Współczesne techniki heurystyczne

Sprawozdanie cząstkowe

PB2 Zastosowanie sieci neuronowej w zadaniu aproksymacyjnym

Rafał Koguciuk, Julita Ołtusek

1. Cel projektu

Celem projektu jest zaprojektowanie sztucznej sieci neuronowej do aproksymacji ciągłej funkcji dwuwymiarowej. Sieć ta powinna nauczyć się określonej liczby punktów, a następnie prawidłowo (z akceptowalnym błędem) aproksymować punkty z drugiego zbioru (testowego). Projekt powinien obejmować wypróbowanie różnych struktur sieci (liczba warstw ukrytych oraz liczba neuronów w warstwie) oraz określenie struktury optymalnej dla problemu.

2. Opis problemu

Aproksymacja jest problemem przybliżania funkcji, polegającym na wyznaczeniu dla danej funkcji F(x) takich funkcji f(x), które w określonym sensie najlepiej przybliżają funkcję F(x) dla danego zbioru wejściowego. Stosuje się ją po to, by skomplikowane byty zastępować prostszymi, z pewnym akceptowalnym błędem przybliżenia.
Jednym ze sposobów na aproksymację funkcji jest wykorzystanie narzędzia obliczeniowego jakim jest sztuczna sieć neuronowa. Imituje ona w sposób uproszczony działanie ludzkiego mózgu, którego neurony były inspiracją do stworzenia sieci neuronowych. Neuron to specyficzny obiekt przetwarzający sygnały i komunikujący się oraz współpracujący z innymi obiektami.
Na podstawie działania neuronu biologicznego zbudowano schemat działania sztucznego neuronu. Składa się on z wielu wejść z których sygnały są sumowane z odpowiednimi wagami a następnie poddawane działaniu funkcji aktywacji.
Rodzaj sieci neuronowej zależy od sposobu połączenia neuronów tej sieci oraz od kierunku przepływu sygnałów w sieci. Każdy typ sieci ma własne metody doboru wag, czyli uczenia. Istnieje bardzo wiele rodzajów sieci neuronowych jednak najbardziej podstawowe, obrazujące budowę i sposób działania to:

- jednokierunkowe
    - jednowarstwowe
    - wielowarstwowe
- rekurencyjne
- komórkowe

Sieci jednokierunkowe których typowym przykładem jest perceptron jednowarstwowy składają się z neuronów ułożonych w warstwach o jednym kierunku przepływu sygnałów i połączeniach międzywarstwowych jedynie między kolejnymi warstwami. Sieć tego typu posiada warstwę wejściową, wyjściową i warstwy ukryte.

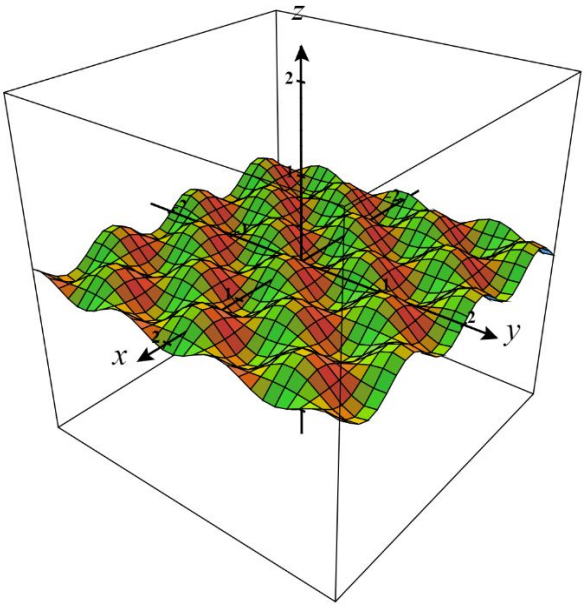
Sieci rekurencyjne - w sieciach tego typu występuje przynajmniej jedno sprzężenie zwrotne. Oznacza to, że sygnały wyjściowe warstwy podawane są na jej wejścia, co powoduje pewną dynamikę w pracy sieci. Sygnały wejściowe w takiej sieci zależą zarówno od aktualnego stanu wejścia jak i od sygnałów wyjściowych w poprzednim cyklu.
Sieci komórkowe - w tych sieciach sprzężenia wzajemne między elementami przetwarzającymi dotyczą jedynie najbliższego sąsiedztwa. Połączenia te są w ogólności nieliniowe i opisane poprzez układ równań różniczkowych. Podstawową trudność w stosowaniu tego typu sieci stanowi opracowanie skutecznej, efektywnej i uniwersalnej metody projektowania.
W zadaniu aproksymacji funkcji mamy do czynienia z regresją, czyli metodą statystyczną, która polega na przewidywaniu nieznanych wartości pewnej zmiennej na podstawie znanych wartości innych zmiennych. Jest to jedna z dwóch głównych metod używanych do uczenia nadzorowanego. Różni się od drugiej z nich, tym, że przewidywana zmienna jest typu ciągłego, a nie dyskretnego, co ma miejsce w przypadku klasyfikacji. Jest to bardzo istotne, ponieważ ocenianie modelu regresyjnego wygląda zupełnie inaczej, niż klasyfikacyjnego. W naszym przypadku musimy użyć takiej metryki jakości rozwiązania, która pozwoli ocenić odległość poszczególnych przewidzianych wartości od rozwiązania prawidłowego, gdyż nie jest możliwe dokładne trafienie w odpowiednią wartość, kiedy możliwych wartości jest nieskończenie wiele.
Zgodnie z uniwersalnym twierdzeniem o aproksymacji, jednokierunkowa sieć neuronowa posiadająca jedną ukrytą warstwę, zawierającą skończoną liczbę neuronów, jest w stanie aproksymować dowolną funkcję ciągłą, przy założeniach, że funkcja aktywacji jest rosnąca i ograniczona. Nie ma natomiast pewności, że znalezienie takiej sieci i jej trening są możliwe w rozsądnym czasie. Z tego względu jest to dość trudny problem, który nie w każdym przypadku udaje się rozwiązać.

