Применение физически информированного глубокого обучения для решения дифференциальных уравнений

Даниил Лаптев

22 марта 2024 г.

Содержание

Введение

Дифференциальные уравнения являются одним из ключевых инструментов для описания и анализа окружающего мира. Существует множество способов их классификации согласно каким-либо особенностям формулировки или свойств, которыми обладают решения. Например, они бывают линейными и нелинейными; существует известное деление линейных ДУ в частных производных на гиперболические, параболические и эллиптические; ДУ могут быть детерминированными или стохастическими.

Существует ряд проблем. Во-первых - для некоторых методов, например таких, как конечные разности, конечные элементы и конечные объёмы, существует необходимость выбора подходящего разбиения пространства на соответствующие ячейки, для центров или для узлов которых мы будем вычислять значение искомой функции. Это может быть неприемлемо в некоторых сценариях (например, для моделирования нестатичной геометрии, которая подвержена значительным деформациям или в которой узлы могут исчезать или появляться), или в том случае, когда решение необходимо искать на довольно точной дискретизации ввиду особенностей решения (выраженных нелинейностей, особых точек, шоков, разрывов и прочего) и, кроме того, в случае большого числа переменных это крайне затратно с вычислительной точки зрения - здесь мы наталкиваемся на проклятье размерности. Во-вторых - зоопарк численных методов и относительная сложность их самостоятельной реализации представляет собой ограничение с точки зрения их использования непрофессионалами. Для каждого отдельного случая универсальный, простой метод может не подойти по причине недостаточной точности или неприменимости для решения конкретной задачи; поиск более подходящего, улучшение существующего или выработка нового представляют собой тяжёлое предприятие, которое может отнять много времени.

Использование методов глубокого обучения для таких задач представляет собой перспективное направление, которое в последние годы всё более активно внедряется в науку и технику. Появление методологии Physics-Informed Neural Networks [5] позволило по-новому взглянуть на то, как можно обучить нейросеть удовлетворять тем или иным законам физики. Технология автоматического дифференцирования позволяет использовать дифференциальные уравнения как один из факторов регуляризации нейросетевой модели - мы внедряем в процесс обучения некоторую априорную информацию, выраженную в форме дифференциальных уравнений, и заставляем нейросетевую модель удовлетворять этим ограничениям. Иными словами, мы информируем нейросетевую модель о том, какие предсказания она должна давать.

В статье [5] этот подход применён для решения дифференциальных уравнений (прямая задача), а также для идентификации системы - при заданном дифференциальном уравнении и данных о протекании процесса мы решаем задачу обнаружения параметров этого ДУ (обратная задача). Впоследствии эта методология стала применяться для решения различных специфических инженерных задач, краткий обзор которых можно найти в [1] вместе с обзором методологии PINN и её возможных путей развития. В свою очередь экспериментальный и теоретический анализ процесса обучения PINNs для решения дифференциальных уравнений нельзя назвать достаточным, в нём остаются незаполненные пробелы - например, что можно сказать касательно того, как в общем случае связаны краевые задачи, размер нейросетевой модели и гиперпараметры процесса обучения? Как именно алгоритм оптимизации сходится к решению, и каковы условия этой сходимости? Какие существуют особенности процесса обучения PINN, которые заслуживают внимания? И многие другие вопросы.

В данном исследовании будет систематически рассмотрено применение классической методологии PINN для решения ряда различных дифференциальных уравнений, которые отличаются между собой формулировкой, поведением решений, а также сложностью

отыскания их решения. В качестве таковых были выбраны простые, но распространённые типы задач - обыкновенное дифференциальное уравнение второго порядка, система ОДУ первого порядка, а также параболическое уравнение в частных производных. Будет предпринята попытка выявить основные принципы работы PINN, основные правила оптимального подбора гиперпараметров, а также обнаружить некоторые особенности тренировки моделей, которые могут представлять интерес с точки зрения теории глубокого обучения или дифференциальных уравнений.

