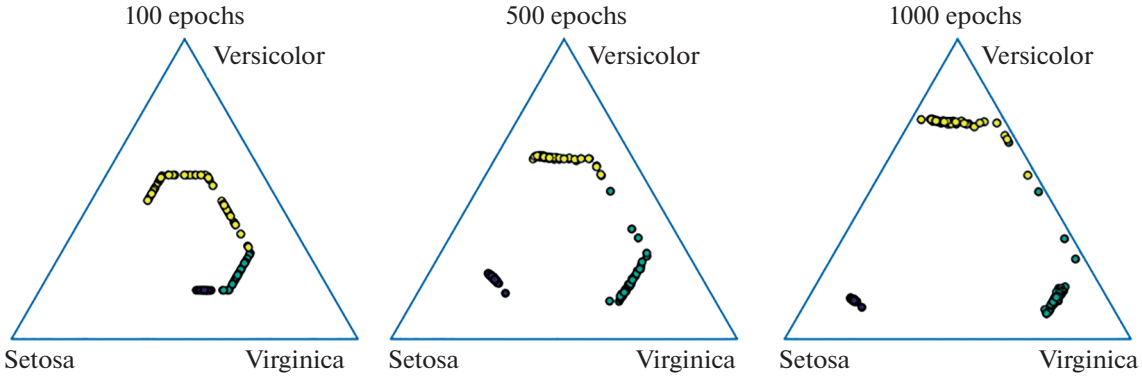

Рис. 7. Вероятности классов при различном числе (100, 500, 1000) эпох обучения.

определенные матрицы $P^{(i)}$ и векторы $q_i \sim \mathcal{N}(0, I)$, $i = 1,\ldots,m$. Затем ограничения смещались так, чтобы удовлетворять ограничениям с зазором $b_i > 0$. В итоге получаем систему квадратичных ограничений $\Omega = \{x \,|\, x^T P^{(i)} x + q_i^T x \le b_i,\ i = 1,\ldots,m\}$.

Относительная ошибка для сравнения моделей вычисляется по значению функции потерь полученного решения $l_i(f_\theta(x_i))$ относительно функции потерь эталонного решения $l_i(x_i^*)$:

$$RE = \frac{\max\{0, l_i(f_\theta(x_i)) - l_i(x_i^*)\}}{|l_i(x_i^*)|} \cdot 100\%.$$

Эталонные решения были получены с помощью алгоритма *OSQP* [14], предназначенного исключительно для решения линейных и квадратичных задач.

В табл. 1 приведены средние относительные ошибки для задач оптимизации с линейными (LL) и квадратичными (QL) функциями потерь, а также линейными (LC) и квадратичными (QC) ограничениями. Из таблицы видно, что метод позволяет оптимизировать входные параметры r и ss для предлагаемого слоя. Видно, что этот слой не ухудшает градиент для всей нейронной сети.

Для сравнения с нейронной сетью была выбрана библиотека CVXPYLayers [7], позволяющая задавать дифференцируемые слои оптимизации внутри нейронной сети. На рис. 5 приведено сравнение времени выполнения оптимизации при одинаковых гиперпараметрах, при условии, что после 5 мин с начала оптимизации, алгоритм CVXPYLayers останавливался, даже если оптимизация не была завершена. Как видно из рисунка, предложенный алгоритм требует значительно меньших затрат с точки зрения производительности.

Структура нейронной сети позволяет реализовать алгоритмы решения произвольных задач как выпуклой, так и невыпуклой оптимизации. На рис. 6 представлены траектории оптимизации для функции Розенброка [15] с квадратичными ограничениями:

$$\mathcal{L}_{Ros}(x) = (1 - x_1^2) + 100(x_2 - x_1^2)^2,$$

$$\Omega_{Ros} = \left\{x \,|\, x_1^2 + x_2^2 \le 2\right\},$$

и функции Bird [16], которая имеет 4 локальных минимума в Ω_{Bird}, два из которых лежат на границе Ω_{Bird}:

$$\mathcal{L}_{Bird}(x) = \sin(x_2) e^{(1-\cos(x_1))^2} + \cos(x_1) e^{(1-\sin(x_2))^2} + (x_1 - x_2)^2,$$

