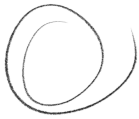
Федеральное государственное автономное
образовательное учреждение высшего образования
«Московский физико-технический институт»
(национальный исследовательский университет)



на правах рукописи

Аврутский Всеволод Игоревич

Алгоритм увеличения точности нейронных сетей и его приложения

Специальность 05.13.18 – Математическое моделирование,
численные методы и комплексы программ

АВТОРЕФЕРАТ
диссертации на соискание учёной степени
кандидата физико-математических наук

Долгопрудный – 2021

Работа прошла апробацию на кафедре компьютерного моделирования Федерального государственного автономного образовательного учреждения высшего образования «Московский физико-технический институт (национальный исследовательский университет)»

Научный руководитель: **Дорофеев Евгений Александрович**
Кандидат физико-математических наук,
доцент кафедры теоретической физики
Московского физико-технического института

Ведущая организация: Федеральное государственное
унитарное предприятие «Центральный
аэрогидродинамический институт
им. профессора Н.Е. Жуковского»

Защита состоится 30 июня 2021 г. в 14 часов на заседании диссертационного совета ФАКТ.05.13.18.004 по адресу 141701, Московская область, г. Долгопрудный, Институтский переулок, д. 9.

С диссертацией можно ознакомиться в библиотеке и на сайте Московского Физико-Технического Института (национального исследовательского университета) https://mipt.ru/education/postgraduate/soiskateli-fiziko-matematicheskie-nauki.php

Работа представлена 10 марта 2021 г. в Аттестационную комиссию федерального государственного автономного образовательного учреждения высшего образования «Московский физико-технический институт (национального исследовательского университет)» для рассмотрения советом по защите диссертаций на соискание ученой степени кандидата наук, доктора наук в соответствии с п.3.1 ст. 4 Федерального закона «О науке и государственной научно-технической политике».

Общая характеристика работы

Актуальность темы исследования. На сегодняшний день нейронные сети применяются для решения широкого спектра прикладных задач. Популярной постановкой является задача об аппроксимации функции, для которой известен набор значений и соответствующих им аргументов. На основе этих данных алгоритм обучения настраивает параметры сети так, чтобы её отклонение от значений функции на данном наборе аргументов было приемлемым. Точность аппроксимации при этом обычно оценивают с помощью отклонения от рассматриваемой функции на тестовом множестве, которое не использовалось для обучения. Достаточно широкий класс составляют сети прямого распространения (далее просто сети), принцип работы которых можно кратко описать следующим образом. Входной вектор поочерёдно подвергается многомерным линейным и покомпонентным нелинейным преобразованиям, пара которых называется слоем сети. Последний слой является выходным, а остальные – скрытыми. Линейные преобразования в слоях задаются матрицами весов, нахождение которых и составляет процесс обучения. Допустимость применения нейронных сетей обоснована теоремами о существовании, утверждающими, что для каждой непрерывной функции существует сеть с таким набором весов, что мера их отклонения друг от друга произвольно мала. Такие теоремы, однако, не являются конструктивными, и обучение сетей представляет собой отдельную задачу. В связи с этим следует отличать сети с одним скрытым слоем от глубоких сетей: если для первых существуют алгоритмы обучения, вычисляющие веса за полиномиальное время, то для глубоких сетей обучение основано на итерационном процессе градиентной минимизации, который во многих ситуациях не находит глобальный минимум.

Очередной всплеск популярности нейронных сетей произошел после 2011 года, когда с помощью глубоких сетей задачу о распознавании образов удалось решить с точностью, сравнимой с человеческой. В этой задаче каждому изображению, состоящему из тысяч цветных пикселей, нужно поставить в соответствие вектор, каждая компонента которого есть вероятность принадлежности изображения к некоторому классу. Область определения функции-классификатора имеет огромную размерность, и важным фактором оказывается способность нейронных сетей обходить «проклятье размерности». В результате, увеличение числа измерений не приводит к экспоненциальному

росту затрат на обучение. Несмотря на то, что этой способностью обладают также и сети с одним скрытым слоем, их использование в подобных многомерных задачах не приводит к столь же качественным результатам. Одним из ключевых факторов для практического применения глубоких сетей стала доступность параллельных вычислений на графических ускорителях (GPU). Эффективность GPU при работе с глубокими сетями значительно выше, чем при обучении сетей с одним скрытым слоем. Сегодня глубокие нейронные сети успешно используются для автоматизации вождения, распознавания голоса и других задачах большой размерности, ранее требовавших присутствия человека.

Нейронные сети также могут применяться для решения дифференциальных уравнений. Фактически, любой численный алгоритм должен сконструировать решение с помощью некоторого числа рациональных параметров. В методе конечных разностей искомая функция описывается с помощью набора значений на сетке. В методе конечных элементов область решения разбивается на ячейки, внутри которых определена достаточно простая функция, например полином, коэффициенты которого определяются в процессе решения. В последние годы получили развитие бессеточные методы, в которых непрерывная среда заменяется на некоторое количество взаимодействующих частиц, но и в этом случае присутствует дискретизация по пространству. При этом результирующая точность зависит от величины этой дискретизации, а значит после решения требуется проверка сходимости по данному параметру. Нейронные сети же обладают достаточными аппроксимационными способностями, чтобы описывать решение во всей области, а значит обойтись без пространственной дискретизации. При таком подходе размерность входного вектора нейронной сети равна количеству независимых переменных, а число выходов равно числу искомых функций. При обучении сети решается одна конкретная краевая задача. Веса настраиваются так, чтобы при «подстановке» сети в уравнение, невязка в тренировочных точках была близка к нулю, а значения сети на границе были близки к поставленным условиям. Сети с одним скрытым слоем здесь опять же отличаются от глубоких сетей. У первых, несмотря на отсутствие явной дискретизации, параметры являются локальными, то есть в пространстве независимых переменных можно выделить относительно небольшую область, на которую влияет

изменение каждого параметра. Для глубоких сетей это не так – вариации почти всех параметров изменяют решение глобально. Однако, для сетей с одним скрытым слоем, как и в случае задач аппроксимации, существуют полиномиальные алгоритмы поиска весов, тогда как для глубоких сетей необходима градиентная минимизация. При этом точность решения при использовании глубоких сетей почти всегда заметно ниже. Более того, зачастую при их градиентном обучении алгоритм не находит глобальный минимум, и в результате присутствует некоторый уровень относительной невязки, снизить которую далее невозможно. Тем не менее, глубокие нейронные сети позволяют решать уравнения в частных производных с размерностью порядка 100, тогда как статьи о сетях с одним скрытым слоем по большей части посвящены размерностям 1, 2 и 3. Для нейросетевого подхода характерно то, что вне зависимости от выбранного вида сетей и алгоритма обучения, единая процедура позволяет решать задачи самого разного типа: эллиптические, гиперболические, смешанные, задачи на собственные значения. Нейросетевой метод существенно отличается от классических способов решения уравнений: результатом является гладкая функция, для которой невязка уравнения в тренировочных точках мала. При этом невязка легко вычислима и в любых других точках, и коль скоро она мала и на тестовом множестве, то уравнение можно считать численно решенным, а проверка на сходимость по шагу не требуется. С помощью вычисления невязки во всей области можно так же получить оценку точности решения. После завершения обучения, точки, использовавшиеся в процессе, уже не играют никакой роли. Фактически, набор тренировочных точек можно свободно менять и в процессе обучения по мере необходимости. Кроме непосредственного вычисления искомых функций, нейросетевой подход обладает дополнительными преимуществами: так, перестройка решения после малой вариации граничных условий занимает относительно немного времени, причем для глубоких сетей она происходит быстрее в силу глобальной зависимости решения от параметров сети. Для многомерных задач хранение решения в виде глубокой нейронной сети намного более эффективно, чем описание через дискретизацию, требования по памяти для которой растут экспоненциально.