3. Założenia

Ustaliliśmy, że trójwymiarowa funkcja, która zostanie przybliżona za pomocą sztucznej sieci neuronowej jest postaci:

$$z = sin(5x) * cos(5y)/5$$

Rysunek 1. przedstawia wykres powierzchni zadanej funkcji.



Rys. 1. Trójwymiarowy wykres powierzchni aproksymowanej funkcji.

Wynikiem projektu powinno być znalezienie struktury sieci neuronowej, która będzie rozwiązywała przedstawiony problem z najniższym błędem, jaki uda się osiągnąć w wyniku eksperymentów.

4. Szczegółowy opis rozwiązania

4.1. Dane

4.1.1. Generowanie danych

Dane zostały wygenerowane automatycznie, przy użyciu prostego skryptu Pythonowego, obliczającego wartość funkcji $z = f(x,y)$ dla 10000 punktów. Zakresy wartości poszczególnych współrzędnych powierzchni są następujące:

$$x \in [-5;\ 4.9]$$
$$y \in [-5\ ;\ 4.9]$$
$$z \in [-0.2\ ;\ 0.2]$$

Po wygenerowaniu dane zostały zapisane w postaci obiektu DataFrame z biblioteki pandas do pliku csv, dzięki czemu można je obejrzeć w Excelu, a także w łatwy sposób załadować do programu w Pythonie.

4.1.2. Przygotowanie danych

Na początku warto dokonać pierwszych spostrzeżeń, co do danych, np tego jaki jest ich rozkład, maksymalna, minimalna, średnia wartość każdego parametru i innych istotnych miar statystycznych.

Ponieważ mamy do czynienia z kompletnymi danymi numerycznymi o równomiernym rozkładzie, nie będzie konieczna wstępna obróbka danych przed podaniem ich do modelu. Dane są więc wczytane z pliku csv do obiektu Pandas.DataFrame, na którym są wykonywane dalsze operacje.

4.1.3. Zbiory danych

Sieć neuronowa jako zbiór danych do treningu dostanie 70% losowo wybranych punktów z wygenerowanego wcześniej zbioru. Zbiór ten będzie dodatkowo podzielony na zbiór treningowy i zbiór walidacyjny w proporcji 8:2. Drugi z nich posłuży do testowania podczas strojenia hiperparametrów sieci. Po zbudowaniu zadowalającej nas architektury sieci, czyli takiej, która da odpowiednio niski poziom błędu wyznaczonego metryką MSE, model zostanie oceniony na podstawie skuteczności przewidywania wartości punktów z pozostałej (30%) części zbioru, która posłuży jako zbiór testowy. Taki zabieg ma na celu ocenę faktyczną modelu, gdyż dane te nie będą znane modelowi wcześniej, w przeciwieństwie do zbioru treningowego i walidacyjnego, do których model mógł się dopasować podczas treningu. W przypadku, kiedy sieć nie byłaby dobrze regularyzowana, błąd na zbiorze testowym znacznie przewyższałby te obliczone na zbiorze treningowym oraz walidacyjnym.

