Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный университет имени М.В.Ломоносова»
Механико-математический факультет

Образовательная программа
«Фундаментальная математика и механика»

КУРСОВАЯ РАБОТА
на тему
«ПРИМЕНЕНИЕ ТЕОРИИ ИСКУССТВЕННЫХ НЕЙРОННЫХ СЕТЕЙ ДЛЯ РЕШЕНИЯ ПРЯМОЙ ЗАДАЧИ ИЗГИБА УПРУГОГО СТЕРЖНЯ»

Студента 3 курса
группы №325
Щербакова Михаил Евгеньевича

Научный руководитель:
Бобылев Александр Александрович

Москва
2021

Содержание

Введение **3**

1 Постановка механической задачи **3**

2 Применение ИНС в механической задаче **5**

3 Выдвижение основополагающей гипотезы **5**

4 Алгоритм определения расстояния до выпуклой оболочки **6**

5 Экспериментальная проверка гипотезы **8**

6 Построение обучающего набора **10**

Выводы **13**

Список литературы **13**

Введение

Рассмотрим возможность применения Искусственных Нейронных Сетей (далее - ИНС) в задачах механики деформируемого твёрдого тела. Целью данной работы является исследование аппроксимационных возможностей ИНС и построение минимального обучающего набора для решения прямой задачи о растяжении упругого стержня.

1 Постановка механической задачи

Рассмотрим простейшую модельную задачу **о растяжении упругого стержня** массовыми силами, параллельными оси стержня.

Будем рассматривать только смещения u точек стержня длины l в направлении оси x, предполагая, что эти смещения зависят только от координаты x. В качестве внешних воздействий рассмотрим объёмные силы с плотностью $F = F(x)$, рассчитываемые на единицу площади поперечного сечения. Площадь поперечного сечения обозначим через $S = S(x)$.

Деформация стержня определяется величиной

$$\varepsilon = \frac{\mathrm{d}u}{\mathrm{d}x}, \tag{1}$$

имеющей смысл относительного удлинения бесконечно малого отрезка длины $\mathrm{d}x$.

Напряженное состояние стержня определяется плотностью поверхностных усилий $\sigma(x)$ в поперечном сечении, направленных параллельно оси x и рассчитываемых на единицу площади сечения.

Связь между σ и ε определяется одномерным **законом Гука**

$$\sigma = E\varepsilon. \tag{2}$$

Дифференциальное **уравнение равновесия стержня** имеет вид

$$-\frac{\mathrm{d}}{\mathrm{d}x}\left[ES(x)\frac{\mathrm{d}u}{\mathrm{d}x}\right] = F(x)S(x) \equiv f(x). \tag{3}$$

Граничные условия для случая жёсткого защемления

$$u(0) = 0,\ u(l) = 0. \tag{4}$$

Задача (3)-(4) представляет собой простейшую **краевую задачу** для обыкновенного дифференциального уравнения второго порядка. Её можно решить **методом конечных элементов** (далее - МКЭ).

Процесс решения краевых задач математической физики методом конечных элементов состоит из следующих этапов [1]:

- Построение вариационной формулировки краевой задачи в виде вариационного уравнения или задачи минимизации функционала энергии
- Разбиение рассматриваемой области на конечное число непересекающихся подобластей, далее именуемых конечными элементами
- Построение кусочно-полиномиальных базисных функций
- Формулировка дискретной задачи метода Бубнова-Галеркина/Ритца
- Формирование СЛАУ для нахождения узловых значений приближенного решения
- Вычисление производных решения и или других искомых величин
- Апостериорный анализ полученного приближенного

В итоге получаем СЛАУ для нахождения обобщённых узловых перемещений вида

$$Ku = R, \tag{5}$$

где K — квадратная матрица жёсткости

u — вектор узловых перемещений

R — вектор распределённых узловых нагрузок

Назовём *прямой задачей* — задачу нахождения обобщённых узловых перемещений u по заданной распределённой нагрузке R.

2 Применение ИНС в механической задаче

Мы имеем прямую задачу, связывающую распределённые нагрузки и обобщенные перемещения в конечных элементах. То есть, присутствует некоторый линейный оператор и нашей целью является попытка аппроксимировать его с помощью ИНС.

Для этого необходимо построить некоторый обучающий набор (т.е. набор экспериментов). Преимущество в том, что мы сами можем генерировать такие наборы. То есть, если задан определенный класс задач, то наша цель – обучить ИНС на таком наборе, на котором она даст наилучший результат (в смысле заданного заранее критерия качества). Кроме того, этот набор должен быть небольшим, чтобы сократить время и необходимые вычислительные ресурсы.