Таким образом, глубокие нейронные сети являются достаточно гибким инструментом и обладают большим потенциалом для решения

многомерных задач. Однако, из-за особенностей градиентной минимизации, их относительная точность проигрывает как сетям с одним скрытым слоем, так и стандартным численным методам при решения задач малой размерности. Для больших размерностей точность многослойных сетей не повышается, однако альтернативные методы практически исчезают. Похожая ситуация присутствует и в задачах аппроксимации – при градиентном обучении глубокой сети даже в двумерном случае легко обнаруживается неснижаемый уровень ошибки. Исправление такого недостатка позволило бы уровнять способности глубоких сетей с остальными численными методами при решении задач малой размерности, а также поднять их точность в многоразмерных задачах. Кроме того, желательно разрешить проблему скорости вычислений. Процесс решения краевых задач подразумевает «подстановку» нейронной сети в уравнение, то есть вычисление её производных и последующее вычисление градиента ошибки. На данный момент статьи о нейросетевом методе решения дифференциальных уравнений не содержат универсального алгоритма для сетей с произвольным числом слоёв. Разумеется, это вычисление можно осуществить с помощью автоматического дифференцирования, однако этот метод до сих пор не имеет эффективных GPU реализаций. Так как успех современных нейронных сетей был связан с увеличением числа слоёв, а также с наличием быстрых вычислений на GPU, то задача построения эффективного алгоритма вычисления производных нейронных сетей имеет прикладной потенциал.

Данная работа посвящена увеличению точности градиентного обучения глубоких нейронных сетей – многослойных персептронов. Под точностью подразумевается отклонение от требуемого результата на тестовом множестве. Для задачи аппроксимации разработан метод увеличения точности при условии, что на обучающем множестве известны не только значения, но и производные аппроксимируемой функции. Рассмотрены примеры аппроксимации в пространствах размерности 2–5. При решении уравнений в частных производных разработанный метод позволяет увеличить точность без использования дополнительной информации. Рассмотрены примеры решения краевых задач для линейных и нелинейных дифференциальных уравнений эллиптического и параболического типов в пространствах размерности 2–5.

Целями данной работы являются: создание способа повышения точности градиентного обучения многослойных персептронов, его применение для решения уравнений в частных производных, и его эффективная реализация на GPU.

Для достижения поставленных целей были решены следующие **задачи**:

1. Создание способа, который позволил бы преодолеть ограничение относительной точности градиентного обучения многослойных нейронных сетей $\sim 10^{-4}$, которое встречается при аппроксимации функций на областях порядка единицы, производные которых имеют значения порядка единицы.
2. Разработка метода решения уравнений на основе найденного способа повышения точности.
3. Создание программного комплекса для эффективного вычисления производных и градиентов глубоких нейронных сетей на GPU.
4. Проведение численных экспериментов, показывающих практические преимущества нейросетевого метода решения дифференциальных уравнений.

Научная новизна работы.

1. Впервые показано, что точность градиентного обучения нейронных сетей может быть увеличена на несколько порядков при включении в процесс обучения дополнительных производных.
2. Впервые продемонстрировано, что можно избежать переобучения нейронных сетей без использования регуляризации, пользуясь исключительно информацией о производных аппроксимируемой функции.
3. Впервые построен алгоритм нейросетевого решения уравнений в частных производных, позволяющий использовать шаг, невозможный для других алгоритмов при сравнимой точности решения.
4. Впервые с помощью нейронной сети решение краевой задачи с относительной точностью порядка ошибки округления было получено быстрее, чем методом конечных разностей.

Теоретическая и практическая значимость. Создание способа, с помощью которого точность градиентного обучения глубоких

нейронных сетей можно повысить на несколько порядков, представляет теоретическую значимость и указывает на необходимость исследований, направленных на достижение схожего результата в задачах классификации. Для малоразмерных задач на основе этого эффекта построен метод обучения с исключением. Его успешное применение к решению дифференциальных уравнений, получение преимуществ над методом конечных разностей, а также эффективная реализация алгоритма на GPU представляют практическую значимость.

Положения, выносимые на защиту.

1. Процедура градиентного обучения многослойных нейронных сетей с использованием дополнительных производных, свойства этой процедуры и результаты экспериментов с её применением.
2. Разработан метод обучения с исключением, позволяющий увеличить относительную точность глубоких нейронных сетей с 10^{-4} до величины порядка ошибки округления 10^{-6} при условии, что производные функции и размер области её аппроксимации имеют порядок единицы. Метод применим для численного решения уравнений математической физики, в том числе при наличии больших градиентов.
3. Разработан метод случайных направлений, снижающий сложность обучения с исключением для задач размерности более 2.
4. Модельные краевые задачи для линейного и нелинейного уравнения Пуассона в пространствах размерности 2–5 решены с помощью нейронных сетей с относительной точностью порядка 10^{-6} (производные решения и область аппроксимации имеют порядок единицы). Для пятимерной задачи нейросетевой метод впервые оказался быстрее конечно-разностного, а требования по памяти снижены с $6.4 \cdot 10^{11}$ байт до $1 \cdot 10^9$ байт. Объем памяти для хранения решения уменьшен с $6.4 \cdot 10^{11}$ до $6 \cdot 10^5$ байт.
5. Разработан программный комплекс «cpde-2», эффективно использующий GPU и предназначенный для дифференцирования многослойных персептронов, минимизации отклонения их производных, а также для решения уравнений в частных производных.

Степень достоверности и апробация результатов работы. Результаты работы докладывались на 57, 58 и 59 всероссийских научных конференциях МФТИ (2014, 2015, 2016 гг, Жуковский), международной конференции Future Technologies Conference (2019, Сан Франциско) где доклад был награжден дипломом за лучшую работу. Результаты также докладывались на семинарах:

- Seminar: Numerical Methods for Partial Differential Equations, MIT, Boston, 2017.
- Seminar: Nerual Networks Catching Up With Finite Differences, Mitsubishi Electric Research Laboratories (MERL), Boston, 2017.
- Семинар: Нейросетевые решения уравнений математической физики, ФАЛТ МФТИ, 2017 г.
- Семинар: Нейронные сети, Кафедра Вычислительной Математики, ФУПМ, 2017, 2018 гг.

Эффект «отрицательного» переобучения, упомянутый в Пункте 2.2.2 был независимо воспроизведён К.А. Рыбаковым. Исходный код доступен по адресу github.com/Avrutskiy/dneural

Публикации. Основные результаты диссертации были изложены в четырёх статьях [**1**–**4**], написанных без участия соавторов. Две из них [**1**, **2**] опубликованы в журналах первой квартили и индексируются Scopus. Работа [**3**] доложена на конференции Future Technologies Conference (San Francisco, 2019), материалы которой опубликованы в книге серии «Advances in Intelligent Systems and Computing» и индексируются в Scopus. Наконец, работа [**4**] доступна на arXiv.org.

Объём и структура работы. Диссертация состоит из введения, четырёх глав, заключения и списка литературы. Полный объем диссертации составляет 118 страниц. Список литературы содержит 129 наименований.

Благодарности. Автор выражает глубокую признательность своему научному руководителю Е. А. Дорофееву за неоценимую поддержку в течении 10 лет, на протяжении которых была написана магистерская и кандидатская диссертация. Кроме того, автор благодарит И. В. Воронича и А. М. Гайфуллина за научную поддержку, а также С. М. Боснякова за критику магистерской работы, которая подтолкнула автора к написанию данной диссертации. Автор также выражает

признательность лаборатории нейронных систем и глубокого обучения МФТИ и лично М. С. Бурцеву.