4.2. Architektura rozwiązania

Wykorzystana zostanie klasyczna, jednokierunkowa, płytka sieć neuronowa, ze wsteczną propagacją błędu, przy użyciu metody największego spadku. Sieć będzie miała postać perceptronu wielowarstwowego, a funkcja aktywacji neuronów w warstwach ukrytych będzie nieliniową funkcją tangensa hiperbolicznego, ze względu na to, że liniowe funkcje aktywacji posiadają pochodną równą zero, co implikuje pewne ograniczenia przy wykorzystaniu sieci ze wsteczną propagacją błędu. Liczbę warstw dopasujemy w trakcie budowy modelu. Kolejnymi parametrami do dopasowania będą m.in. liczba neuronów w warstwie, wskaźnik uczenia (eng. learning rate, rozmiar kroku do optymalizacji) oraz liczba epok uczenia.

4.2.1. Liczba warstw sieci

Liczba warstw zależy głównie od charakteru danych wejściowych. Zwykle jest to mała wartość, ze względu na wydłużający się czas obliczeń. Zbyt wiele warstw może skutkować pogorszeniu procesu uczenia.

4.2.2. Liczba neuronów w warstwie

Liczba neuronów podobnie jak liczba warstw z reguł zależy od liczby i charakteru danych wejściowych. Zbyt duża ich liczba może skutkować tym, że czas potrzebny na obliczenia znacznie się zwiększy, ale również może doprowadzić do wystąpienia zjawiska przeuczenia, które polega na tym, że sieć za dobrze dopasuje się do danych wejściowych lecz utraci zdolność uogólniania na zbiorze testującym. Zbyt mała ilość neuronów może powodować słabą zdolność do dopasowywania się modelu do funkcji.

4.2.3. Wskaźnik uczenia

Zazwyczaj przyjmuje on wartość z zakresu 0.0001~0.1 i bezpośrednio wpływa na szybkość zbliżania się do rozwiązania optymalnego. Źle dobrana wartość może skutkować tym, że sieć będzie zbyt wolno dochodzić do rozwiązania, z drugiej strony może także gwałtownie zmieniać parametry sieci, przez co trudno będzie znaleźć optimum.

4.2.4. Liczba epok

Parametr ten wskazuje, ile iteracji algorytmu uczenia sieć ma wykonać. W teorii im większa liczba epok tym lepiej, gdyż sieć może się uczyć coraz lepiej. Zazwyczaj jednak jest tak, że po pewnej liczbie iteracji, skuteczność uczenia się nie wzrasta, lub jest na tyle mała, że nie warto prowadzić więcej iteracji algorytmu.

4.2.5. Funkcja aktywacji

Bardzo ważnym elementem sieci neuronowej jest funkcja aktywacji, według której obliczana jest wartość wyjścia neuronów sieci neuronowej. Determinuje ona czy dany neuron jest aktywowany czy nie, innymi słowy czy wyjście neurony jest brane pod uwagę w następnych warstwach czy nie.

4.3. Testy

Uczenie sieci zostanie przeprowadzone dla dwóch przypadków, dla danych testowych oraz dla zaszumionych danych testowych. Na podstawie miary błędu, niezależnej od liczby punktów, zostaną przeprowadzone liczne eksperymenty, w wyniku których zostaną znalezione parametry, które będą w najlepszy sposób aproksymować funkcję.
Przeanalizujemy wiele aspektów problemu. Zmierzymy czas treningu dla poszczególnych architektur. Porównamy błąd MSE, na trzech wcześniej wymienionych zbiorach danych.

4.3.1. Metryki skuteczności rozwiązania

Skuteczność rozwiązania będzie mierzona za pomocą błędu średniokwadratowego. Jest on wartością oczekiwaną kwadratu błędu, czyli różnicy pomiędzy estymatorem i wartością estymowaną. Dzięki niemu, będzie można ocenić dopasowanie modelu, powstałego w wyniku estymacji, do danych rzeczywistych. Z uwagi na fakt, że wykorzystuje się średnią błędu estymacji, metryka ta jest odporna na liczbę próbek, na podstawie których model jest aproksymowany.

4.4. Implementacja

Rozwiązanie zadania zaimplementujemy w języku Pythonie, w wersji 3.7. Użyjemy następujących bibliotek:

- keras - do zbudowania modelu sieci,
- TensorFlow - jako backend dla kerasa,
- scikit-learn - m.in. do podziału zbioru danych oraz obliczenia metryki MSE,
- pandas - do zapisu i odczytu pliku csv, do przechowywanie danych w obiekcie DataFrame
- math - do obliczenia funkcji sqrt, sin, cos,
- matlpotlib - do rysowania wykresów,
- numpy - do przechowywania danych w tablicach.