Politechnika Warszawska

WYDZIAŁ ELEKTRONIKI
I TECHNIK INFORMACYJNYCH

Sprawozdanie

Wprowadzenie do sztucznej inteligencji

Ćwiczenie nr. 5

Franciszek Biel

Numer albumu 336357

Mikołaj Bańkowski

Numer albumu 310408

prowadzący
Grzegorz Rypeść

Warszawa 2023

Atrybutu

```
2 usages
def is_collision(game_state: dict, block_size: int, bounds: tuple, action: Direction):
    head = game_state["snake_body"][-1]
    new_head = get_new_head(head, action, block_size)
    if new_head[0] < 0 or new_head[0] >= bounds[0] or new_head[1] < 0 or new_head[1] >= \
            bounds[1]:
        return True
    if new_head in game_state["snake_body"]:
        return True
    return False


3 usages
def game_state_to_data_sample(game_state: dict, block_size: int, bounds: tuple):
    date_sample = np.zeros(8)
    for direction in Direction:
        if is_collision(game_state, block_size, bounds, direction):
            date_sample[direction.value] = 1

    if game_state["food"][1] < game_state["snake_body"][-1][1]:
        date_sample[4] = 1
    elif game_state["food"][1] > game_state["snake_body"][-1][1]:
        date_sample[6] = 1

    if game_state["food"][0] < game_state["snake_body"][-1][0]:
        date_sample[7] = 1
    elif game_state["food"][0] > game_state["snake_body"][-1][0]:
        date_sample[5] = 1

    return date_sample
```

Cechy dotyczące przeszkód - 4 atrybuty binarne

date_sample[0]: Zderzenie, jeśli wąż ruszy w GÓRĘ.
date_sample[1]: Zderzenie, jeśli wąż ruszy w PRAWO.
date_sample[2]: Zderzenie, jeśli wąż ruszy w DÓŁ.
date_sample[3]: Zderzenie, jeśli wąż ruszy w LEWO.

Cechy dotyczące lokalizacji jedzenia - 4 atrybuty binarne

date_sample[4]: Jedzenie znajduje się powyżej węża (GÓRA).
date_sample[5]: Jedzenie znajduje się po prawej stronie węża (PRAWO).
date_sample[6]: Jedzenie znajduje się poniżej węża (DÓŁ).
date_sample[7]: Jedzenie znajduje się po lewej stronie węża (LEWO)
Zadanie 1.

Zadanie 1
MLP składa się z warstwy wyjściowej oraz warstw ukrytych z funkcjami aktywacji.

Proszę wytrenować MLP dla następujących funkcji aktywacji: {tożsamościowa, ReLU, LeakyReLU z odchyleniem 0.01 i jedna dowolna} oraz liczb warstw ukrytych: {1, 2, 5, 30}.

Proszę opisać zaobserwowane rezultaty i wyciągnąć wnioski.

Wykres funkcji straty:

Wykres funkcji straty pokazuje, jak zmienia się wartość funkcji straty w miarę postępu treningu. Funkcja straty to metryka, która ocenia, jak dobrze model przewiduje wartości wyjściowe w porównaniu z rzeczywistymi danymi. Im niższa wartość funkcji straty, tym lepiej model radzi sobie z przewidywaniem danych treningowych. Wykres funkcji straty może pomóc ocenić, czy model jest odpowiednio uczony, czy może występuje overfitting lub underfitting. Idealnie wykres straty powinien maleć w miarę postępu treningu, a różnica między stratą treningową a stratą walidacyjną (jeśli jest używana walidacja) powinna być stabilna.

Wykres dokładności:

Wykres dokładności pokazuje, jak zmienia się dokładność modelu w miarę postępu treningu. Dokładność to miara, która mierzy, jak dobrze model przewiduje prawidłowe etykiety klas. Im wyższa wartość dokładności, tym lepiej model radzi sobie z przewidywaniem klas danych treningowych. Wykres dokładności może być użyteczny do oceny wydajności modelu, jego zdolności do generalizacji oraz do monitorowania, czy model nie jest przeuczony (overfitting) ani niedouczony (underfitting). Idealnie wykres dokładności powinien rosnąć w miarę postępu treningu, a różnica między dokładnością treningową a dokładnością walidacyjną (jeśli jest używana walidacja) powinna być stabilna.

