Wydział Informatyki i Sztucznej Inteligencji

LAB5-a: Opracowanie bazowej sieci neuronowej do modelowania problemu wyprzedzającego przepływu ciepła w materiale 1D

dr inż. Konrad M. Gruszka,*

5 września 2024

Streszczenie

Ten dokument stanowi instrukcję do wykonania ćwiczenia z zagadnienia transferu ciepła w jednowymiarowym, stacjonarnym przypadku do przedmiotu Rozwiązywanie Zadań Odwrotnych. Bazując na niniejszym opisie należy zmodyfikować wcześniej napisany algorytm MRS a także zaprojektować i zaimplementować SSN do modelowania przepływu ciepła w 1D materiale jednorodnym (stacjonarnie).

1 Informacje wstępne

Wprowadzenie
Ninejsza instrukcja bazuje na wcześniej napisanym oprogramowaniu implementującym metodę MRS do rozwiązywania jednowymiarowego problemu transferu ciepła dla przypadku jednorodnego. Konieczna będzie modyfikacja kodu źródłowego dodająca nową funkcjonalność w postaci eksportowania wyników MRS do plików, które będzie można następnie wczytywać i traktować jako zbiory uczące i weryfikujące.

Cel ćwiczenia
Celem tego ćwiczenia jest zrozumienie podstawowych zasad modelowania procesów fizycznych oraz zastosowanie metod uczenia maszynowego do rozwiązania problemu przewidywania rozkładu temperatury w jednowymiarowym pręcie. W ramach zadania należy stworzyć program w Pythonie, który będzie wczytywał dane generowane przez metodę różnic skończonych, a następnie na ich podstawie trenować sieć neuronową przewidującą rozkład temperatury w pręcie na podstawie podanych warunków brzegowych.

Środowisko pracy

- Python w wersji > 3
- Visual Studio Code z dodatkiem Jupyter Notebook

Biblioteki

- numpy
- matplotlib
- time
- keras
- sklern
- os - do wczytywania/zapisywania plików

*Katedra Informatyki, Wydział Inżynierii Mechanicznej i Informatyki (kgruszka@icis.pcz.pl)

2 Modyfikacja kodu MRS 1D w przypadku jednorodnym stacjonarnym

Zakres zadania:

1. Generowanie danych wejściowych
   (a) Użyj metody różnic skończonych (MRS) do symulacji przepływu ciepła w jednowymiarowym materiale.
   (b) Przygotuj skrypt generujący pliki z temperaturami w węzłach dla różnych warunków brzegowych.
2. Przetwarzanie danych
   (a) Opracuj funkcję wczytującą wygenerowane pliki i przekształcającą dane wejściowe na format odpowiedni do uczenia sieci neuronowej.

Wymagania funkcjonalne algorytmu

1. Program ma korzystać z wcześniej napisanego algorytmu MRS który należy skopiować do nowo tworzonego kodu
2. Podczas generowania danych należy posługiwać się wersją z ustawialną zbieżnością (parametr *tolerance*)
3. Warunki brzegowe mają być wygenerowane losowo, ale należy zadbać o to aby dane te się nie powtarzały
4. Pliki z danymi muszą zawierać informacje o temperaturze w każdym węźle a ponadto:
   (a) Wygenerowane pliki mają trafiać do podkatalogu "data\"
   (b) Nazwa pliku musi zawierać informacje o unikalnym numerze pliku, warunkach brzegowych z obydwu stron materiału oraz o iteracji po której osiągnięto zbieżność, wszystko oddzielone znakiem "_" przykładowa nazwa pliku: "70_28_106_3805.txt", gdzie:
      i. 70 - to unikalny numer pliku
      ii. 28 - temperatura z lewej strony materiału
      iii. 106 - temperatura z prawej strony materiału
      iv. 3805 - numer iteracji po jakiej osiągnięto zbieżność
   (c) Pierwsza linia w pliku ma zawierać następujące informacje oddzielone przecinkami (przykładowo: 28,106,0.01,3805):
      i. 28 - T_L - warunek brzegowy (lewa strona)
      ii. 106 - T_P - warunek brzegowy (prawa strona)
      iii. 0.01 - tolerancja (użyty parametr progu zbieżności)
      iv. 3805 - numer iteracji po którym osiągnięto zadaną zbieżność
   (d) Dane zawierające temperatury w poszczególnych węzłach (od pierwszego do ostatniego) mają być oddzielone przecinkiem, z wyjątkiem ostatniej temperatury w ostatnim węźle.
5. Skrypt ma umożliwić wywołanie go z linii poleceń wraz z parametrami symulacji lub bezpośrednio z VSCode z dodatkiem Jupyter.
6. Skrypt musi posiadać możliwość zdefiniowania liczby generowanych plików oraz przedziału temperatur w jakim ma generować warunki brzegowe.