Основное содержание работы

Первая глава является обзорной. Параграф 1.1 кратко описывает устройство коры головного мозга и принцип работы биологических нейронов. В Параграфе 1.2 описаны упрощения, позволяющие перейти от живых нейронных сетей к искусственным многослойным персептронам. Согласно этой модели, входной вектор $\vec{x} \equiv (x_1, \dots x_N) \equiv x_\alpha$ поэтапно преобразуется каждым слоем нейронной сети. Первый слой умножает его на матрицу связей $W^{\beta\alpha}$ и прибавляет вектор сдвига t^β. Результатом является вектор активности нейронов первого скрытого слоя z^β:

$$z^\beta = t^\beta + \sum_\alpha W^{\beta\alpha} x_\alpha,$$

к которому далее покомпонентно применяется некоторое нелинейное преобразование σ. Затем происходит умножение на следующую матрицу связей:

$$z^\gamma = t^\gamma + \sum_\beta W^{\gamma\beta} \sigma\left(z^\beta\right). \tag{1}$$

Результатом является z^γ – вектор активности нейронов второго скрытого слоя. Если таких слоёв всего два, а выход сети v является скаляром, то можно записать

$$v = t + \sum_\gamma W^\gamma \sigma\left(z^\gamma\right). \tag{2}$$

При большем числе скрытых слоёв формулу вида (1) следует применять несколько раз. В данной работе для всех нейронов используется логистическая функция активации $\sigma(x) = 1/(1 + e^{-x})$. Отображение $v(\vec{x})$ задаётся матрицами весов $W^{\beta\alpha}$, $W^{\gamma\beta}$, W^γ и векторами сдвига t^β, t^γ, t. В Пункте 1.2.1 приводится теорема об аппроксимационных способностях такого отображения. Согласно ей, существует набор весов, которые обеспечивают произвольно малую меру отклонения сети v от произвольной непрерывной функции f на единичном кубе. Поточечное приближение этой меры в дальнейшем называется ошибкой, а её значение в точке – локальной ошибкой. Если функция f является гладкой, то нейронная сеть может приближать и её производные. В

конце параграфа указаны причины, по которым может потребоваться увеличение числа скрытых слоёв.

Несмотря на существование весов, обеспечивающих малую ошибку аппроксимации, конструктивного алгоритма их нахождения не существует. В Пункте 1.2.2 содержится вывод широко известного алгоритма обратного распространения, в котором вычисляются частные производные ошибки нейронной сети по её весам. С их помощью можно построить итерационную процедуру минимизации ошибки в виде градиентного спуска. Вывод построен по схеме, которая будет использована в Главе 4 для обобщения алгоритма на обучение любых производных многослойного персептрона. В силу линейности, рассматривается вклад в градиент только от одной точки обучающего множества. Процедура начинается с дифференцирования локальной ошибки $e = 1/2\,(v-f)^2$ по выходу нейронной сети

$$\frac{\partial e}{\partial v} = (v-f)\,.$$

С помощью замены $e\,(v) \to e\,(W^\gamma)$, формулы (2) и правила преобразования производной $\partial e/\partial W^\gamma = \partial e/\partial v \cdot \partial v/\partial W^\gamma$ получается

$$\frac{\partial e}{\partial W^\gamma} = (v-f)\,\sigma\,(z^\gamma)\,.$$

Далее вычисляется градиент e по активностям нейронов на последнем скрытом слое. Для этого рассматривается замена $e\,(v) \to e\,(z^\gamma)$:

$$\frac{\partial e}{\partial z^\gamma} = \frac{\partial e}{\partial v}\frac{\partial v}{\partial z^\gamma}.$$

С помощью этого выражения и замены $e\,(z^\gamma) \to e\left(W^{\gamma\beta}\right)$, для которой справедливо

$$\frac{\partial e}{\partial W^{\gamma\beta}} = \sum_{\gamma'} \frac{\partial e}{\partial z^{\gamma'}}\frac{\partial z^{\gamma'}}{\partial W^{\gamma\beta}},$$

вычисляется градиент весов

$$\frac{\partial e}{\partial W^{\gamma\beta}} = \frac{\partial e}{\partial z^\gamma}\sigma\left(z^\beta\right). \tag{3}$$

Далее $\partial e/\partial z^\gamma$ распространяется к слою β. Согласно правилу дифференцирования

$$\frac{\partial e}{\partial z^\beta} = \sum_{\gamma} \frac{\partial e}{\partial z^\gamma}\frac{\partial z^\gamma}{\partial z^\beta}.$$

После соответствующих упрощений получено выражение

$$\frac{\partial e}{\partial z^\beta} = \sigma'\left(z^\beta\right) \sum_\gamma \frac{\partial e}{\partial z^\gamma} W^{\gamma\beta}. \tag{4}$$

Для первой матрицы градиент есть

$$\frac{\partial e}{\partial W^{\beta\alpha}} = \frac{\partial e}{\partial z^\beta} x_\alpha.$$

Для всех векторов сдвига справедлива формула вида

$$\frac{\partial e}{\partial t^\gamma} = \frac{\partial e}{\partial z^\gamma}.$$

При большем количестве слоёв формулы вида (4) и (3) применяются несколько раз. После вычисления градиента, веса нейронной сети могут быть обновлены таким образом, что ошибка на обучающем множестве уменьшится. Суммарная ошибка в остальной области, вообще говоря, может увеличится. Такая ситуация называется «переобучением». Для проверки результатов обучения вводится понятие тестового множества как набора точек, достаточно хорошо характеризующего ошибку в остальной области. Для численного описания переобучения автор вводит коэффициент r, равный отношению ошибок на тестовом и тренировочном множестве, $r \geq 1$. В последнем абзаце обсуждается генерация начального состояния весов сети, которое обеспечивает корректную работу алгоритма обратного распространения ошибки.

Основной областью применения нейронных сетей является аппроксимация функций в пространстве большой размерности $N \sim 1000$, например, классификаторов изображений. Пункт 1.2.3 описывает отличия подобных задач от аппроксимации функций в пространстве малой размерности $N \sim 2$, к которым относится, в том числе, решение уравнений математической физики. Первое ключевое отличие состоит в том, что существует множество прикладных малоразмерных задач, для которых шум в данных отсутствует. Это приводит к важному свойству обучения – тренировочное множество можно выбрать так, чтобы величина ошибки на нём коррелировала с ошибкой сети на тестовом множестве. В результате обучение можно будет вести сколь угодно долго, тогда как в присутствии шума существует момент, при котором дальнейшее обучение только увеличивает тестовую ошибку. К сожалению, для популярных задач классификации это не так. Крайне трудно представить изображение, в котором полностью отсутствует шум: им неявно являются все посторонние объекты, а также

особенности классифицируемого предмета, напрямую не связанные с его сущностью. В результате, начиная с некоторой итерации обучения, нейронная сеть перестаёт обобщать данные и начинает «запоминать» конкретные изображения, при этом ошибка на изображениях, не вошедших в обучающую выборку, увеличивается.

Второе ключевое отличие заключается в том, что для малого количества измерений можно построить сетку с такой плотностью, что ошибка на тренировочном множестве не будет сильно отличаться от ошибки на тестовом. При этом число весов нейронной сети может быть заведомо больше, чем число обучающих точек, а ошибка на тренировочном множестве не будет обращаться в ноль. Указанные особенности малоразмерного обучения далее демонстрируются на примере аппроксимации функции

$$f(x_1, x_2) = x_1 + \sin x_2 + x_1^2 + x_2 \cos x_1^2. \tag{5}$$

Кроме того, сравниваются результаты нескольких градиентных алгоритмов. Самым быстрым оказывается RProp (Resilient backPropagation), он и будет использован в данной работе. Отмечено, что для тренировки классификаторов RProp используется редко, однако в данном случае его экспоненциально быстрая настройка шага идеально сочетается с первым ключевым отличием малоразмерных задач. В конце параграфа продемонстрирован эффект «потолка» точности: даже простую функцию, задаваемую выражением (5), сеть, имеющая 10^4 весов может приблизить лишь со средней относительной точностью $3.3 \cdot 10^{-4}$, а сеть, имеющая 10^5 весов, с точностью $3.2 \cdot 10^{-4}$. Качественно улучшить эту ситуацию не удается даже с помощью очень мелкой обучающей сетки и ещё большого числа весов. Похожие значения ошибки возникают и в большинстве статей, посвященных малоразмерным применениям глубоких нейронных сетей. Так как величина 10^{-4} сильно отстаёт от точности стандартных методов аппроксимации, поиск алгоритма для её увеличения является актуальной задачей.