$$\Omega_{Bird} = \{x \,|\, (x_1 + 5)^2 + (x_2 + 5)^2 < 25\}.$$

Функция $\mathcal{L}_{Ros}(x)$ имеет глобальный минимум в точке (1, 1). Границы для Ω_{Ros} и Ω_{Bird} показаны большими черными окружностями. Использовались 2000 итераций алгоритма Adam со скоростью обучения 0.1. 9 точек на равномерной сетке от –0.75 до 0.75 выбраны как начальные точки, для которых траектории оптимизации показаны на рис. 6, где конечная точка обозначена белой звездой.

3.2. Пример классификации

Для иллюстрации возможности предлагаемого метода рассмотрена задача классификации на примере простейшего набора данных “Iris” с ограничениями на верхние границы $\overline{p}_i$ для каждой вероятности класса x_i:

$$f_\theta(z) \in \left\{x \,|\, 0 \le x_i \le \overline{p}_i, \mathbf{1}^T x = 1\right\}.$$

Эти ограничения могут играть уравновешивающую роль в обучении, уменьшая влияние уже правильно классифицированных точек на функцию потерь. Чтобы проиллюстрировать, как нейронная сеть обучается с этими ограничениями, и упростить визуализацию результатов, выбран

именно этот классический набор данных, содержащий 3 класса и 150 примеров. Три класса позволяют визуализировать результаты с помощью единичного симплекса вероятностей. В этом примере мы устанавливаем верхние границы $\overline{p}_i = 0.75$ для $i = 1, 2, 3$. На рис. 7 показаны единичные симплексы и точки, которые являются выходными данными нейронной сети, обученной на 100, 500 и 1000 эпохах. Нейронная сеть состоит из 5 слоев размером 64. Цвета точек обозначают соответствующие классы (Setosa, Versicolor, Virginica). Из рис. 7 видно, что ограничения влияют не только при очень близком расположении выходных точек к ним, но и на всем протяжении обучения сети.

4. ЗАКЛЮЧЕНИЕ

Представлен метод, который накладывает жесткие ограничения на выходные значения нейронной сети. Метод предельно прост с вычислительной точки зрения, реализуется дополнительным слоем нейронной сети и позволяет накладывать большой класс ограничений: линейные и квадратичные неравенства, линейные равенства, совместные ограничения на входы и выходы. Метод непосредственно может быть расширен на любые выпуклые ограничения. Он позволяет аппроксимировать ортогональные проекции на выпуклое множество или генерировать векторы на граничном множестве.

Недостаток метода заключается в том, что он не позволяет работать с исключительно коническими ограничениями. В дальнейшем можно модифицировать предложенный метод для решения задач с такими ограничениями. Кроме того, интересно распространить идеи предлагаемого метода на случай невыпуклых ограничений.

Важным направлением также является рассмотрение различных приложений машинного обучения, требующих учета ограничений, например нейронных сетей с учетом физики (physics-informed) [17]. Каждое приложение может рассматриваться как отдельная задача для дальнейшего изучения.

ИСТОЧНИК ФИНАНСИРОВАНИЯ

Работа выполнена при поддержке гранта РНФ № 21-11-00116.