Однако, это свобода в построении обучающей базы данных (далее - БД) имеет и обратную сторону: мы не знаем с чего начать. Существует несчётное количество всевозможных способов построить тренировочный набор. И не ясно, как выбрать подходящий. Таким образом выдвинем гипотезу, на основании который мы сможем построить оптимальный обучающий набор для каждого конкретного класса задач.

3 Выдвижение основополагающей гипотезы

Пусть дана некоторая обучающая БД, т.е. некоторый набор векторов. Обучим ИНС на этом наборе. Выдвинем гипотезу: сеть работает лучше (в смысле выбранного заранее критерия качества) на векторах, принадлежащих выпуклой оболочке, построенной на этом наборе, чем на векторах, не принадлежащих ей.

Если гипотеза верна, то для данного класса задач возможно будет осмысленно построить обучающий набор. А именно, такой, чтобы как можно больше векторов, принадлежащих этому классу, лежали внутри выпуклой оболочки обучающего набора.

Однако, теперь появилась нетривиальная задача — необходимо придумать алгоритм для определения расстояния от точки до оболочки, построенной на данных векторах. Такой алгоритм будет продемонстрирован в следующем параграфе.

4 Алгоритм определения расстояния до выпуклой оболочки

Определение 1. *Проекцией точки v на выпуклое множество X называют такую точку $p = p(x) \in X$, что $\| p - v \| = \inf_{x \in X} \| x - v \|$.*

Пусть $u_1, ..., u_n \in \mathbb{R}^m$, $X = Conv(\{u_1, ..., u_n\})$, т.е.

$$X = \left\{ \sum_{i=1}^{n} \lambda^i u_i \mid n \in \mathbb{N}, \sum_{i=1}^{n} \lambda^i = 1, \lambda^i \geq 0 \right\}.$$

Пусть $b \in \mathbb{R}^m$ — рассматриваемая точка пространства, $p = p(b)$ — её проекция на X, т.е.

$$\| p - b \| = \inf x \in X \| x - b \| = \inf_{\lambda} \| \sum_{i=1}^{n} \lambda^i u_i - b \|,$$

где $\| \cdot \|$ — стандартная евклидова норма в $\mathbb{R}^m$

Следовательно, необходимо найти $\min_{\lambda} \| \lambda^i u_i - b \| = \rho$. Если $\rho = 0$, то $b \equiv p(b) \Leftrightarrow b \in X$. Иначе — $b \notin X$. Получаем задачу **квадратичного программирования**.

Теорема 1. *Пусть U — выпуклое множество из E^n. Тогда $\forall\, u \in E^n$ $\exists!$ проекция на U.*

То есть , если оперировать в терминах элементов множества X, то задача определена однозначно. Однако, при переходе к минимизации по λ, то однозначность утрачивается, так как в общем случае одной и той же точке могут соответствовать различные разложения:

$$x = \lambda_1^k u_k = \lambda_2^k u_k = ...$$

Утверждение 1. *Любая точка локального минимума выпуклой функции на выпуклом множестве является точкой глобального минимума.*

Введём такие обозначения:

$$(u_1 \mid u_2 \mid ... \mid u_n) = \underset{m \times n}{\mathrm{A}},$$
$$\left(\lambda^1, ..., \lambda^n\right)^T = \lambda,$$
$$\left(b^1, ..., b^n\right)^T = b,$$

где b — это точка, принадлежность которой выпуклой оболочке $Conv(\{u_i\})$ необходимо определить.

Таким образом, для решения поставленной задачи необходимо найти минимум квадратичной функции

$$F(\lambda) = \frac{1}{2} \parallel \lambda^i u_i - b \parallel^2 = \frac{1}{2} \left(\mathrm{A}\lambda - b\right)^T \left(\mathrm{A}\lambda - b\right) \to \min_{\lambda}. \tag{6}$$

При заданных ограничениях

$$\sum_{i=1}^{n} \lambda^i = 1; \;\; \lambda^k \geq 0, \; k = \overline{1, n}. \tag{7}$$

Задачу (6)-(7) будем решать **методом градиентного спуска**.

Для начала выберем начальное решение. Оно должно удовлетворять ограничениям (7). Возьмём $\lambda_0 = (\frac{1}{n}, ..., \frac{1}{n})$. Градиентный метод заключается в построении последовательности $\{\lambda_k\}$ по правилу:

$$\lambda_{k+1} = \lambda_k + \alpha^k \cdot grad\left(F(\lambda_k)\right), \tag{8}$$

где коэффициент α^k называют *длиной шага*.

