Wrocław 30.12.2023

Jakub Piotrowski 266502

Sprawozdanie z Implementacji Sieci Neuronowej

Wstęp:
Celem projektu jest zaimplementowanie i ocena sieci neuronowej, która na podstawie danych o sile sygnału z 16 nadajników radiowych określa położenie (pozycję) odbiornika. Dane są dostępne w formacie CSV.

Obsługa Danych:
W ramach wczytywania danych, zauważono, że trzecia kolumna w pliku *target_data.csv* zawierała wyłącznie wartości zerowe. W związku z tym, kolumna ta została pominięta podczas przetwarzania danych, ponieważ nie wnosiła wartościowych informacji dla zadania przewidywania pozycji x,y.

1. Struktura Sieci:

- Model składa się z dwóch warstw ukrytych, każda z 64 neuronami, oraz warstwy wyjściowej z dwoma neuronami.
- Warstwy ukryte używają funkcji aktywacji ReLU, co pomaga w uniknięciu problemów z zanikającymi gradientami i przyspiesza uczenie.
- Warstwa wyjściowa, odpowiadająca za przewidywanie pozycji x,y, nie używa funkcji aktywacji, co jest typowe w problemach regresji.
- Warstwy są połączone sekwencyjnie, co jest standardowym podejściem w głębokich sieciach neuronowych.

Proces Wyboru Liczby Neuronów i Warstw:
W procesie projektowania modelu przetestowano różne konfiguracje liczby neuronów i warstw ukrytych. Ostateczny wybór dwóch warstw ukrytych z 64 neuronami każda wynikał z serii eksperymentów:

Inne Testowane Konfiguracje:

- Jedna warstwa ukryta z różną liczbą neuronów (32, 128).

- Trzy warstwy ukryte z 32, 64 i 16 neuronami odpowiednio.
- Cztery warstwy ukryte z różnymi kombinacjami liczby neuronów.

Decyzja:

- Wybrano dwie warstwy ukryte po 64 neurony każda, ponieważ ta konfiguracja zapewniała najlepszy kompromis między dokładnością modelu a złożonością obliczeniową.
- Większa liczba warstw lub neuronów nie przynosiła znaczącej poprawy wyników, a czasem prowadziła do nadmiernego dopasowania.

Testowane Funkcje Aktywacji:

- Sigmoid: Dawała dobre wyniki w początkowej fazie uczenia, ale prowadziła do problemów z zanikającymi gradientami przy głębszych sieciach.
- Tanh: Podobne wyniki jak sigmoid, ale lepiej radziła sobie z reprezentacją wartości ujemnych.
- Leaky ReLU: Wariant ReLU, który pozwala na przepływ małych gradientów dla ujemnych wartości wejściowych, testowany jako alternatywa dla standardowego ReLU.

Decyzja:

Ostatecznie zdecydowano się na standardowe ReLU, ponieważ zapewniało ono najbardziej stabilne i szybkie uczenie się dla tego konkretnego zestawu danych. Leaky ReLU nie przynosił znaczącej poprawy, a funkcje sigmoid i tanh miały tendencję do spowalniania procesu uczenia w głębszych warstwach sieci.

2. Algorytm Uczenia:

Wybrano algorytm Adam jako optymalizator. Jego wybór jest uzasadniony adaptacyjnymi właściwościami w zakresie współczynnika uczenia oraz skutecznością w różnych zastosowaniach, co sprawia, że jest on skuteczny i nie wymaga precyzyjnego dostrajania parametrów. Adam łączy w sobie idee z algorytmów momentum i RMSprop, co czyni go bardzo efektywnym w praktyce. Jest łatwy w implementacji i szeroko stosowany w społeczności uczenia maszynowego, co ułatwia dostęp do zasobów i wsparcia. Jego dobra wydajność w różnych problemach głębokiego uczenia czyni go odpowiednim wyborem dla tego projektu.