В Пункте 1.2.4 описывается нейросетевой метод решения уравнений в частных производных. Впервые он был изложен М. Диссанаяке (M. Dissanayake) в 1994 году. Суть его в следующем: так как отображение $v(\vec{x})$ является гладким, то имея алгоритм вычисления производных v по компонентам $\vec{x}$, нейронную сеть можно «подставить» в любое уравнение, то есть вычислить его невязку V. Далее, имея алгоритм дифференцирования этой невязки по весам сети можно построить процедуру минимизации V^2. Для удовлетворения граничных

условий возможны два варианта. В первом случае, квадрат отклонения v от требуемого условия на границе прибавляется к невязке, и они минимизируются вместе. Такой подход более универсален, но поскольку для аппроксимации значений существует ограничение относительной точности $\sim 10^{-4}$, то и ошибка всего метода будет иметь такой-же порядок малости. И. Лагарис (I. Lagaris) в 1998 году применил другой подход, который заключается в замене функции. Пусть на границе $\partial\Gamma$ области решения Γ имеется условие

$$u\Big|_{\partial\Gamma} = f.$$

Введём новую функцию w с помощью соотношения

$$u = w \cdot \phi + \hat{f}, \tag{6}$$

где функции ϕ и $\hat{f}$ подобраны специальным образом. Функция ϕ на границе обращается в ноль, и строго больше нуля внутри Γ. Далее, $\hat{f}$ есть гладкая на Γ функция, совпадающая с f на границе. При условии гладкости всех членов, граничное условие для w сводится к тому, что значения w в процессе обучения не должны расходиться. Действительно, все решения, кроме искомого, на границе обращаются в бесконечность. Содержательная часть замены (6) состоит в том, что на практике при минимизации невязки уравнений расходимости не наблюдается. Так как замена освобождает нас от необходимости аппроксимации значений, относительная точность которой ограничена 10^{-4}, общее качество решения увеличивается. Далее метод замены и метод прямой тренировки граничных условий применяются для решения краевой задачи нелинейного уравнения Пуассона

$$\Delta u - u^2 - 3u^3/2 = 0, \tag{7}$$

$$\Gamma : x_1^2 + x_2^2 \leq 1, \tag{8}$$

$$u|_{\partial\Gamma} = -2, \tag{9}$$

точное решение которого $u_{\text{a}} = 4/\left(3 - x_1^2 - x_2^2\right)$. После 1000 шагов обучения RProp с удержанием границ, получено среднеквадратичное отклонение от точного решения $\vartheta = 2.5 \cdot 10^{-3}$ (величина нормирована на дисперсию u_{a}). Решение с помощью замены функции приводит к $\vartheta = 4.2 \cdot 10^{-5}$. Однако, проблема точности ещё далеко не решена. Дело в том, для минимизируемой величины, в данном случае – невязки, всё равно оказывается справедлива оценка 10^{-4}. Для конкретного уравнения такое значение приводит к точности решения $4.2 \cdot 10^{-5}$, однако

эту точность аналогично нельзя увеличить ни измельчением сетки, ни увеличением числа весов. В конце параграфа изображено двумерное поле невязки уравнения.

С **Главы 2** начинается изложение результатов диссертации. Параграф 2.1 описывает предпосылки и историю обнаружения способа, позволившего увеличить точность градиентного обучения нейронных сетей и преодолеть характерную относительную ошибку в 10^{-4}. Его суть заключается в одновременной минимизации не только требуемого отклонения, но и его производных. В Пункте 2.2.1 вводятся мульти-индексные обозначения: если порядок переменных зафиксирован: $(x_1, x_2 \dots)$, то производные по ним можно обозначать с помощью набора целых неотрицательных чисел $s = (s_1, s_2, \dots)$, каждое из которых есть порядок дифференцирования по соответствующей переменной

$$\partial^s \equiv \frac{\partial^{|s|}}{\partial x_1^{s_1} \partial x_2^{s_2} \dots}.$$

Здесь $|s| \equiv \sum s_i$. Мульти-индексы s можно складывать поэлементно, что соответствует произведению операторов $\partial^s \partial^q = \partial^{s+q}$. Аналогично можно ввести оператор ∂^{s-q}, если компоненты $s - q$ неотрицательны.

Пункт 2.2.2 количественно описывает эффект от минимизации отклонения производных. Рассматривается задача аппроксимации на двумерной области $\Gamma = [-1, 1]^2$ функции $f(x_1, x_2)$, заданной рядом Фурье до десятых гармоник. Для решения был выбран персептрон с 4 скрытыми слоями по 128 нейронов в каждом. При классическом обучении, локальную ошибку можно записать как

$$e_0 = \frac{1}{D^2[f]} (v - f)^2, \tag{10}$$

здесь введена нормировка ошибки на $D^2[f]$ – дисперсию f на Γ, а индекс 0 обозначает, что обучается только значение, то есть «нулевая» производная. После 1000 итераций RProp, среднее квадратичное отклонение $\sqrt{\langle e_0 \rangle}$ достигает величины $2.5 \cdot 10^{-2}$. На тестовом множестве оно равно $3.5 \cdot 10^{-2}$, то есть переобучение незначительно. С помощью соответствующего алгоритма (он получен в Главе 4), можно обучить производную персептрона:

$$e = \frac{1}{D^2[\partial^s f]} (\partial^s v - \partial^s f)^2.$$

$\|s\|$	0	1	2	3
трен	0.07%	0.18%	0.77%	1.9%
тест	0.06%	0.17%	0.61%	2.8%

Таблица 1. Зависимость точности аппроксимации производной от её порядка при минимизации e_3.

После 1000 итераций точность производной аналогично $\sqrt{\langle e \rangle} = 2.5 \cdot 10^{-2}$ на тренировочном и $3.5 \cdot 10^{-2}$ на тестовом множестве. Однако, если рассмотреть ошибку, содержащую отклонения значений и всех 9 возможных производных вплоть до третьего порядка

$$e_3 = \sum_{|s|\leq 3} \frac{1}{D^2\left[\partial^s f\right]} \left(\partial^s v - \partial^s f\right)^2, \tag{11}$$

то после 1000 итераций величина $\sqrt{\langle e_3 \rangle}$ оказывается равной $4 \cdot 10^{-2}$ ($5.6 \cdot 10^{-2}$ на тестовом множестве), что вдвое меньше, чем можно было ожидать: $\sqrt{10} \cdot 2.5 \cdot 10^{-2} \simeq 8 \cdot 10^{-2}$. Интересным выглядит распределение точности аппроксимации производных различного порядка. Оно приведено в Таблице 1, все ошибки нормированы на дисперсию соответствующих им производных, а результаты усреднены в пределах одного порядка. Ошибка значений оказывается в 35 раз меньше чем при классическом обучении. Необычным обстоятельством является более высокая точность производных второго порядка на тестовом множестве, чем на тренировочном. Этот результат фактически противоречит работе А. Пукриттаякамее (A. Pukrittayakamee), где тренировка производных, напротив, приводила к более сильному переобучению. Однако, так как суммарная минимизируемая величина на тестовом множестве оказывается больше, чем на тренировочном, противоречий с общими представлениями о переобучении нейронных сетей не возникает.

Далее исследуется как точность аппроксимации значений зависит от максимального порядка обучаемых производных. Оказалось, что по сравнению с e_0, использование e_1, e_2 и e_3 повышает точность аппроксимации значений в 4, 39 и 58 раз соответственно, однако для e_4 и e_5 эффект ослабевает до 13 а затем до 9. Такой результат объяснён малостью слагаемого $(v - f)^2 / D^2\left[f\right]$ относительно остальных членов выражения вида (11).

k	0	1	2	3	4	5
$\sqrt{\langle e_0 \rangle}$	$2.3 \cdot 10^{-3}$	$2 \cdot 10^{-4}$	$3 \cdot 10^{-5}$	$1 \cdot 10^{-5}$	$5.2 \cdot 10^{-6}$	$2.2 \cdot 10^{-6}$

Таблица 2. Точность двумерной аппроксимации в зависимости от порядка обучения с исключением. Результаты усреднены по 25 попыткам обучения.