Если на k-ом шаге $grad\left(F(\lambda_k)\right) = 0$, то процесс принудительно останавливают и полагают λ_k — минимум функции.

Длину шага α^k будем определять таким образом:

$$\alpha^k = \underset{\alpha}{argmin}(F(\lambda_k + \alpha \cdot g_k), \tag{9}$$

где $g_k := grad\left(F(\lambda_k)\right)$.

Однако, не стоит забывать и об ограничениях (7). Рассмотрим оба из них по отдельности.

Первое ограничение. Если $\sum_{i=1}^{n} \lambda_k^i = 1$, то $\sum_{i=1}^{n} \lambda_{k+1}^i = 1 \Leftrightarrow \sum_{i=1}^{n} g_k^i = 0$. Однако, для вектора градиента это условие в общем случае не выполняется. Поэтому, давайте спроектируем градиент функции на плоскость, задаваемую ограничением $\sum_{i=1}^{n} \lambda^i = 1$. Получим

$$g = \tilde{g} + g^*, \tag{10}$$

где g^* — составляющая градиента, не лежащая в плоскости,

$\tilde{g}$ — составляющая, лежащая в плоскости, т.е. $\sum_{i=1}^{n} \tilde{g}^i = \langle \tilde{g} \cdot n \rangle = 0$

Тогда, вместо направления шага g, возьмём $\tilde{g}$, и, в результате, на каждом шаге градиентного спуска ограничение $\sum_{i=1}^{n} \lambda_k^i = 1$ будет выполняться. А условие остановки теперь будет $\tilde{g} = 0$.

Второе ограничение. Необходимо, чтобы $\lambda^i \geq 0,\ k = \overline{1,n}$. Рассмотрим j-ю компоненту на k-ом шаге:

$$\lambda^j_{k+1} = \lambda^j_k + \alpha^k \cdot g^j_k. \tag{11}$$

Пусть $\lambda^j_k \geq 0$, но $\lambda^j_{k+1} < 0$. Что делать в такой ситуации?
Имеем $\lambda^j_{k+1} \geq 0 \Leftrightarrow \alpha^k \geq -\frac{\lambda^j_k}{g^j_k}$. Поэтому достаточно взять $\alpha^k = -\frac{\lambda^j_k}{g^j_k}$. То есть мы ограничиваем длину шага, чтобы удовлетворялась неотрицательность. Но что делать, если ограничение нарушается не только в j-й компоненте?

Есть два основных способа решения этой проблемы:

1. Взять $\alpha^k = \min\limits_j \left(\frac{\lambda^j_k}{g^j_k}\right)$;
2. Всё же сделать шаг в "отрицательную область", но после — спроектировать получившуюся точку на $\left\{\lambda^i \geq 0\right\}$.

В итоге, совместив первое и второе ограничения, получим метод градиентного спуска, применимый к исходной задаче определения расстояния до выпуклой оболочки.

5 Экспериментальная проверка гипотезы

В качестве платформы для работы с ИНС был использован фреймворк поддержки глубокого обучения *Keras* [2]. Это открытая нейросетевая библиотека, написанная на языке *Python*. Она представляет собой надстройку над фреймворками *TensorFlow*, *Theano* и *Deeplearning4j*.

Для проверки гипотезы, выдвинутой в параграфе 3, проводился ряд экспериментов. Сначала генерировались обучающий и валидационный наборы. Далее, определялись расстояния до векторов валидационного набора при помощи алгоритма, приведённого в параграфе 4 и задавались границы для расстояния, определяющие принадлежность образца оболочке. Валидационный набор делился на две части: вектора, принадлежащие и не принадлежащие выпуклой оболочке.

Модель обучалась на тренировочном наборе в течение некоторого числа эпох. После каждой эпохи вычислялась функция потерь (*"MSE"* из пакета *Keras*) и, с помощью заранее выбранного оптимизатора, корректировались веса сети. По завершении обучения, модель тестировалась на внутренних и внешних валидационных наборах. В качестве валидационной метрики была взята средняя относительная ошибка по всем образцам. Гипотеза считается подтверждённой, если значение метрики оказывается меньше на внутреннем наборе, чем на внешнем.

Во всех проведённых экспериментах была использована ИНС с 3-мя скрытыми слоями, с линейными функциями активации, по 1024 нейрона каждый. Оптимизатор: *adam*. Модель обучалась 1000 эпох. Исследуемый стержень был разбит на 20 конечных элементов. Далее, рассмотрим один из подобных экспериментов.

