Аппроксимация Паде ядра свертки решения волнового уравнения в канале с помощью нейронной сети

Родионов Данила, 326 группа
Научный руководитель: Николай Альбертович Зайцев

Кафедра вычислительной механики

31/03/2022

Содержание

Аналитическая постановка задачи

Рассмотрим задачу о распространении волн в бесконечном канале кругового сечения радиуса a. Будем считать, что ось канала совпадает с осью z цилиндрической системы координат. Будем искать численное решение задачи в конечной подобласти Ω этого канала, которая ограничена искусственными границами - плоскостями $z = z_L$ и $z = z_R$

Волновое уравнение

Движение описывается волновым уравнением:

$$\frac{1}{c^2}\frac{\partial^2 w}{\partial t^2} = \Delta w$$

где c - скорость звука среды, t - время, Δ - оператор Лапласа
Граничные условия при $r = a$:

$$\alpha\frac{\partial w}{\partial r} + \beta w = 0, |\alpha| + |\beta| \neq 0$$

На границах $z = z_L$ и $z = z_R$ требуется поставить такие граничные условия, чтобы решение задачи в Ω совпадало с решением во всей неограниченной области.

Получение точных граничных условий

Рассмотрим разложение решения w по собственным функциям оператора Лапласа по собственным функциям оператора Лапласа в круге радиуса a:

$$w(r, \varphi, z, t) = \sum_{n=0}^{\infty} \sum_{k=1}^{\infty} \sum_{m=1}^{2} \psi_{k,n,m}(r, \varphi) w_{k,n,m}(z, t)$$

При этом выполняется соотношение: $\Delta\psi_{k,n,m} + \lambda_{k,n}\psi_{k,n,m} = 0$
Подставим разложение по собственным функциям Лапласа функции w в исходное волновое уравнение. Тогда получим следующее соотношение для коэффициентов Фурье:

$$\frac{1}{c^2} \frac{\partial^2 w_{k,m,n}}{\partial t^2} = \frac{\partial^2 w_{k,m,n}}{\partial z^2} - \lambda_{k,n} w_{k,m,n} \text{ (a)}$$

Введем замену координат:

$$\bar{t} = c\lambda_{k,n}t$$
$$\bar{z} = \lambda_{k,n}z$$

Тогда уравнение (a) перейдет в уравнение вида:

$$\ddot{w}_{k,m,n} = w''_{k,m,n} - w_{k,m,n}$$

Исходя из этого будем строить ПГУ для уравнения: $\ddot{u} = u'' - u$
Буквой u здесь обозначена амплитуда гармоники $\psi_{k,n,m}$
После применения преобразования Лапласа это уравнение перейдет в уравнение:

$$U'' = (s^2 + 1)U$$

Его можно переписать в виде $U' = PU$, так как это и есть искомое граничное условие. Теперь необходимо определить P.
Подставим полученный вид функции U в уравнение второго порядка, полученное после преобразования Лапласа.

Для ограниченности решения на правой и левой границах получим $P = \pm\sqrt{s^2+1}$.

Выделим наибольшие степени по переменной s функции P перед тем как перейти к оригиналу:

$$P(s) = p_1 s + p_0 + F(s)$$

Для задачи с постоянными коэффициентами получим:

$$F(s) = \sqrt{s^2 + 1} - s$$

Теперь наша задача - построить аппроксимацию $F(s)$ с помощью нейронной сети. Аналитически это выражение будет являть собой сумму экспонент, которая в пространстве изображений даст сумму полюсов, которую, очевидно, можно привести к виду:

$$\tilde{F}(s) = \frac{Q_{M-1}(s)}{P_M(s)}$$

где $Q_{M-1}(s), P_M(s)$ - полиномы степеней соответствующих индексов.

Существование решения задачи аппроксимации

Определение

Пусть $\sigma(x)$ - некоторая функция вещественной переменной x, такая, что:

$$\sigma(x) \to 1, x \to +\infty$$
$$\sigma(x) \to 0, x \to -\infty$$

Такую функцию функцию будем называть *сигмоидой*.

В качестве функции активации в определенных архитектурах нейронных сетей используется функция

$$sigmoid = \frac{1}{1+e^{-x}}$$

Теорема Цыбенко

Теорема Цыбенко(универсальная теорема об аппроксимации)
Пусть φ *- некоторая непрерывная сигмоидальная функция,* $f(x)$ *- произвольная непрерывная функция, заданная на некотором компактном в* R^N *множестве* K*. Тогда* $\forall\epsilon > 0$ $\exists$ *набор векторов* $w_1, ..., w_N, \theta, \alpha$ *и функция* $G = \sum_{i=1}^{N} \alpha_i \varphi(w_i x + \theta_i)$*, такая, что* $\forall x \in R^N, |G - f(x)| < \epsilon$.

Архитектура нейронной сети

Для решения этой задачи воспользуемся API scipy и Tensorflow, для реализации последней будем использовать библиотеку Keras.

Метод scipy.interpolate.Pade

Первоначально обратимся к библиотеке scipy. Она имеет в своем арсенале метод scipy.interpolate.Pade, который представляет разложение Паде для заданной функции. Отметим при этом, что точность разложения, вообще говоря, имеет значительный спектр вариации, так как на вход метод получает коэффициенты ряда Тейлора разложения исходной функции. Поэтому очевидно, что если использовать, например, разложение в ряд Тейлора с остаточным членом в форме Пеано с точностью до $\bar{o}(x^4)$ и $\bar{o}(x^5)$, то точность полученного разложения будет соответственно ниже и выше.

Далее спроектируем сеть для аппроксимации:
1) архитектура нейронной сети - полносвязная сеть прямого распространения
2) количество слоёв равно трем(в среднем, для задач аппроксимации достаточно двух-трех)
3) функция потерь - среднеквадратичная ошибка

$$L(y, \bar{y}) = \frac{\sqrt{\sum_{i=1}^{m}(y_i - \bar{y}_i)^2}}{m}$$

где $\bar{y}_i$ - выходное значение $i-$го нейрона
4) Стратегия обучения - оптимизация функции потерь методом градиентного спуска(GD, SGD, AdaGrad, Momentum, RMSPROP и Adam показывают примерно одинаковые результаты по точности и скорости на задачах аппроксимации)

Улучшение корректности модели. Проблема переобучения

Методы решения проблемы переобучения:
1) Регуляризация
2) Ансамблевые алгоритмы(dropout-слои, ранняя остановка)

Спасибо за внимание!