Funkcje Aktywacji:

Wybór funkcji aktywacji może mieć kluczowe znaczenie dla efektywności uczenia się sieci neuronowej. Na przykład, funkcja tożsamościowa może prowadzić do słabych rezultatów ze względu na brak nieliniowości, co jest istotne dla modelowania złożonych zależności w danych. Z kolei funkcje takie jak ReLU i LeakyReLU eliminują problem zanikającego gradientu, co sprzyja szybkiemu uczeniu się modelu. Różnice w funkcjach aktywacji mogą mieć istotny wpływ na zdolność modelu do generalizacji na nowe dane i unikania overfittingu.

Ilość Warstw:

Liczba warstw w sieci neuronowej determinuje jej zdolność do modelowania złożonych zależności w danych. Na ogół większa liczba warstw umożliwia modelowi wykrywanie bardziej abstrakcyjnych cech i wzorców, co może prowadzić do lepszych wyników. Jednak nadmierne zwiększenie liczby warstw może skutkować overfittingiem, szczególnie gdy model trenowany jest na niewielkich zbiorach danych. Dlatego ważne jest, aby odpowiednio dostosować liczbę warstw w zależności od złożoności problemu i dostępnych danych treningowych.

Wnioski:

Wybór odpowiedniej funkcji aktywacji jest kluczowy dla skuteczności modelu. Funkcje takie jak ReLU i LeakyReLU są często stosowane ze względu na ich zdolność do eliminacji problemu zanikającego gradientu i szybkiego uczenia się.

Odpowiednio dostosowana liczba warstw może znacząco poprawić zdolność modelu do modelowania złożonych zależności w danych. Jednak nadmierne zwiększenie liczby warstw może prowadzić do overfittingu.

Ważne jest monitorowanie wyników modelu podczas trenowania i testowania, aby dostosować konfigurację modelu i uniknąć problemów z overfittingiem lub underfittingiem.

Zadanie 2.
Dla liczby warstw ukrytych 30 i ReLU proszę dla macierzy wag każdej warstwy zaraportować średnią normę (ang. matrix norm) gradientów w czasie pierwszej epoki trenowania. Rezultatem będzie 31 skalarów.

Co można zaobserwować i z czego to wynika?

Lp.	MATRIX NORM
1	4.709264755249023
2	4.598005294799805
3	4.6154465675354
4	4.6172261238098145
5	4.589712619781494
6	4.597729206085205
7	4.649808406829834
8	4.599535942077637
9	4.5842976570129395
10	4.611098766326904
11	4.592907428741455
12	4.588437557220459
13	4.642634391784668
14	4.665700435638428
15	4.635103702545166
16	4.585705280303955
17	4.6355390548706055
18	4.633591651916504
19	4.639869213104248
20	4.550837516784668
21	4.627939701080322
22	4.668936729431152
23	4.629230976104736
24	4.6057658195495605
25	4.68157958984375
26	4.655726909637451
27	4.634756088256836
28	4.540440559387207
29	4.61602258682251
30	4.584156036376953
31	1.1533406972885132

Analizując listę wyników, możemy zauważyć, że większość wartości znajduje się w przedziale od około 4.5 do 4.7, z jednym wyjątkiem wynoszącym około 1.15.

Większość wartości oscyluje wokół podobnego poziomu, co sugeruje, że normy macierzy gradientów dla różnych warstw w pierwszej epoce trenowania są zbliżone do siebie. Jest to dość oczekiwane zachowanie się sieci neuronowej w początkowej fazie trenowania, gdzie wagi są losowo zainicjowane, a gradienty są wstępnie obliczane.