В Подпункте 2.2.2.1 формулируется процедура обучения с исключением, позволяющая эффективно использовать информацию о старших производных. Она состоит в следующем. Вначале выбирается максимальный порядок k и начинается обучение с ошибкой e_k. После T шагов оно завершается, и начинается обучение с e_{k-1}. Процесс продолжается аналогичным образом и заканчивается минимизацией e_0. Число T для всех этапов обучения одинаково, тренировочное множество не изменяется. Процедура целиком названа «обучение с исключением порядка k». Далее приведены условия корректного сравнения обучения с исключением разных порядков. Во-первых, суммарное количество арифметических операций для разных k должно быть одинаковым. Во-вторых, должно быть равно количество «информации» о целевой функции. С учетом обоих условий получены результаты, приведённые в Таблице 2. Отмечено, что эффект увеличения точности в значительной степени зависит от количества весов сети. Например, если число нейронов во всех скрытых слоях уменьшить со 128 до 64, а затем до 32, то $\sqrt{\langle e_0 \rangle}$ для $k = 5$ увеличится с $2.2 \cdot 10^{-6}$ до $2.4 \cdot 10^{-5}$, а затем до $2.4 \cdot 10^{-4}$. Для классического обучения ($k = 0$) величина $\sqrt{\langle e_0 \rangle}$ при таком уменьшении практически не изменяется: $2.3 \cdot 10^{-3}$, $3.0 \cdot 10^{-3}$ и $5.3 \cdot 10^{-3}$ соответственно. Таким образом, была обнаружена дополнительная мотивация для увеличения числа нейронов.

В Пункте 2.2.3 обучение с исключением применено для автокодировщика. Результаты схожи со случаем двумерной аппроксимации. В Пункте 2.2.4 рассматриваются общие проблемы, связанные с вычислением производных от сложных отображений, в частности классификаторов. Указываются возможные пути их решения. Отмечено, что для задачи классификации, получение даже малой доли наблюдаемого эффекта окажется существенным достижением.

В Параграфе 2.3 замечается, что разработанный алгоритм не требует никакой дополнительной информации в случае решения уравнений в частных производных. Действительно, так как в данном случае минимизируется невязка V, которая почти всегда имеет простой вид, то все необходимые величины могут быть получены её прямым дифференцированием. Например, для получения выражения e_1 классическая ошибка $e_0 = V^2$ может быть дополнена квадратами первых производных V

$$e_1 = V^2 + (\partial V/\partial x_1)^2 + (\partial V/\partial x_2)^2 + \dots \tag{12}$$

Абсолютно аналогичные выражения могут быть записаны и для высших порядков. Далее описываются последствия дифференцирования V: повышение минимального класса гладкости решения, изменение ландшафта невязки (который должен разрешаться сеткой). Указывается на необходимость нормировки слагаемых и на увеличение числа арифметических операций, приходящихся на одну точку сетки.

Далее в Пункте 2.3.1 методом обучения с исключением решается краевая задача для нелинейного уравнения Пуассона (7)-(9). Для исключения граничных условий используется замена вида (6). Поскольку для дифференцирования невязки априорной информации о производных не требуется, для сравнения различных порядков обучения с исключением достаточно уравновесить общее количество арифметических операций. Его грубо можно считать пропорциональным общему числу производных сети, использованных для вычисления выражений e_k. Более точный расчет приведён в Главе 4. В каждом случае шаг λ выбран так, чтобы $r < 1.2$, иначе качество решений ухудшалось. Результаты обучения с исключением приведены в Таблице 3, где M – общее число точек сетки. С помощью обучения с исключением ошибка решения была уменьшена в 13 раз при сохранении общего числа арифметических операций. Отмечено, что среднеквадратичное отклонение от точного решения для $k = 3$ близко к минимальному, определяемому округлением $\vartheta_{32\text{бит}} \sim 0.36 \cdot 10^{-6}$. Из-за использования GPU, которые эффективно работают только в режиме 32 бит, для точности всех дальнейших результатов ставится требование $\sim 10^{-6}$. Последующее увеличение требовало бы использования более дорогостоящих GPU, работающих в режиме 64 бит. Кроме эффекта увеличения точности, замечено, что максимальный шаг, при котором отсутствует переобучение ($r < 1.2$) для $k = 3$ оказывается в 2.88 раза больше, чем для

k	0	1	2	3
$\sqrt{\langle e_0 \rangle}$	$3 \cdot 10^{-3}$	$5.4 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	$9.3 \cdot 10^{-5}$
ϑ	$2.8 \cdot 10^{-5}$	$9 \cdot 10^{-6}$	$3.6 \cdot 10^{-6}$	$2.1 \cdot 10^{-6}$
λ	0.052	0.1	0.125	0.15
M	1210	352	233	157

Таблица 3. Решение краевой задачи для нелинейного уравнения Пуассона, обучение с исключением. Результаты усреднены по 25 попыткам.

$k = 0$. В двумерном случае это приводит к уменьшению необходимого числа точек в 7.7 раз, однако, для больших размерностей эффект может оказаться более значительным. Это наблюдение приводит к постановке задачи о влиянии тренировки производных на коэффициент переобучения r, которая рассматривается в Параграфе 2.4.

В Пункте 2.3.2 решается уравнение Бюргерса

$$\frac{\partial u}{\partial t} + u \frac{\partial u}{\partial x} = \frac{1}{\mathtt{R}} \frac{\partial^2 u}{\partial x^2}. \tag{13}$$

Оно часто используется при первичной верификации численных схем для уравнений движения сплошной среды. Поставлено начальное условие

$$u\big|_{t=0} = -\frac{2x}{1+x^2}. \tag{14}$$

Его точное решение для чётных $\mathtt{R}$ имеет вид

$$u_{\mathrm{B}}(t,x) = -\frac{2}{\mathtt{R}} \frac{\sum_{n=0}^{\mathtt{R}/2} \left[\frac{x^{2n-1}}{2^{n-1}(n-1)!(2n-1)!!} \sum_{m=n}^{\mathtt{R}/2} \frac{(2m)!}{m!(\mathtt{R}/2-m)!(m-n)!} \left(\frac{t}{\mathtt{R}}\right)^{m-n} \right]}{\sum_{n=0}^{\mathtt{R}/2} \left[\frac{x^{2n}}{2^n n!(2n-1)!!} \sum_{m=n}^{\mathtt{R}/2} \frac{(2m)!}{m!(\mathtt{R}/2-m)!(m-n)!} \left(\frac{t}{\mathtt{R}}\right)^{m-n} \right]}.$$

Рассматриваемый случай $\mathtt{R} = 60$ изображен на Рисунке 1. Для численного решения выбрана область $t \in [0,1]$, $x \in [-0.65, 0.65]$, при этом на границе $x = 0.65$ вводится условие $u = u_{\mathrm{B}}(t, 0.65)$ и аналогично для $x = -0.65$. Все граничные условия исключаются с помощью замены

$$u = w \cdot t \cdot (x - 0.65)(x + 0.65) + u(0,x) - x/0.65 \left\{ u_{\mathrm{B}}(0, 0.65) - u_{\mathrm{B}}(t, 0.65) \right\}.$$

При $t \sim 1$ вблизи $x = 0$ имеется область с большим градиентом, что приводит к необходимости сгущения сетки. Для относительной точности решения ставится требование $\sim 10^{-6}$, а необходимая плотность

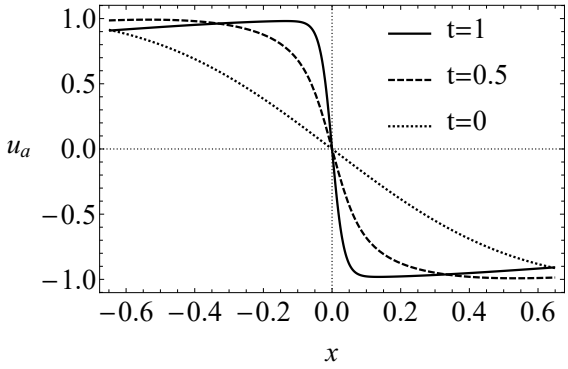


Рис. 1. Решение уравнения Бюргерса с начальным условием (14) для разных моментов времени, $\mathtt{R} = 60$.