Целью нашего исследования является рассмотрение принципов применения методологии PINN для решения некоторых распространённых типов дифференциальных уравнений (в качестве иллюстративных примеров мы выбрали гармонический осциллятор с затуханием, систему Лотки-Вольтерры и уравнение диффузии), а также выявление того, какую роль играют гиперпараметры нашего сеттинга обучения. Для достижения цели были поставлены следующие задачи:

1. Реализовать классические методы для получения решений каждого из рассматриваемых ДУ. Эти решения будут эталонными.
2. Решить каждую из задач с помощью PINN. Обнаружить, при каких гиперпараметрах решение можно назвать удовлетворительным.
3. Выявить, как гиперпараметры влияют на решение каждой из задач.
4. Рассмотреть модели разного размера. Как меняются оптимальные гиперпараметры?
5. Рассмотреть те же задачи, но с другими параметрами и граничными условиями. Провести анализ влияния гиперпараметров и моделей на качество решения. Осталась ли картина той же, что в пунктах 3 и 4?
6. Сделать выводы относительно того, как достичь удовлетворительного результата, используя методологию PINN, и можно ли это сделать.

1 Классическая методология PINNs

Мы полагаемся на следующую гарантию - почти всякое решение ДУ можно приблизить с произвольной точностью некоторой нейросетевой моделью, если правильно подобрать её архитектуру. В общем случае это утверждение представляется как свойство универсальной аппроксимации [2]. Таким образом, мы больше заинтересованы в процессе приближения решения - достаточно ли экспрессивная модель выбрана, существует ли проблема нехватки информации, затухания градиентов и т.п., насколько хорошие гиперпараметры выбраны, способен ли метод оптимизации эффективно сойтись к минимуму и так далее - нежели в анализе того, какие свойства присущи искомому решению. Может быть, эти свойства прольют свет на какие-то проблемы во время тренировки, или позволят построить нейросетевую модель с определёнными предположениями о решении и тем самым улучшить качество решения, но это является скорее вспомогательным источником информации, нежели необходимым.

Рассмотрим общую постановку PINN. Пусть нам задана некоторая задача с граничными условиями [1]:

$$\begin{aligned} \mathcal{D}[u(z), \gamma] &= f(z) \quad z \in \Omega, \\ \mathcal{B}[u(z)] &= g(z) \quad z \in \partial\Omega, \end{aligned} \tag{1}$$

где элемент z обычно интерпретируется как набор из D пространственных координат и одной временной координаты, т.е. $z = [x_1, ..., x_D, t]^T \in \mathbb{R}^{D+1}$. Граничные условия также

описывают начальные условия при нижней границе $t = 0$. Довольно часто $f(z) \equiv 0$. Дифференциальный оператор $\mathcal{D}$ обычно параметризован набором чисел γ, который определяет специфический вид ДУ.

Имея некоторую параметризованную функцию $\mathcal{N}(z;\theta)$, мы должны найти такой набор параметров θ^*, что $\mathcal{N}(z;\theta^*) \approx u(z)$ на всём домене Ω.

Методология PINN в сущности представляет собой метод регуляризации процесса обучения нейросетевой модели, при котором мы ограничиваем множество возможных параметров при помощи информации из заданных нами дифференциальных уравнений. Иными словами, множество допустимых параметров Θ описывается как

$$\Theta = \{\theta : \quad \mathcal{D}[\mathcal{N}(z;\theta),\gamma] \approx f(z) \quad \text{и} \quad \mathcal{B}[\mathcal{N}(z;\theta)] \approx g(z)\}$$

Учитывая, что нейросетевые модели редко представляют собой точную аппроксимацию какой-либо функции, мы используем знак $\approx$ вместо $=$, однако понятие близости двух функций очень сильно зависит от конкретной задачи. Возникает следующая ситуация. В реальных условиях, используя нейросетевые аппроксиматоры и ограниченные вычислительные ресурсы, мы не надеемся получить $\mathcal{N}$ такую, что она будет целиком удовлетворять заданным условиям и являться точным выражением $u(z)$ - всегда существует ошибка того или иного рода, которую нам хотелось бы свести к минимуму. В данном случае можем явно выразить интересующую нас ошибку следующим образом.