Jednak znaczne odchylenie pojedynczej wartości (1.15) może wskazywać na wystąpienie problemu w czasie trenowania, takiego jak niestabilność numeryczna, ekstremalnie duże lub małe wartości gradientów w jednej z warstw.

Zadanie 3

Następnie dla 1 warstwy ukrytej oraz funkcji aktywacji ReLU proszę zbadać wpływ liczby neuronów w warstwie ukrytej na wyniki.

1) Czy pojawiło się niedouczenie lub przeuczenie? Dlaczego? Jak temu zaradzić?

Wyniki szkolenia modelu wskazują na stabilną poprawę dokładności zarówno na danych treningowych, jak i walidacyjnych w ciągu kolejnych epok. Nie obserwuje się istotnego przeuczenia, co sugeruje, że model dobrze generalizuje na dane testowe. Jednakże, mimo że dokładność na zbiorze walidacyjnym utrzymuje się na stosunkowo wysokim poziomie, nie ma dalszego wzrostu, co sugeruje, że model może być ograniczony w swojej zdolności do dalszej nauki. Aby zaradzić temu, możliwe jest zastosowanie bardziej złożonych modeli, przetwarzanie wstępne danych lub zwiększenie ilości dostępnych danych treningowych. Takie kroki mogą pomóc w dalszej poprawie dokładności modelu na zbiorze walidacyjnym.

2) Proszę zastosować środki zapobiegawcze przeuczeniu, takie jak odrzucanie oraz regularyzacja L2. Jak wpłynęły one na wyniki?

Odrzucanie (Dropout): Odrzucanie jest techniką losowego wyłączania pewnego procentu neuronów w warstwach ukrytych podczas treningu. To powoduje, że modele stają się mniej wrażliwe na pojedyncze neurony i warstwy, co może pomóc w zapobieżeniu nadmiernemu dopasowaniu.

Regularyzacja L2: Regularyzacja L2 dodaje karę do funkcji kosztu proporcjonalną do kwadratu wartości wag modelu. Jest to technika zapobiegająca dużym wagom i zmuszająca model do używania wszystkich cech danych wejściowych zamiast polegania na niewielkiej liczbie silnych cech

Wyniki wskazują, że oba te środki zapobiegawcze miały pozytywny wpływ na wyniki modelu, przyczyniając się do poprawy zdolności generalizacji i redukcji nadmiernego dopasowania.

Zadanie 4
Proszę wybrać najlepszy model i przetestować go na zbiorze testowym. Wnioski. Proszę zawrzeć log z trenowania tego modelu w raporcie.

Wyniki testowania na zbiorze testowym.

```
model_ReLU_1_layer_8_neurons_drop_reg
Test Loss: 0.9179453528844393, Test Accuracy: 0.921875
model_ReLU_1_layer_16_neurons_drop_reg
Test Loss: 0.9472603522814237, Test Accuracy: 0.9086538553237915
model_ReLU_1_layer_32_neurons_drop_reg
Test Loss: 0.9057132372489343, Test Accuracy: 0.9200721383094788
model_ReLU_1_layer_64_neurons_drop_reg
Test Loss: 0.8957633421971247, Test Accuracy: 0.924879789352417
```

Średni wynik na 100 gier

```
Scores: [6, 18, 6, 1, 28, 13, 2, 8, 3, 20, 20, 4, 4, 14, 20, 5, 9, 19, 14, 9, 5, 6, 7, 9, 7, 3, 10, 16, 9, 6, 11, 4,
  16, 9, 4, 23, 18, 7, 20, 1, 19, 18, 15, 15, 2, 10, 12, 11, 2, 6, 15, 1, 7, 13, 27, 12, 5, 5, 3, 11, 9, 12, 4, 18,
 5, 16, 3, 6, 28, 17, 4, 15, 6, 8, 2, 5, 10, 14, 6, 19, 1, 13, 4, 10, 15, 1, 15, 5, 11, 12, 13, 7, 13, 1, 15, 17,
 11, 13, 5, 2]
Avg. Score: 10.14
```