Wydajność obliczeniowa
W celu ustalenia metryk wydajności obliczeniowej oraz ustalenia, w jaki sposób dobór parametrów empirycznych symulacji wpływa na całkowity czas wykonania należy zaimplementować:

- Pomiar czasu generowania jednego pliku
- Pomiar całkowitego czasu wygenerowania wszystkich tworzonych plików,

3 Opracowanie sieci neuronowej do modelowania problemu wyprzedzającego

W tym segmencie zadania należy zbudować model sieci neuronowej, który będzie w stanie nauczyć się i przewidywać rozkład temperatury w jednowymiarowym pręcie na podstawie jego warunków brzegowych. W tym celu należy zaimplementować funkcję wczytującą pliki wygenerowane w pierwszej części zadania.

Wczytywanie danych:
Dane z wcześniej wyeksportowanych plików należy wczytać do macierzy *ndarray*, zapewniając odpowiedni wymiar tej macierzy. Przykładowo, dla 50 wyeksportowanych plików przy założeniu że siatka MRS miała 100 węzłów to doliczając dodatkowe cechy z pierwszej linii każdego z plików potrzebujemy rozmiaru macierzy 50×104 elementy. Przy wczytywaniu elementów mogą być przydatne funkcje takie jak *file.readline()*, *os.listdir()*, *.split()* i *.strip()*.

Proces budowy SSN obejmuje kilka kluczowych etapów:

1. **Ogólna budowa modelu sieci neuronowej:**
   - Zdefiniuj architekturę sieci neuronowej, która będzie uczyć się na podstawie wczytanych danych.
   - Zastosuj bibliotekę Keras do definicji i trenowania modelu.
   - Zaimplementuj podział danych na zestawy uczące i testowe.
   - Oceń skuteczność modelu na danych testowych.
2. **Wizualizacja wyników:**
   - Użyj biblioteki matplotlib do porównania przewidywanych rozkładów temperatury z rzeczywistymi danymi.

Poniżej przedstawiono szczegółowe wymagania do budowy modelu sieci neuronowej z wykorzystaniem Keras. Zadanie to podzielono na następujące etapy:

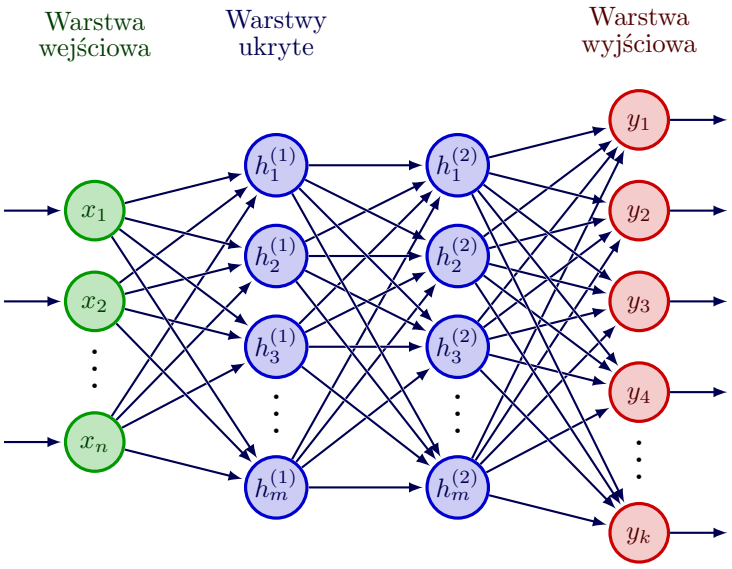
1. **Inicjalizacja modelu:** Sieć ma realizować model sekwencyjny (patrz rys. 1)
2. **Dodawanie warstw:**
   (a) Warstwa wejściowa: Przyjmie 4 cechy (temperatura na obu końcach pręta, tolerancja, liczba iteracji zbieżności). Użyj lyr.Input(shape=(4,)).
   (b) Warstwy ukryte: Dodaj kilka warstw gęstych (Dense) z funkcjami aktywacji ReLU. Początkowo zaleca się użyć co najmniej dwóch warstw gęstych, np. 256 i 512 neuronów.
   (c) Warstwa wyjściowa: Powinna zwracać rozkład temperatury wzdłuż materiału, co oznacza N wartości (jeden neuron warstwy wyjściowej = jeden węzeł siatki MRS). Użyj warstwy Dense z domyślną funkcją aktywacji (liniową).
3. **Kompilacja modelu:** Skonfiguruj proces uczenia poprzez wybór optymalizatora (np. Adam) i funkcji straty (np. średni błąd bezwzględny - MAE). Ustawienia można zdefiniować za pomocą model.compile().