Testowane Optymalizatory:

- SGD (Stochastic Gradient Descent): Podstawowy optymalizator, który był używany jako punkt odniesienia. Chociaż jest skuteczny w wielu przypadkach, w tym projekcie okazał się mniej efektywny w porównaniu z innymi bardziej zaawansowanymi optymalizatorami.
- RMSprop: Optymalizator, który dostosowuje współczynnik uczenia dla każdej cechy indywidualnie, co okazało się skuteczniejsze niż tradycyjny SGD, ale nadal nie osiągnęło wyników oferowanych przez Adam.
- AdaGrad: Ten optymalizator, choć użyteczny w problemach z rzadkimi danymi, nie wykazał lepszej wydajności dla tego szczególnego zadania..

3. Parametry Sieci:

- **Liczba epok:** Ustawiona na 100, co pozwala modelowi na wystarczająco długie uczenie się i konwergencję do optymalnych wag.
- **Rozmiar batcha:** Rozmiar batcha wynosi 32, co stanowi zbalansowany wybór między efektywnością obliczeniową a zdolnością modelu do generalizacji.
- **Funkcja straty:** Użyto błędu średniokwadratowego (MSE) jako funkcji straty, co jest typowym wyborem dla zadań regresji.

4. Testowanie - K-krotna Walidacja Krzyżowa:

Zastosowano 5-krotną walidację krzyżową, aby ocenić generalizację modelu na różnych podzbiorach danych. Wspominam tu o użyciu walidacji krzyżowej jako oddzielnego procesu oceny po zakończeniu treningu modelu. W tym kontekście, walidacja krzyżowa służy do dokładnego ocenienia, jak dobrze model generalizuje na nieznanych danych, poprzez podział całego zestawu danych na k równych podzbiorów (tzw. "folds"), a następnie trenowanie i testowanie modelu k razy, za każdym razem z innym podzbiorem jako zestawem testowym.

Wyniki Symulacji:

- Model osiągnął błąd średniokwadratowy (MSE) na poziomie *2.8240857312980475e-05* na zbiorze testowym.
- Średni MSE z walidacji krzyżowej wyniósł *0.0018487021122512628.*
- Przykładowe porównania rzeczywistych i przewidzianych pozycji:
    - Rzeczywiste: *[0.05942862 -0.03773976]*, Przewidziane: *[0.05262944 -0.03373949]*
    - Rzeczywiste: *[-0.07259632 -0.04249084]*, Przewidziane: *[-0.06727013 -0.04377679]*

Wnioski:

- Wyniki wskazują na wysoką skuteczność modelu w przewidywaniu położenia odbiornika, co jest potwierdzone niskim błędem średniokwadratowym.
- Walidacja krzyżowa wykazała stabilną wydajność modelu na różnych podzbiorach danych, co sugeruje dobrą generalizację.
- Eksperymentowanie z większą liczbą warstw, neuronów, a także technikami regularyzacji może przynieść dalsze usprawnienia.
- Porównanie rzeczywistych i przewidywanych pozycji wskazuje na to, że model dobrze generalizuje i jest zdolny do dokładnego przewidywania położenia na podstawie danych wejściowych.
- Średni MSE uzyskany z walidacji krzyżowej potwierdza stabilność modelu w różnych scenariuszach danych.

Dodatkowo:

- Konieczność pominięcia trzeciej kolumny w target_data.csv podkreśla ważność wstępnej analizy danych przed przystąpieniem do modelowania.
- Dalsze testy na większych i bardziej zróżnicowanych zestawach danych są zalecane, aby lepiej zrozumieć zdolności modelu do generalizacji w różnych scenariuszach.
- Dalsze badania mogą obejmować eksperymentowanie z większą liczbą warstw lub neuronów, a także zastosowanie technik regularyzacji, aby sprawdzić, czy można uzyskać lepsze wyniki, szczególnie w przypadkach bardziej złożonych scenariuszy sygnałowych.