точек вычисляется на основе решения предыдущей краевой задачи (7)-(9). Из него можно сделать оценку шага $\lambda \sim 1/3$ при условии, что производные решения имеют порядок единицы. При больших значениях производных следует найти такое масштабирование, при котором они имеют порядок единицы, а затем сетку с шагом $1/3$ на данном масштабе перевести обратно в исходную область. С помощью этого принципа строится неравномерная сетка с переменным шагом по x: $\lambda_x \in [0.006, 0.18]$ и по t: $\lambda_t \in [0.04, 0.25]$ которая имеет $M = 568$ точек. Сетка для обучения без дополнительных производных имеет втрое больший шаг и $M = 4285$ точек. Обучение 4 порядка с числом эпох $T = 2000$ приводит к погрешности $\vartheta = 3{\cdot}10^{-6}$, максимальному отклонению от точного решения $\vartheta_{\max} = 1.7{\cdot}10^{-5}$ и невязке $\sqrt{\langle e_0 \rangle} = 10^{-5}$. Обучение без дополнительных производных при тех же вычислительных ресурсах имеет среднюю точность $\vartheta = 1.2 \cdot 10^{-5}$, максимальное отклонение $\vartheta_{\max} = 1.1 \cdot 10^{-4}$ и среднюю невязку $\sqrt{\langle e_0 \rangle} = 8.2 \cdot 10^{-4}$. Отмечается, что соотношение шага и характерной точности, которое наблюдалось при решении уравнения Пуассона оказались справедливо и при решении уравнения Бюргерса. Указывается на необходимость и возможность построения адаптивного алгоритма поиска сетки в процессе решения.

В Параграфе 2.4 на примере решения краевой задачи для линейного уравнения Пуассона более подробно исследуется эффект отсутствия переобучения. Вводится понятие критического λ_c, как шага, при котором коэффициент переобучения r для невязки становится

больше 1.2. Строится последовательность сеток с постепенно увеличивающимся шагом. С целью оценки зависимости λ_c от порядка k минимизируемой ошибки e_k, решение на каждой сетке производится путём минимизации e_0 и e_4 без использования процедуры исключения, количества арифметических операций уравнены. С целью оценки зависимости λ_c от числа весов, каждое решение получено с помощью двух персептронов с $2 \cdot 10^4$ и 10^6 весов соответственно. Приведены зависимости коэффициентов переобучения, а также относительной точности решения от λ. Из них следует, что для сети с 10^6 весов $\lambda_c(e_4) = 1.4$, $\lambda_c(e_0) < 0.07$. Для сети с $2 \cdot 10^4$ весов $\lambda_c(e_4) = 1.4$, $\lambda_c(e_0) = 0.1$. Отмечено, что такое же отсутствие переобучения, как и при минимизации e_4, можно получить и стандартными методами регуляризации. Однако, в отличии от них, метод обучения с производными дополнительно увеличивает точность решения примерно на порядок. В Пункте 2.4.1 приведено сравнение e_0 и e_4 с конечно-разностными шаблонами на основе комбинации точности решения и использованного шага. Установлено, что e_0 соответствует шаблон четвертого, а e_4 – шаблон шестого порядка. Далее отмечено, что шаблон шестого порядка не смог бы поместиться на сетку, использованную для обучения с e_4. В конце параграфа указывается на то, что при использовании производных допустимо значительное превышение количества весов нейронной сети над количеством тренировочных точек. Так, при минимизации e_4, сеть с миллионом весов можно обучить на 11 точках с $r < 1.2$ при условии, что производные решения, а также диаметр области решения – все имеют порядок единицы. Впервые подобный результат был достигнут только лишь с помощью информации о минимизируемой величине.

В Параграфе 2.5 рассматривается задача вычисления градиента по весам от ошибки нулевого и первого порядка:

$$E_0 = \sum_{a=1}^{M} C_a e^2(\vec{x}_a), \quad E_1 = E_0 + \sum_{a=1}^{M} C_a \sum_{i=1}^{N} \left(\frac{\partial e(\vec{x}_a)}{\partial x_i} \right)^2,$$

на достаточно плотном обучающем множестве $\vec{x}_a \in \Gamma \subset R^N$. Константы C_a есть веса узлов численного интегрирования. Доказывается

Теорема. *Если $M \to \infty$ так, что выражения E_0 и E_1 стремятся к соответствующим мерам, то существуют такие $\widetilde{C}_a$ что*

$$\sum_{a=1}^{M} \widetilde{C}_a \frac{\partial e(\vec{x}_a)}{\partial W} \to \frac{\partial E_1}{\partial W},$$

то есть градиент ошибки первого порядка можно представить линейной комбинацией градиентов ошибки нулевого порядка. При этом

$$\widetilde{C}_a = C_a \cdot 2 \left\{ e\left(\vec{x}_a\right) - \Delta e\left(\vec{x}_a\right) \right\}.$$

Из неё следует, что преимущества обучения с производными можно попытаться получить, умножая вклад от каждой точки в обычный градиент на некоторую величину. Однако, вычисление этих величин потребует знания дополнительных производных от минимизируемого выражения.

Глава 3 посвящена использованию преимущества по величине шага для решения задач большой размерности. В первом параграфе приведен анализ обстоятельств, от которых зависит максимальная величина допустимого шага. Невязка вне точек сетки выражена через формулу Тейлора

$$V\left(\vec{x} + \delta\vec{j}\right) = V\left(\vec{x}\right) + \delta\frac{\partial V}{\partial\vec{j}} + \frac{\delta^2}{2}\frac{\partial^2 V}{\partial\vec{j}^2} + \frac{\delta^3}{6}\frac{\partial^3 V}{\partial\vec{j}^3} + \ldots,$$

Где $\partial/\partial\vec{j}$ есть производная по некоторому направлению:

$$\frac{\partial}{\partial\vec{j}} = j_1\frac{\partial}{\partial x_1} + j_2\frac{\partial}{\partial x_2} + \ldots$$

Так как в общем случае высшие производные по направлению могут зависеть от всех возможных смешанных производных, допустимость больших λ напрямую зависит от малости этих производных. Алгоритм предыдущей главы минимизировал все возможные производные. При переходе к пространствам большой размерности, их количество растёт пропорционально $N^k/k!$, где k – максимальный порядок обучаемых производных. Так как время решения прямо пропорционально количеству вычисляемых производных, ставится задача уменьшения этого множителя.

Для поиска возможных упрощений производятся численные решения краевой задачи для уравнения Пуассона для пространств $2 \le N \le 4$:

$$\Delta u \equiv \sum_{i=1}^{N}\frac{\partial^2 u}{\partial x_i^2} = g, \tag{15}$$

$$\Gamma : \sum_{i=1}^{N} x_i^2 < 1, \tag{16}$$

$$u|_{\partial\Gamma} = 0. \tag{17}$$

Источник g выбран так, чтобы точное решение имело все производные порядка единицы. Для исключения граничного условия используется замена

$$u = w \cdot \left(1 - \sum_{i=1}^{N} x_i^2\right). \tag{18}$$

Согласно результатам Пункта 2.3.2, для решения задачи с точностью $\vartheta \sim 10^{-6}$ достаточно выбрать шаг $\lambda = 1/3$. Минимизируемая величина во всех случаях имеет вид

$$e = \sum_{s \in S} \left(\partial^s V\right)^2. \tag{19}$$

Различается лишь множество S вычисляемых производных. Поскольку стоит задача об исследовании переобучения, метод исключения не используется. Рассматриваются случаи $|s| \leq 3$ и три типа обучения: в первом S содержит все возможные производные. Во втором случае производные берутся только по координатным осям. В третьем случае в каждой точке для дифференцирования выбираются $L \leq N$ случайных перпендикулярных направлений. Далее приведены результаты численных экспериментов, которые показывают, что для всех типов S, невязка на сетке статистически неотличима от невязки во всей области. Даже при минимизации только производных по осям, все смешанные производные оказываются малы. Однако, установлено, что от типа и количества минимизируемых производных зависит среднее значение самой невязки. На основании полученных результатов для оптимальной редукции сложности при сохранении малости невязки выбран метод случайных направлений с $L = 2$. Это позволило уменьшить количество арифметических операций для $N = 4$ вдвое, лишь незначительно увеличив результирующую невязку. В худшем случае можно предположить, что коэффициент увеличения сложности вместо $N^k/k!$ теперь имеет вид $\sim Nk$. В Пункте 3.2.5 описана процедура нормировки членов выражения (19). Пункт 3.2.6 описывает полную процедуру обучения с исключением при использовании случайных направлений.