В качестве обучающего набора было выбрано 40 единичных нагрузок. У первой половины нагрузка по величине была равна 0.5, а у второй — 1.5. В качестве валидационного набора было взято 960 векторов, первая половина из которых лежит внутри выпуклой оболочки, построенной на тренировочном наборе (далее — внутренний набор), а вторая — снаружи (далее — внешний набор). После, модель была обучена и протестирована на обоих наборах.

В результате эксперимента были получены следующие результаты:

Ошибка	**Валидационный набор**
$5.248 \cdot 10^{-7}$	внутренний
$1.328 \cdot 10^{-6}$	внешний

Получается, ошибка снаружи более чем в 2 раза больше, чем внутри. Во всех остальных экспериментах были получены аналогичные результаты. Отсюда — полагаем, что гипотеза верна.

6 Построение обучающего набора

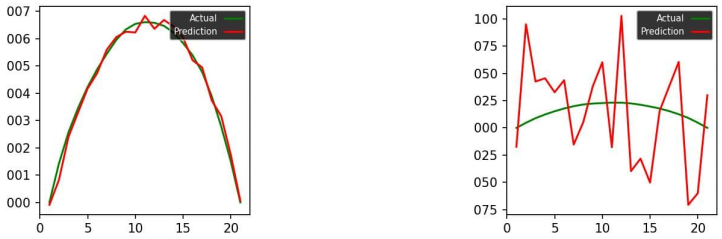
Для начала определимся с валидационным набором. Рассмотрим частный случай распределённой нагрузки. Пусть в каждом из экспериментов, распределённая нагрузка на каждом из 20-ти образцов принимает значения из отрезка $[0, 1.5]$. Таких образцов возьмём 960 штук. В качестве тренировочного набора для начала возьмём образцы из первого эксперимента в 5 параграфе. ИНС будем тренировать каждый раз в течение 1000 эпох, используя оптимизатор *adam*. Теперь, давайте определимся с топологией сети.

Функция активации. Чтобы сильно не усложнять модель, будем изначально брать одинаковые функции активации на всех слоях. Пусть изначально ИНС состоит из трёх слоёв по 32 нейрона каждый, с нелинейными функциями активации *relu*. Обучим сеть и построим по одной эпюре для внутреннего и внешнего образцов валидационного набора.

in out

Рис. 1: Эпюры двух валидационных образцов.

Результаты получились неудовлетворительными. Помимо функции *relu*, аналогичные результаты были получены для функций *tanh* и *sigmoid*. Однако, известно, что оператор, который мы пытаемся аппроксимировать — линейный. Поэтому давайте попробуем взять линейную функцию активации *linear*.



in out

Рис. 2: Эпюры двух валидационных образцов.

Результат значительно улучшился. Это означает, что линейная модель хорошо приближает рассматриваемый линейный оператор, в то время как нелинейная справляется с этим намного хуже. К тому же, глядя на рис. 2 можно заметить, что выдвинутая гипотеза действительно имеет место. Результаты на образцах, лежащих вне выпуклой оболочки значительно хуже тех, что внутри. Тем не менее, полученные результаты ещё очень далеки от того результата, которой нам хотелось бы получить. Относительная ошибка всё ещё слишком велика. Судя по всему, мы берём недостаточное количество нейронов. Поэтому на следующем шаге определим их оптимальное количество.

Количество нейронов. Будем постепенно увеличивать число нейронов на слоях. Возьмём 64, 128, 256, 512 и 1024 нейрона соответственно на каждом слое и вычислим относительную ошибку на внутреннем наборе, на внешнем и на всём наборе вместе.

Нейроны	**Внутренний**	**Внешний**	**Вместе**
64	$4.294 \cdot 10^{-6}$	$1.134 \cdot 10^{-4}$	$4.161 \cdot 10^{-5}$
128	$5.048 \cdot 10^{-6}$	$6.096 \cdot 10^{-5}$	$2.3 \cdot 10^{-5}$
256	$7.12 \cdot 10^{-6}$	$6.92 \cdot 10^{-5}$	$2.632 \cdot 10^{-5}$
512	$7.488 \cdot 10^{-6}$	$1.05 \cdot 10^{-4}$	$3.938 \cdot 10^{-5}$
1024	$3.547 \cdot 10^{-6}$	$3.49 \cdot 10^{-5}$	$1.334 \cdot 10^{-5}$

Таблица 1: Относительная ошибка при разном кол-ве нейронов.