СПИСОК ЛИТЕРАТУРЫ

1. *Marquez-Neila P., Salzmann M., Fua P.* Imposing Hard Constraints on Deep Networks: Promises and Limitations. *CVPR Workshop on Negative Results in Computer Vision*. 2017. P. 1–9.
2. *Frerix T., Niessner M., Cremers D.* Homogeneous Linear Inequality Constraints for Neural Network Activations. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020. P. 748–49.
3. *Lee J.Y., Mehta S.V., Wick M., Tristan J.-B., Carbonell J.* Gradient-Based Inference for Networks with Output Constraints. *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI-19)*. 2019. V. 33. P. 4147–54.
4. *Donti P.L., Rolnick D., Kolter J.Z.* DC3: A Learning Method for Optimization with Hard Constraints. *International Conference on Learning Representations (ICLR 2021)*. 2021. P. 1–17.
5. *Brosowsky M., Keck F., Dunkel O., Zollner M.* Sample-Specific Output Constraints for Neural Networks. *The Thirty-Fifth AAAI Conference on Artificial Intelligence (AAAI-21)*. 2021. P. 6812–21.
6. *Amos B., Kolter J.Z.* Optnet: Differentiable Optimization as a Layer in Neural Networks. *International Conference on Machine Learning*. PMLR, 2017. P. 136–145.
7. *Agrawal A., Amos B., Barratt S., Boyd S., Diamond S., Kolter J.Z.* Differentiable Convex Optimization Layers. *Advances in Neural Information Processing Systems*. 2019. V. 32. P. 1–13.
8. *Li M., Kolouri S., Mohammadi J.* Learning to Solve Optimization Problems with Hard Linear Constraints. *IEEE Access*. 2023. V. 11. P. 59995–60004.
9. *Balestriero R., LeCun Y.* Police: Provably Optimal Linear Constraint Enforcement for Deep Neural Networks. *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023. P. 1–5.
10. *Chen Y., Huang D., Zhang D., Zeng J., Wang N., Zhang H., Yan J.* Theory-Guided Hard Constraint Projection (HCP): A Knowledge-Based Data-Driven Scientific Machine Learning Method. *Journal of Computational Physics*. 2021. V. 445 (110624).
11. *Negiar G., Mahoney M.W., Krishnapriyan A.* Learning Differentiable Solvers for Systems with Hard Constraints. *The Eleventh International Conference on Learning Representations (ICLR 2023)*. 2023. P. 1–19.
12. *Kotary J., Fioretto F., Van Hentenryck P.* Learning Hard Optimization Problems: A Data Generation Perspective. *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*. 2021. V. 34. P. 24981–24992.
13. *Kotary J., Fioretto F., Van Hentenryck P., Wilder B.* End-to-End Constrained Optimization Learning: A Survey. *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)*. 2021. P. 4475–82.
14. *Stellato B., Banjac G., Goulart P., Bemporad A., Boyd S.* OSQP: An Operator Splitting Solver for Quadratic Programs. *Mathematical Programming Computation*. 2020. V. 12. № 4. P. 637–72.
15. *Rosenbrock H.H.* An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*. 1960. V. 3. № 3. P. 175–84.
16. *Mishra S.K.* Some New Test Functions for Global Optimization and Performance of Repulsive Particle Swarm Method. *Available at SSRN* 926132. 2006. P. 1–24.
17. *Raissi M., Perdikaris P., Karniadakis G.E.* Physics-Informed Neural Networks: A Deep Learning Framework for Solving Forward and Inverse Problems Involving Nonlinear Partial Differential Equations. *Journal of Computational Physics* 2019. V. 378. P. 686–707.

A NEW COMPUTATIONALLY SIMPLE APPROACH FOR IMPLEMENTING NEURAL NETWORKS WITH OUTPUT HARD CONSTRAINTS

A. V. Konstantinov[a] and L. V. Utkin[a]

[a] *Higher School of Artificial Intelligence Technologies Peter the Great St.Petersburg Polytechnic University, St. Petersburg, Russia*

Presented by Academician of the RAS A.L. Semenov

A new computationally simple method of imposing hard convex constraints on the neural network output values is proposed. The key idea behind the method is to map a vector of hidden parameters of the network to a point that is guaranteed to be inside the feasible set defined by a set of constraints. The mapping is implemented by the additional neural network layer with constraints for output. The proposed method is simply extended to the case when constraints are imposed not only on the output vectors, but also on joint constraints depending on inputs. The projection approach to imposing constraints on outputs can simply be implemented in the framework of the proposed method. It is shown how to incorporate different types of constraints into the proposed method, including linear and quadratic constraints, equality constraints, and dynamic constraints, constraints in the form of boundaries. An important feature of the method is its computational simplicity. Complexities of the forward pass of the proposed neural network layer by linear and quadratic constraints are $O(nm)$ and $O(n^2m)$, respectively, where n is the number of variables, m is the number of constraints. Numerical experiments illustrate the method by solving optimization and classification problems. The code implementing the method is publicly available.

Keywords: neural network, hard constraints, convex set, projection model, optimization problem, classification