В Параграфе 3.3 рассматривается решение краевой задачи (15)-(17) методом обучения с исключением с использованием двух случайных направлений для размерностей $2 \leq N \leq 5$. Шаг $\lambda = 1/3$, граничное условие исключается с помощью выражения (18). Сетка для $N = 2$ приведена на Рисунке 2. Решения получены на GPU NVIDIA

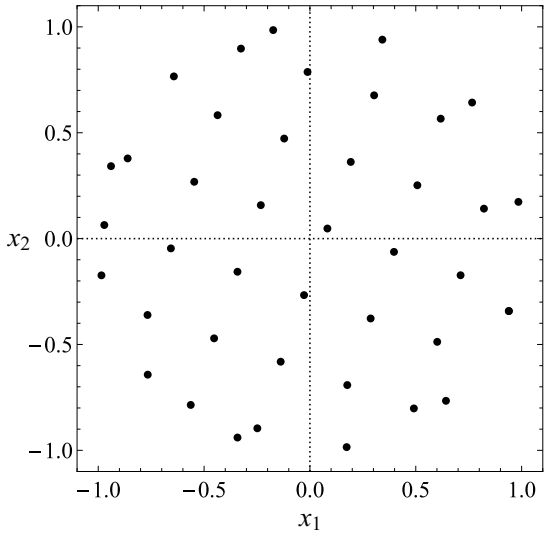


Рис. 2. Двумерная сетка с шагом $\lambda = 1/3$, которая позволяет получить решение с относительной точностью $\sim 10^{-6}$ при условии, что его производные имеют порядок единицы.

Tesla P100 с помощью эффективного алгоритма, описанного в Главе 4. Время решения для $N = 2, 3, 4, 5$ составляет 114, 226, 560 и 1280 секунд соответственно. Отклонения решений от точных во всех случаях имеют медианный модуль $\vartheta_{\text{median}} < 1.5 \cdot 10^{-6}$ и максимальный модуль $\vartheta_{\max} < 9 \cdot 10^{-6}$. Для $N = 5$ использование случайных направлений уменьшает количество арифметических операций в 5 раз.

В Параграфе 3.4 в уравнение (15) добавляется нелинейное слагаемое

$$\Delta u + u^2 = h.$$

Далее оно аналогично решается с краевыми условиями (16), (17) и заменой (18). Для достижения точности $\sim 10^{-6}$ шаг λ уменьшается до $1/4$, что мотивируется более сложным поведением невязки. Время решения для $N = 2$, 3, 4, 5 составляет 180, 380, 1160 и 5000 секунд соответственно.

В Параграфе 3.5 оценивается минимальное время и количество памяти, требуемое для решения линейной задачи (15)-(17) при $N = 5$

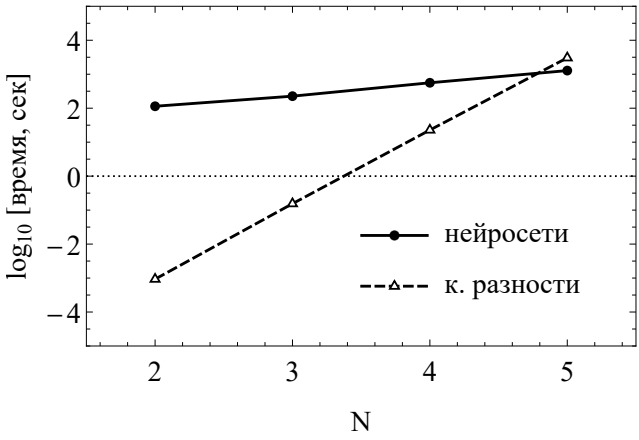


Рис. 3. Зависимость времени решения от размерности задачи для нейронных сетей и метода конечных разностей.

с помощью конечно-разностного метода с центральным шаблоном второго порядка. На основе явного вида решения и требования точности $\vartheta_{\max} < 10^{-5}$ минимальный шаг для метода конечных разностей оценивается как $\lambda = 0.008$. Для области Γ это приводит к оценке $1.6 \cdot 10^{11}$ точек, из которой следует, что уже для хранения решения потребуется $6.4 \cdot 10^{11}$ байт. Нейронная сеть при этом занимает $6 \cdot 10^5$ байт. Далее, с помощью результатов статей Х. Лиу (H. Liu), в которых рассматриваются реализации многосеточных алгебраических решателей на GPU, приводится оценка снизу для времени решения соответствующей линейной системы

$$t_{5\mathrm{D}} = 3000 \text{ сек.}$$

Далее приводятся зависимости времени решения (Рисунок 3) и требуемой памяти (Рисунок 4) от размерности пространства для конечно-разностного и нейросетевого метода. Указанные величины для конечных разностей являются оценками снизу, тогда как для нейросетевого метода они получены прямым измерением. Показывается, что использование замены функции (18) не влияет существенно на результаты сравнения, полученные в данной главе. В конце параграфа указывается, что конечно-разностный шаблон, обеспечивающий шаг $\lambda \sim 1/3$

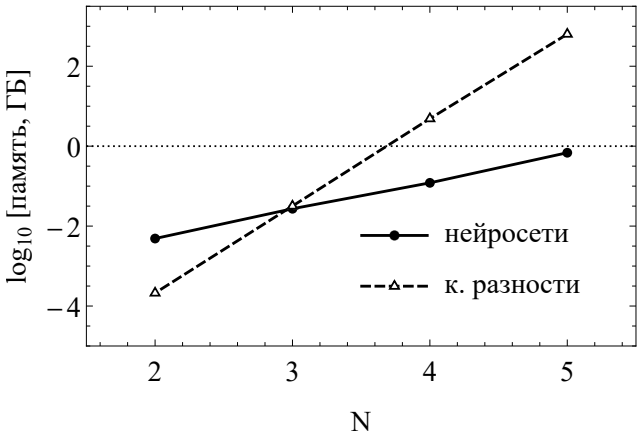


Рис. 4. Зависимость требований по памяти от размерности задачи для нейронных сетей и метода конечных разностей.

не смог бы поместиться на сетку с таким шагом, построенную в области Г.

Параграф 3.6 посвящён практическим аспектам решения уравнений. В Пункте 3.6.1 предложен критерий остановки обучения. Его применимость продемонстрирована на примере решения двух уравнений разных типов. В Пункте 3.6.2 описывается процедура верификации точности решения краевой задачи (15)-(17). Она основана на существовании невязки, легко вычислимой во всей области, и требует порядка 30% от времени решения исходного уравнения. В результате получен гарантированный интервал точности решения, который отличается от истинного в 1.6 раза.

Глава 4 содержит обобщение алгоритма обратного распространения ошибки, которое пригодно для вычисления производных нейронной сети и минимизации невязки уравнений. Алгоритм записан в матричной форме, позволяющей реализовать его на GPU с максимальной эффективностью. Входные вектора x_α теперь перечисляются индексом a и собираются в матрицу $x_{\alpha a}$. Классическая формула (1) для прямого распространения, записанная для двух смежных слоёв κ и θ в таких обозначениях имеет вид:

$$z^{\theta a} = t^\theta + W^{\theta\kappa} \times \sigma\left(z^{\kappa a}\right).$$

С помощью последовательного её применения может быть вычислена матрица значений на выходе: $v^{\omega a}$, где ω есть индекс, перечисляющий нейроны выходного слоя. Путём дифференцирования обеих частей, получена формула прямого распространения для производных

$$\partial^s z^{\theta a} = W^{\theta\kappa} \times \partial^s \sigma\left(z^{\kappa a}\right), \tag{20}$$

здесь член $\partial^s \sigma\left(z^{\kappa a}\right)$ раскрывается как производная от сложной функции. Зная набор матриц $\partial^s x_{\alpha a}$, с помощью (20) можно дойти до выходного слоя и получить все производные $\partial^s v^{\omega a}$. Если дифференцирование идёт по компонентам входа, то в случае первого порядка матрица $\partial^s x_{\alpha a}$ имеет единицы в соответствующей строке, а остальные элементы равны нулю. При больших порядках дифференцирования матрица состоит из нулей. Далее приведены частные случаи выражений $\partial^s \sigma\left(z^{\kappa a}\right)$, которые требовались для решения задач из Глав 2 и 3.

