Zadanie końcowe

Cel

Celem zadania jest stworzenie modelu sieci neuronowej, który klasyfikuje obrazy cyfr z zestawu danych MNIST, a następnie wykorzystanie algorytmu genetycznego do optymalizacji hiperparametrów modelu. Algorytm genetyczny pomoże znaleźć najlepsze zestawienie liczby neuronów, funkcji aktywacji oraz współczynnika uczenia, które maksymalizuje dokładność modelu na zbiorze walidacyjnym.

Zbiór danych

Zbiór danych MNIST zawiera 60,000 obrazów treningowych i 10,000 obrazów testowych przedstawiających ręcznie pisane cyfry od 0 do 9. Obrazy mają rozmiar 28x28 pikseli.

Hiperparametry do optymalizacji

1. **Liczba neuronów w warstwie ukrytej**:
    - Różne wartości (np. 32, 64, 128, 256) mogą znacząco wpłynąć na zdolność modelu do uchwycenia wzorców w danych.
2. **Funkcja aktywacji**:
    - Wybór funkcji aktywacji (np. ReLU, Sigmoid, Tanh) ma znaczenie dla nieliniowości modelu. Różne funkcje mogą prowadzić do różnych wyników.
3. **Współczynnik uczenia**:
    - Odpowiada za tempo, w jakim model aktualizuje swoje wagi. Zbyt wysoki współczynnik może prowadzić do niestabilności, a zbyt niski może spowolnić proces uczenia.

Algorytm genetyczny

Algorytm genetyczny jest techniką optymalizacji inspirowaną procesem ewolucji biologicznej. W kontekście tego zadania, algorytm będzie działał w następujący sposób:

1. **Inicjalizacja populacji**:
    - Tworzymy początkową populację losowych zestawów hiperparametrów. Każdy osobnik w populacji to kombinacja liczby neuronów, funkcji aktywacji i współczynnika uczenia.
2. **Ocena fitness**:
    - Każdy osobnik jest oceniany na podstawie dokładności modelu, który został wytrenowany na zbiorze treningowym i zwalidowany na zbiorze walidacyjnym. Dokładność stanowi miarę "sprawności" danego zestawu hiperparametrów.
3. **Selekcja**:
    - Wybieramy najlepszych osobników na podstawie ich wyników. Można zastosować różne strategie selekcji, takie jak selekcja turniejowa, aby wyłonić rodziców do dalszej reprodukcji.

4. **Krzyżowanie**:
    - Rodzice są łączeni, aby stworzyć nowe "potomstwo". Krzyżowanie może polegać na losowym wyborze niektórych cech od jednego z rodziców.
5. **Mutacja**:
    - Wprowadza niewielkie losowe zmiany do potomstwa, aby zwiększyć różnorodność w populacji. Na przykład można zmienić liczbę neuronów lub funkcję aktywacji.
6. **Nowa populacja**:
    - Po stworzeniu nowych osobników, zastępują one starą populację, a proces powtarza się przez ustaloną liczbę pokoleń.
7. **Finalizacja**:
    - Po zakończeniu iteracji algorytmu genetycznego, najlepszy zestaw hiperparametrów jest używany do trenowania ostatecznego modelu.

Implementacja w Pythonie

Implementacja opiera się na bibliotece TensorFlow, która umożliwia łatwe tworzenie i trenowanie modeli sieci neuronowych. Algorytm genetyczny implementowany jest w Pythonie przy użyciu standardowych funkcji.

Przebieg zadania

1. **Załaduj dane MNIST**.
2. **Przygotuj dane** - normalizacja i kodowanie etykiet.
3. **Zdefiniuj funkcję modelu** - ustal strukturę sieci i hiperparametry.
4. **Zainicjuj algorytm genetyczny** - utwórz początkową populację i zdefiniuj funkcje do oceny, selekcji, krzyżowania i mutacji.
5. **Wykonaj algorytm genetyczny** - iteracyjnie optymalizuj hiperparametry.
6. **Przeszkol model z najlepszymi hiperparametrami**.
7. **Oceń końcowy model na zbiorze testowym**.

Wnioski

Zadanie to łączy teorię uczenia maszynowego i algorytmy optymalizacji, ilustrując, jak można używać zaawansowanych technik, aby poprawić wydajność modeli. Dzięki zastosowaniu algorytmu genetycznego, użytkownik zyskuje narzędzie do efektywnego poszukiwania najlepszego zestawienia hiperparametrów, co może być czasochłonnym zadaniem, jeśli wykonywane byłoby ręcznie.