В результате, с увеличением количества нейронов, улучшается и качество предсказаний. Однако, мы остановимся на 1024 нейронах на каждом слое, т.к. если брать больше то это сильно увеличивает время обучения сети при незначительном уменьшении относительной ошибки.

Количество слоёв. Будем постепенно увеличивать число слоёв. Возьмём 1, 2, 3 и 4 слоя по 1024 нейронов каждый и вычислим относительную ошибку в каждом из случаев.

Слои	**Внутренний**	**Внешний**	**Вместе**
1	$1.369 \cdot 10^{-2}$	$1.435 \cdot 10^{-2}$	$5.317 \cdot 10^{-2}$
2	$6.681 \cdot 10^{-2}$	$4.181 \cdot 10^{-1}$	$1.737 \cdot 10^{-1}$
3	$3.547 \cdot 10^{-6}$	$3.49 \cdot 10^{-5}$	$1.334 \cdot 10^{-5}$
4	$1.697 \cdot 10^{-6}$	$1.98 \cdot 10^{-5}$	$7.486 \cdot 10^{-6}$

Таблица 2: Относительная ошибка при разном кол-ве слоёв.

В результате получаем, что при увеличении количества слоёв результат улучшается. Но мы остановимся всё же на 3-х слоях, т.к. если берём больше, то относительная ошибка падает не более, чем в 1.5 раза, однако количество параметров сети увеличивается очень сильно, что ведёт к очень значительному увеличению времени обучения сети.

В итоге, результирующая ИНС, обученная на данном тренировочном наборе показывает общую относительную ошибку: $1.334 \cdot 10^{-5}$. При этом, действительно, внутренние точки аппроксимируются лучше, чем внешние, что в очередной раз подтверждает выдвинутую гипотезу.

Тем не менее, давайте проведём ещё пару экспериментов с целью получить более хорошие результаты, в которых возьмём такие обучающие наборы, что выпуклая оболочка, построенная на них, будет покрывать больше точек валидационного набора.

Эксперимент 2. Возьмём обучающий набор так, чтобы выпуклая оболочка покрывала ровно половину валидационного набора ("гиперкуба"). Для этого одну из нагрузок задаём нулевой, а остальные 20-ть нагрузок единичные со значением 1.5 в ненулевых компонентах (аналогично предыдущему примеру). Получаем 21 образец в тренировочной БД. Теперь обучим сеть и получим результаты.

Ошибка	**Тип набора**
$2.775 \cdot 10^{-6}$	внутренний
$1.678 \cdot 10^{-5}$	внешний
$1.494 \cdot 10^{-5}$	вместе

Таблица 3: Левая половина гиперкуба

Эксперимент 3. Возьмём обучающий набор так, чтобы выпуклая оболочка целиком покрывала валидационный набор ("гиперкуба"). Для этого одну из нагрузок задаём нулевой, а остальные 20-ть нагрузок единичные со значением $1.5 \cdot 20$ в ненулевых компонентах (аналогично предыдущему примеру). Получаем 21 образец в тренировочной БД. Теперь обучим сеть и получим результаты для всего валидационного набора, т.к. все образцы находятся внутри оболочки.

Ошибка	**Тип набора**
$1.17 \cdot 10^{-5}$	вместе

Таблица 4: Полное покрытие гиперкуба

По результатам экспериментов можно сделать вывод, что для прямой задачи изначально выбранный способ построения обучающего набора является оптимальным, т.к. во всех остальных случаях хоть и было использовано меньше образцов, но и результаты были значительно хуже.

Выводы

1. Выдвинутая гипотеза была экспериментально подтверждена и на её основе был получен способ построения оптимального обучающего набора для прямой задачи растяжения упругой балки.

2. Для осмысленного выбора топологии ИНС необходимо исследовать свойства аппроксимируемого оператора. Если рассматриваемый оператор — линейный, то стоит выбрать линейную модель, т.к. нелинейная показывает свою неспособность к качественной аппроксимации

Список литературы

[1] Bathe K.J. *Finite Element Procedures. Second Edition.*. MIT, 2014. 77-238 pp.

[2] Chollet, F. *Deep Learning with Python.* New York: Manning Publications Co., 2018. 25-120 pp.

[3] Roozbeh Yousefzadeh. *Deep Learning Generalization and the Convex Hull of Training Sets.* Yale University and VA Connecticut Healthcare System New Haven, CT 06510, USA

[4] Fernández-Fuentes, X.; Mera, D.; Gómez, A.; Vidal-Franco, I. *'Towards a Fast and Accurate EIT Inverse Problem Solver: A Machine Learning Approach.* Electronics., 7: 422, 2018.