После того, как матрицы производных выхода известны, можно вычислить локальную ошибку

$$e\left(a\right) = e\left(x_{\alpha a}, v^{\omega a}, \partial^{s[1]} v^{\omega a}, \ldots, \partial^{s[k]} v^{\omega a}\right),$$

и минимизируемую величину

$$E\left(x_{\alpha a}, v^{\omega a}, \partial^{s[1]} v^{\omega a}, \ldots, \partial^{s[k]} v^{\omega a}\right) = \sum e\left(a\right).$$

Теперь она зависит не только от значений на выходном слое, но и от их производных. Обратное распространение начинается с вычисления градиента E по элементам всех матриц выхода:

$$\frac{\partial E}{\partial\left(\partial^s v^{\omega a}\right)}.$$

Этот градиент есть набор матриц такого же размера. Он должен распространяться от последующего слоя к предыдущему через замену переменного. Для двух смежных слоёв справедлива формула

$$\frac{\partial E}{\partial\left(\partial^p z^{\kappa a}\right)} = \sum_{s,\theta,a'} \frac{\partial E}{\partial\left(\partial^s z^{\theta a'}\right)} \frac{\partial\left(\partial^s z^{\theta a'}\right)}{\partial\left(\partial^p z^{\kappa a}\right)}. \tag{21}$$

Первый сомножитель под знаком суммы уже был вычислен на предыдущем шаге алгоритма. Для установления общего вида второго сомножителя формулируется

Лемма. *Если* $\partial^s = \partial^p \partial^q$ *то справедлива формула*

$$\frac{\partial}{\partial\left(\partial^p z\right)} \partial^s \sigma\left(z\right) = \binom{s}{p} \partial^q \sigma'\left(z\right),$$

где $\binom{s}{p}$ *есть произведение биномиальных коэффициентов*

$$\binom{s}{p} = \prod_i \frac{s_i!}{p_i!\left(s_i - p_i\right)!}.$$

С её помощью получаются общая формула для обратного распространения

$$\frac{\partial E}{\partial\left(\partial^p z^{\kappa a}\right)} = \sum_s \binom{s}{p} \partial^{s-p} \sigma'\left(z^{\kappa a}\right) \left(\left[W^{\theta\kappa}\right]^T \times \frac{\partial E}{\partial\left(\partial^s z^{\theta a}\right)}\right). \tag{22}$$

Суммирование по s ведется по всему набору распространяемых производных. Из-за сходства выражения $\partial^{s-p}\sigma'\left(z^{\kappa a}\right)$ с формулой прямого распространения, оказывается, что его вычисление требует крайне малого количества дополнительных арифметических операций. Градиент ошибки по матрице весов $W^{\theta\kappa}$ вычисляется с помощью рассмотрения замены $E\left(\partial^s z^{\theta' a}\right) \to E\left(W^{\theta\kappa}\right)$. После упрощений получена формула

$$\frac{\partial E}{\partial W^{\theta\kappa}} = \sum_s \frac{\partial E}{\partial\left(\partial^s z^{\theta a}\right)} \times \left[\partial^s \sigma\left(z^{\kappa a}\right)\right]^T.$$

Выражение градиента по векторам сдвига совпадает с классическим.

$$\frac{\partial E}{\partial t^\kappa} = \sum_a \frac{\partial E}{\partial z^{\kappa a}}$$

В Параграфе 4.3 производится точный подсчет числа арифметических операций для всех задач, рассмотренных в Главе 2. Параграф 4.4 содержит краткое описание платформы CUDA (Compute Unified Device Architecture) в том объеме, который необходим для эффективной реализации алгоритма на GPU. Пункт 4.4.3 описывает некоторые полезные методики, с помощью которых можно увеличить эффективность расчетов для небольших сеток в несколько раз. В последнем абзаце описан метод уменьшения количества промежуточной памяти на 20%. Подпункт 4.4.4 фактически является руководством по применению программного комплекса «cpde-2», разработанного согласно

описанным методикам. Приводятся подробные инструкции для дифференцирования нейронных сетей, приближения их производных и решения дифференциальных уравнений. Отмечается, что без эффективного алгоритма обучения, решение многомерных задач оказывалось бы слишком затратным.

Заключение

В работе было впервые продемонстрировано, что включение производных в процесс градиентного обучения позволяет в несколько десятков раз повысить точность глубоких нейронных сетей, а также понизить число точек, необходимых для их тренировки. На основе этой процедуры был построен алгоритм обучения с исключением, позволяющий увеличить качество тренировки на несколько порядков и достичь относительной точности 10^{-6}, тогда как стандартный метод градиентного обучения в аналогичной ситуации имеет точность порядка 10^{-3}. При решении уравнений в частных производных, обучение с исключением позволяет сравнять точность нейросетевого метода с точностью классических методов, а также существенно понизить требования на минимальный шаг. Метод также применим для получения решений с большими градиентами, при наличии которых требуется введение неравномерной сетки. Для многомерных краевых задач, в совокупности с разработанным методом случайных направлений, метод обучения с исключением впервые оказывается быстрее конечно-разностной схемы, а также значительно снижает требования по памяти из-за большего допустимого шага сетки. Математические причины увеличения точности и строгие доказательства выходят за пределы данной работы. Тем не менее, нахождение класса задач, точность и эффективность решения которых с помощью нейронных сетей может быть значительно увеличена, имеет практическое значение.

Для большинства примеров, рассмотренных в данной диссертации, разработанный программный комплекс «cpde-2» имеет аппаратную эффективность (отношение реальной производительности к теоретической) порядка 50–80%. Для классического обучения нейронных сетей без дополнительных производных, эффективность широкодоступных комплексов программ сейчас является величиной того же порядка. Однако, эта производительность является результатом бурного развития машинного обучения в последние годы. Так, ещё в 2015 году их эффективность для стандартных постановок была около 12%.

Так как задачи, связанные с высоким порядком дифференцирования до сих пор практически не исследовались, эффективность общедоступных комплексов для них сейчас еще ниже 12%. В данной работе у обучения с производными обнаружен значительный потенциал, а разработанный для этих целей комплекс программ «cpde-2», эффективно использующий GPU, может ускорить получение новых результатов в этой области.

Эффект от включения производных в процесс обучения глубоких сетей ни в коем случае не ограничиваются случаями малой размерности, и автором было зафиксировано крайне благоприятное их влияние на аппроксимацию функции, определённой на 64–мерном единичном кубе. Большим потенциалом обладает задача об адаптации результатов данной диссертации к распознаванию образов. По-видимому, для этого необходимо вычисление производных от функций типа классификатора.

Публикации автора по теме диссертации

[1] V.I. Avrutskiy. Enhancing function approximation abilities of neural networks by training derivatives. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. https://doi.org/10.1109/TNNLS.2020.2979706.

[2] V.I. Avrutskiy. Neural networks catching up with finite differences in solving partial differential equations in higher dimensions. *Neural Computing and Applications*, 32:13425–13440, 2020. https://doi.org/10.1007/s00521-020-04743-8.

[3] V.I. Avrutskiy. Preventing overfitting by training derivatives. In *Proceedings of the Future Technologies Conference*, pages 144–163. Springer, 2019. https://doi.org/10.1007/978-3-030-32520-6_12.

[4] V.I. Avrutskiy. Backpropagation generalized for output derivatives. *arXiv preprint arXiv:1712.04185*, 2017. https://arxiv.org/abs/1712.04185.

Отпечатано с оригинал-макетов Заказчика
в типографии "Переплетофф"
Адрес: г. Долгопрудный, ул. Циолковского, 4.
Тел: 8(903) 511 76 03. www.perepletoff.ru
Формат 148 x 210 мм. Бумага офсетная.
Печать цифровая. Тираж 20 экз.
Мягкий переплет.
Заказ № _____. 07.04.21 г.