Пусть зафиксирована какая-либо архитектура нейросетевой модели, и для неё задан конкретный вектор параметров $\theta \in \mathbb{R}^p$. Тогда качество нейросетевой аппроксимации мы определяем как

$$\begin{aligned} \mathcal{L}_D(\theta) &= \int_\Omega \|\mathcal{D}[\mathcal{N}(z;\theta),\gamma] - f(z)\| dz \\ \mathcal{L}_B(\theta) &= \int_{\partial\Omega} \|\mathcal{B}[\mathcal{N}(z;\theta)] - g(z)\| dz \end{aligned} \tag{2}$$

На практике эти величины можно оценить методом Монте-Карло, а в качестве нормы выбрать квадрат Евклидовой нормы - тогда мы получим метрику Mean Squared Error, используемую нами в качестве функции потерь для оптимизации нейросетевой модели.

Функции ошибки $\mathcal{L}_D$ и $\mathcal{L}_B$ позволяют оценить, насколько хорошо нейросетевая модель удовлетворяет постановке краевой задачи. Мы ожидаем, что модель, минимизирующая обе функции ошибки, тем самым аппроксимирует решение. Более того, в некоторых случаях мы даже обладаем гарантиями сходимости (см. например [4], теорема 1).

Сформулируем полную функцию ошибки. На границе мы выбираем пары $\{z_i, u(z_i)\}$, исходя из постановки задачи, а в области Ω мы выбираем произвольные точки $\{z_j\}$. Пусть количество выбранных точек на границе будет $N_\mathcal{B}$, внутри области - $N_\mathcal{D}$. Тогда в качестве функции ошибки можно использовать взвешенную сумму ошибок внутри области Ω и на её границе:

$$\mathcal{L}(\theta) = \frac{\alpha}{N_\mathcal{B}} \sum_{i=1}^{N_\mathcal{B}} (\mathcal{B}[\mathcal{N}(z_i;\theta)] - u(z_i))^2 + \frac{\beta}{N_\mathcal{D}} \sum_{j=1}^{N_\mathcal{D}} (\mathcal{D}[\mathcal{N}(z_j;\theta),\gamma] - f(z_j))^2 \tag{3}$$

Отметим, что до сих пор мы обходимся без каких-либо данных, кроме тех, которые являются частью постановки краевой задачи. В стандартном сеттинге обучения с учителем мы обладаем информацией о значениях искомой функции в некоторых точках - чем её больше, тем лучше. В текущем сеттинге мы обладаем лишь информацией о том, какие значения принимает искомая функция (и/или её производные) на границах заданного домена Ω. Кроме того, мы можем получить приближённое решение, даже не имея явно заданной функции на границе, а только набор её измерений - это мы рассмотрим далее.

Если вдруг у нас имеется набор значений искомой функции, мы можем добавить ещё один терм функции потерь.

В конечном итоге для какой-либо фиксированной нейросетевой архитектуры мы ищем такой набор параметров θ^*, что

$$\theta^* = \arg\min_\theta \mathcal{L}(\theta)$$

В качестве нейросетевой архитектуры мы используем модели прямого распространения (FF), которые представляются как композиция аффинных и нелинейных преобразований:

$$\begin{aligned} Linear_i(x) = W_i x + b_i, \\ FF(z) = Linear_L(...\nu(Linear_1(z))), \end{aligned} \tag{4}$$

где ν - некая нелинейная функция, в нашем случае - гиперболический тангенс.

Для наших экспериментов мы возьмём такие модели, у которых количество строк в матрицах $W_1, ..., W_{L-1}$ одинаково. Пусть это количество (ширина нейросети) записывается как W. Тогда модель глубины L и ширины W будет обозначаться как $\mathrm{FF}_{L,W}$.

2 Решение дифференциальных уравнений

2.1 Гармонический осциллятор с затуханием

На данный момент нас не интересуют её физические свойства, только математическая формулировка. Система описывается следующим образом:

$$\frac{\mathrm{d}^2 x}{\mathrm{d}t^2} + 2\zeta\omega_0 \frac{\mathrm{d}x}{\mathrm{d}t} + \omega_0^2 x = 0, \tag{5}$$

где функция $x(t) : \mathbb{R} \to \mathbb{R}$ описывает положение осциллирующей массы, а параметры ζ и ω влияют на характер затухания осцилляции.