Przykładowy kod definicji modelu:

```
model = keras.Sequential([
    lyr.Input(shape=(4,)),
    lyr.Dense(256, activation='relu'),
    lyr.Dense(512, activation='relu'),
    lyr.Dense(100)
])
model.compile(optimizer='adam', loss='mae')
```

Ilość neuronów warstwy wejściowej jest determinowana ilością dostarczanych parametrów, wczytanych z pierwszej linii pliku zapisywanego z MRS. W tym przypadku istotne są:

- T_L



Rysunek 1: Schemat struktury sieci neuronowej

- T_R
- tolerancja
- numer ostatniej iteracji

Ilość neuronów w warstwie wyjściowej jest z kolei determinowana poprzez liczbę węzłów siatki MRS. Należy zatem zadbać aby te dwie wielkości się zgadzały (najlepiej już na etapie wczytywania danych).

Po zdefiniowaniu modelu, konieczne jest jego przetestowanie i ocena skuteczności. Proces ewaluacji modelu obejmuje:

1. **Podział danych na zestawy treningowe i testowe:** Użyj funkcji *train_test_split()* z biblioteki *sklern.model_selection* do podzielenia danych na zestawy treningowe (80% danych) i testowe (20% danych). Możesz także wybrać niewielki podzestaw danych treningowych jako zestaw walidacyjny (np. 10% danych treningowych).
2. **Trenowanie modelu:** Wykonaj proces uczenia modelu za pomocą metody model.fit(), określając liczbę epok, rozmiar batcha oraz podział na dane walidacyjne. Początkowe parametry to 100 epok, lerning_rate=0.005, batch_size=4
3. **Testowanie modelu:** Po wytrenowaniu modelu użyj zbioru testowego do oceny jego dokładności za pomocą model.evaluate(), który zwróci wartość funkcji straty na danych testowych.
4. **Wizualizacja wyników:** Przygotuj wykresy porównujące przewidywane temperatury z rzeczywistymi danymi testowymi, używając matplotlib. Możesz użyć plt.plot() do stworzenia wykresów liniowych przedstawiających obie serie danych na jednym wykresie. Ponadto przygotuj wykres który zobrazuje bezwzględne różnice między temperaturami w węzłach obliczonymi za pomocą MRS i SSN.

---

4 Forma i ocena wykonania ćwiczenia

Forma oddania zadania
Zadanie należy oddać w formie plików z rozszerzeniem .py zawierających zmodyfikowany kod MRS z eksportem danych oraz implementację algorytmu SSN. Nie należy przesyłać plików zawierających dane uczące do SSN, ale zapewnić możliwość ich łatwego generowania i wczytywania z aktualnego katalogu (podkatalog "data\").
Następnie, należy przeprowadzić test uczenia dla następujących parametrów generowanych plików:

- Liczba wygenerowanych plików z MRS: 150
- Liczba węzłów N: 50
- $\kappa = 237.0[W/m \cdot K]$ (aluminium)
- Przedział temperatur: 0 - 300 °C
- $tolerancja = 0.5$

Następnie należy przygotować raport (pdf), w którym znaleźć się mają następujące informacje:

- Udokumentowanie przeprowadzonego testu dla parametrów podanych powyżej, wyniki oraz odpowiednie komentarze i wyjaśnienia.
- Parametry sieci dla jakich uzyskano prezentowane wyniki.
- Wykresy przedstawiające rozkład temperatur w materiale (z MRS) w połączeniu z temperaturami uzyskanymi z SSN dla dwóch przypadków testowych oraz dla każdej pary wykres przedstawiający absolutne różnice temperatur między odpowiadającymi sobie węzłami.
- Uzyskane czasy całkowite generowania danych z MRS

Ocena ćwiczenia będzie bazować na:

1. Poprawności implementacji algorytmu.
2. Kompletności przeprowadzonych testów.
3. Jakości dokumentacji i wyjaśnień teoretycznych.
4. Implementacji usprawnień działania algorytmu.

Przykładowe "usprawnienia"mogą obejmować:

- automatyczne określanie ilości plików do wczytania
- automatyczne określanie ilości węzłów w plikach
- podawanie dodatkowych danych statystycznych na temat procesu uczenia sieci
- itp.