Пусть дано начальное положение массы $x(0) = x_0$ и начальная скорость $v(0) = \frac{\mathrm{d}x}{\mathrm{d}t}(0) = v_0$. Зафиксируем некоторую нейросетевую архитектуру $\mathcal{N}(z;\theta)$. Тогда мы должны отыскать такие параметры θ, которые минимизируют следующий функционал:

$$\mathcal{L}(\theta) = \frac{\alpha}{2}((\mathcal{N}(0;\theta) - x_0)^2 + (\frac{\mathrm{d}\mathcal{N}}{\mathrm{d}t}(0;\theta) - v_0)^2) + \frac{\beta}{N_\mathcal{D}} \sum_{i=1}^{N_\mathcal{D}} \Big(\Big[\frac{\mathrm{d}^2}{\mathrm{d}t^2} + 2\zeta\omega_0 \frac{\mathrm{d}}{\mathrm{d}t} + \omega_0^2 \Big] \mathcal{N}(t_i;\theta) \Big)^2$$

2.2 Система Лотки-Вольтерры

Теперь рассмотрим модель взаимодействия двух биологических видов, один из которых выполняет роль хищника, другой - жертвы. Такая система часто моделируется с помощью стандартной системы Лотки-Вольтерры. Пусть число особей в их популяциях описывается, соответственно, $y(t)$ и $x(t)$, которые являются непрерывными функциями. Тогда динамика числа особей может быть выражена с помощью системы ОДУ первого порядка

$$\begin{cases} \frac{\mathrm{d}x}{\mathrm{d}t} = \alpha x - \beta xy \\ \frac{\mathrm{d}y}{\mathrm{d}t} = \delta yx - \gamma y \end{cases}, \tag{6}$$

где параметры $\alpha, \beta, \delta, \gamma$ характеризуют рождаемость и смертность популяций.

Пусть нейросетевая модель $\mathcal{N} : \mathbb{R} \to \mathbb{R}^2$ будет аппроксимировать значение сразу двух искомых функций в момент времени t посредством двумерного вектора на её выходе, и мы будем писать

$$\mathcal{N}(t;\theta) = \begin{bmatrix} \mathcal{X}(t) \\ \mathcal{Y}(t) \end{bmatrix}_\theta \approx \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}$$

Пусть даны начальные значения x_0, y_0. Нам необходимо найти такие параметры θ, которые минимизируют следующие функции ошибки:

$$\begin{aligned}
\mathcal{L}_{\mathcal{I}} &= \frac{1}{2}\Big(\mathcal{N}(0;\theta) - \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}\Big)^2 \\
\mathcal{L}_{\mathcal{X}} &= \frac{1}{N_{\mathcal{D}}}\sum_{i=1}^{N_{\mathcal{D}}}\Big(\frac{\mathrm{d}\mathcal{X}}{\mathrm{d}t}(t_i) - \alpha\mathcal{X}(t_i) + \beta\mathcal{X}(t_i)\mathcal{Y}(t_i)\Big)^2 \\
\mathcal{L}_{\mathcal{Y}} &= \frac{1}{N_{\mathcal{D}}}\sum_{i=1}^{N_{\mathcal{D}}}\Big(\frac{\mathrm{d}\mathcal{Y}}{\mathrm{d}t}(t_i) - \delta\mathcal{X}(t_i)\mathcal{Y}(t_i) + \gamma\mathcal{Y}(t_i)\Big)^2
\end{aligned} \tag{7}$$

В качестве итоговой функции потерь мы возьмём их линейную комбинацию:

$$\mathcal{L}_{\mathcal{T}} = w_1\mathcal{L}_{\mathcal{I}} + w_2\mathcal{L}_{\mathcal{X}} + w_3\mathcal{L}_{\mathcal{Y}}$$

2.3 Уравнение диффузии

Эта модель используется как самостоятельное описание процесса диффузии или как особый случай некоторых других моделей. Мы рассмотрим одномерный случай и задачу с граничными условиями.

Пусть имеется функция $u(\mathbf{x}, t) : \mathbb{R}^{N+1} \to \mathbb{R}$, которая описывает, например, концентрацию вещества в какой-либо точке пространства и времени (в нашем случае $N = 1$). Пусть число D, называемое коэффициентом диффузии, является постоянным на всей ограниченной области $[A, B] \times [0, T]$. Кроме того, на границах этой области нам известно поведение искомой функции. Тогда краевую задачу для уравнения диффузии мы запишем как

$$\begin{aligned}
&\frac{\partial u}{\partial t} = D\frac{\partial^2 u}{\partial x^2}, \\
&u(x, 0) = f(x), \\
&u(A, t) = g_1(t), \\
&u(B, t) = g_2(t),
\end{aligned} \tag{8}$$

причём вместо явно заданных функций на границах мы можем иметь лишь некоторый набор измерений. Методы машинного обучения позволят приблизить решение на основе неполных данных, что является ещё одним достоинством PINNs.

Пусть при $t = 0$ и на границах выбирается соответственно N_I, N_A, N_B точек вместе с заданными в них значениями искомой функции. Внутри области мы берём N_D произвольных точек. Тогда функции потерь мы можем записать как

$$\begin{aligned}
\mathcal{L}_{\mathcal{I}} &= \frac{1}{N_I}\sum_{i=1}^{N_I}(\mathcal{N}(x_i, 0) - f(x_i))^2, \\
\mathcal{L}_{\mathcal{B}} &= \frac{1}{N_A}\sum_{i=1}^{N_A}(\mathcal{N}(A, t_i) - g_1(t_i))^2 + \frac{1}{N_B}\sum_{i=1}^{N_B}(\mathcal{N}(B, t_i) - g_2(t_i))^2, \\
\mathcal{L}_{\mathcal{D}} &= \frac{1}{N_D}\sum_{i=1}^{N_D}\Big(\frac{\partial\mathcal{N}}{\partial t}(x_i, t_i) - D\frac{\partial^2\mathcal{N}}{\partial x^2}(x_i, t_i)\Big)^2
\end{aligned} \tag{9}$$

Вместо того, чтобы считать просто взвешенную сумму термов функции потерь, мы выбираем коэффициент $\alpha \in (0, 1)$ и вычисляем сумму следующего вида:

$$\mathcal{L}_{\mathcal{T}} = \alpha(\mathcal{L}_{\mathcal{I}} + \mathcal{L}_{\mathcal{B}}) + (1 - \alpha)\mathcal{L}_{\mathcal{D}}.$$

Гипотетически, это позволяет нам балансировать между качеством аппроксимации на границе и внутри домена. Этот подход вдохновлён работой [4], в которой даются теоретические оценки оптимального выбора коэффициента α.

3 Анализ гиперпараметров

Мы занимаемся поиском точки θ^* из пространства параметров $\mathbb{R}^p$, которая удовлетворяет условиям 2. Поиск производится итеративно с помощью алгоритма градиентного спуска. На i-й итерации для набора данных $\mathbf{X}$ мы вычисляем градиент функции потерь $\nabla\mathcal{L}(\mathbf{X};\theta_i)$ относительно параметров θ_i с помощью автоматического дифференцирования, и выполняем переход к θ_{i+1} согласно какому-либо правилу (алгоритму оптимизации), тем самым обновляя параметры нашей нейросетевой модели.

Этот процесс можно представить как движение оптимизатора по *ландшафту потерь*, который определяется как поверхность, заданная сильно невыпуклой функцией потерь $\mathcal{L} : \mathbb{R}^p \to \mathbb{R}$. В силу невыпуклости ландшафта, а также неоптимальной желаемой скорости сходимости оптимизатора, мы можем столкнуться с различными проблемами его движения (например, застревание в локальных минимумах или резкие минимумы, в которые оптимизатор зайти не может), которые обычно преодолеваются либо конструированием нового ландшафта потерь, либо подбором более подходящих гиперпараметров.

Таким образом, в вопросе сходимости процесса обучения является важным рассмотрение геометрии ландшафта потерь и движения по нему оптимизатора. Теоретические исследования показывают, что геометрия ландшафта потерь имеет некоторую связь со способностью нейросетевой модели к обобщению, и, кроме того, выбор нейросетевой архитектуры может значительно влиять на сложность оптимизации из-за изменения ландшафта потерь [3]. Анализ ландшафта потерь как такового мы оставляем на другой раз, однако для лучшего понимания процесса обучения мы должны разделять эти две его стороны.

Говоря о ландшафте потерь, мы также понимаем под этим вычислительный граф, который формируется при вычислении значения функции потерь. Всё, что влияет на результат этих вычислений, влияет и на геометрию ландшафта потерь.

Введём следующую классификацию параметров:

1. Краевая задача: дифференциальное уравнение, граничные условия, параметры γ.
2. Нейросетевая модель: архитектура и конкретный способ её использования.
3. Задача оптимизации: функция потерь и коэффициенты входящих в неё термов.
4. Обучение: параметры оптимизатора, инициализация модели, точки коллокации.

Эта классификация поможет нам быть более осмотрительными в вопросе анализа гиперпараметров. В общем случае можно сказать, что в ней существуют только 3 и 4 пункты, т.к. нейросеть и постановка задачи встроены в функцию потерь, однако мы рассчитываем использовать нейросетевую модель после обучения, поэтому выделяем её в отдельную сущность, а постановку задачи рассматриваем как наиболее важный пункт, определяющий все остальные. Ландшафт потерь определяется первыми тремя пунктами, движение по нему - четвёртым.

Ради упрощения мы предположим, что на каждой итерации процесса оптимизации используются одни и те же граничные точки и соответствующие им значения искомой функции, т.е., используется фиксированный заранее полный тренировочный датасет. Кроме того, мы рассмотрим лишь решение модели Лотки-Вольтерры (второй подход) и уравнение диффузии.

Для анализа гиперпараметров мы выберем четыре различные постановки задачи, обнаружим для него удовлетворительное решение и будем отталкиваться от полученного набора гиперпараметров и итогового RMSE. Детали описаны в приложении **??**.

4 Анализ сходимости PINNs к решению ДУ

4.1 Теоретические гарантии сходимости

4.2 Оценка глобальной погрешности решения

Простейший способ проанализировать, насколько хорошо работает наше нейросетевое решение - сравнить его с общепринятыми методами численного решения, а ещё лучше - рассмотреть задачи с доступным точным решением. Доверие к нему, таким образом, будет опираться на опыт. Однако существует ряд теоретических методов, которые мы рассмотрим далее.

Возьмём нотацию, описанную в [1]. Пусть $u(z) : \Omega \to X$ - искомая функция, $\mathcal{N}(z;\theta)$ - нейросетевая модель с фиксированной архитектурой. Мы хотим получить оценку величины

$$\|\mathcal{N}(z;\theta) - u(z)\|,$$

основываясь на информации об архитектуре модели и тренировочных данных, а также, может быть, на догадках о каких-либо свойствах искомой функции.

Стоит ещё раз заметить, что функция $\mathcal{N}(z;\theta)$ не определяется единственным образом своими параметрами, а имеет также и архитектуру - конкретный способ реализации параметров. Он может заметно влиять на качество решения и его также необходимо брать в расчёт. Для того, чтобы стандартизовать анализ, можно ввести понятие сложности функции, или её экспрессивности, и выработать способ количественного измерения этой характеристики. Возможно, это позволит упростить методологию.

Пусть дан набор точек z_i и соответствующих значений функции $u(z_i)$, где $i \in \{1, ..., N\}$ (например, это какая-либо совокупность измерений искомого решения ДУ, начальных и граничных условий). Тогда *эмпирический риск* для модели с фиксированными параметрами θ определяется как

$$\hat{\mathcal{R}}[\mathcal{N}] = \frac{1}{N}\sum_{i=1}^{N} \|\mathcal{N}(z_i) - u(z_i)\|^2$$

Общий *риск* модели, характеризующий её способность аппроксимировать функцию на всём Ω, будет определяться как

$$\mathcal{R}[\mathcal{N}] = \int_{\Omega} (\mathcal{N}(z) - u(z))^2 dz$$

Качество решения можно оценить, исходя из трёх факторов.

Ошибка оптимизации. Обозначим модель, полученную в результате тренировки, как $\mathcal{N}^*$. Мы введём *ошибку оптимизации* как меру качества тренировочного процесса для данной нейросетевой архитектуры на данных тренировочных точках. Мы ожидаем, что если достигнут минимум ошибки оптимизации, дальнейший способ улучшения качества - поиск новых данных и использование новых архитектур.

Обычно ошибка оптимизации нам неизвестна на практике, т.к. функция потерь является сильно невыпуклой, а стандартные методы оптимизации дают лишь приближённое решение (нередко застревая в локальном минимуме). Математически она будет выражаться

как разница между эмпирическим риском полученной модели $\hat{u}^*$ и наименьшим эмпирическим риском из всех возможных:

$$\varepsilon_O = \hat{\mathcal{R}}[\mathcal{N}^*] - \inf_\theta \hat{\mathcal{R}}[\mathcal{N}]$$

Равенство $\varepsilon_O = 0$ будет означать, что мы нашли глобальный минимум.

Ошибка аппроксимации. Введём *ошибку аппроксимации*, описывающую, насколько хорошо мы можем аппроксимировать искомую функцию, используя данную архитектуру. Математически она определяется как наименьший возможный общий риск для данной архитектуры:

$$\varepsilon_A = \inf_\theta \mathcal{R}[\mathcal{N}]$$

Способность нейросетей аппроксимировать различные классы функций широко изучена в литературе и именно в данном контексте играют роль теоремы универсальной аппроксимации, которые, в сущности, утверждают, что ϵ_A можно сделать сколь угодно малой при правильном выборе архитектуры модели и количества параметров. Редко, однако, говорится о том, как именно выбирать архитектуру и размер модели - эта область ещё мало изучена, хоть в ней и есть свои результаты.

Ошибка обобщения. Она характеризует способность нейросети делать верные предсказания на тех данных, которые не встречались в тренировочном наборе. Она выразится как максимальное отклонение общего риска от эмпирического риска при данной архитектуре и данных тренировочных точках:

$$\varepsilon_G = \sup_\theta |\mathcal{R}[\mathcal{N}] - \hat{\mathcal{R}}[\mathcal{N}]|$$

Заключение

Список источников

[1] Salvatore Cuomo и др. «Scientific Machine Learning Through Physics–Informed Neural Networks: Where we are and What's Next». В: *Journal of Scientific Computing* 92.3 (июль 2022), с. 88. ISSN: 1573-7691. DOI: 10.1007/s10915-022-01939-z. URL: https://doi.org/10.1007/s10915-022-01939-z.

[2] Anastasis Kratsios. «The Universal Approximation Property». В: *Annals of Mathematics and Artificial Intelligence* 89.5 (июнь 2021), с. 435—469. ISSN: 1573-7470. DOI: 10.1007/s10472-020-09723-1. URL: https://doi.org/10.1007/s10472-020-09723-1.

[3] Hao Li и др. *Visualizing the Loss Landscape of Neural Nets*. 2018. arXiv: 1712.09913 [cs.LG].

[4] Remco van der Meer, Cornelis Oosterlee и Anastasia Borovykh. *Optimally weighted loss functions for solving PDEs with Neural Networks*. 2021. arXiv: 2002.06269 [math.NA].

[5] M. Raissi, P. Perdikaris и G.E. Karniadakis. «Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations». В: *Journal of Computational Physics* 378 (2019), с. 686—707. ISSN: 0021-9991. DOI: https://doi.org/10.1016/j.jcp.2018.10.045. URL: https://www.sciencedirect.com/science/article/pii/S0021999118307125.