Raport do zadania 4

Regresja logistyczna

Kacper Siemionek

Numer indeksu: 331430

1. Wstępne obliczenia oraz opis modelu

Rodzaj klasyfikatora modelu został zdefiniowany poprzez resztę z dzielenia liczby x przez 3, gdzie x oznacza 4 pierwsze liczby po przecinku wyniku uzyskanego z działania:

$$\sqrt{3{,}14 * 331430} \approx 1\,020{,}14224498$$

$$x = 1422$$

$$x\ mod\ 3 = 1422\ mod\ 3 = 0$$

Liczba 0 oznacza implementację regresji logistycznej. Do nauki modelu opartego na regresji zdefiniowano 12 atrybutów opisujących aktualną pozycję węża.

- 4 informujące o tym, czy w sąsiadującym kwadracie jest przeszkoda.
- 4 informujące o tym, czy w danym kierunku jest jedzenie.
- 4 informujące o obecnym kierunku głowy węża.

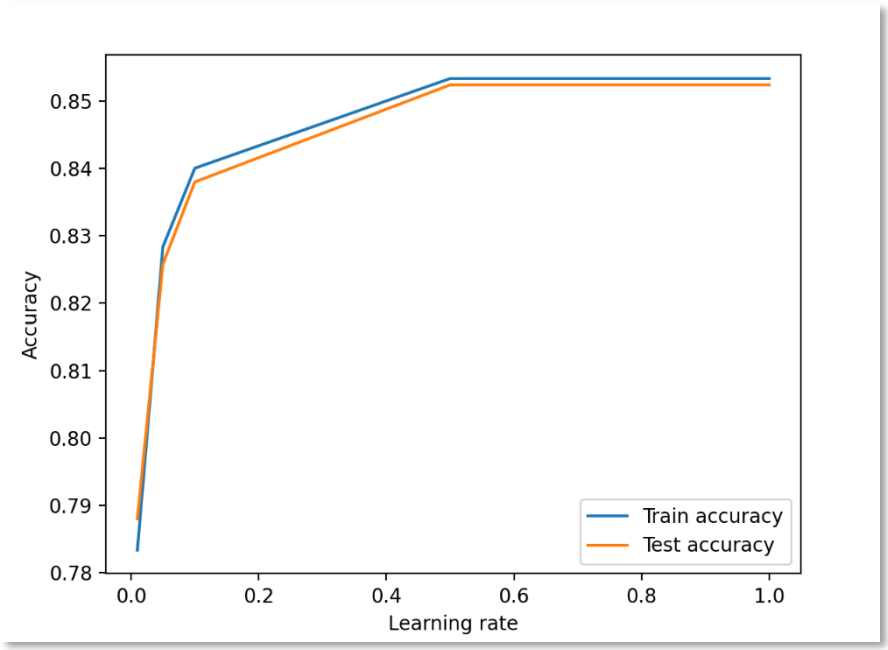
2. Ocena jakości modelu

Do nauczenia modelu użyto 4501 przykładów danych, kroku uczącego o wartości 0.3 oraz 7000 iteracji. Dane wygenerowano obierając dwie różne taktyki, jedna z nich polegała na zdobywaniu jedzenia, pokonując tym samym najkrótszą ścieżkę, natomiast druga skupiała się na odwiedzeniu każdego pola planszy, wykonując powtarzające się ruchy. Model został nauczony na zbiorze treningowym wynoszącym 80% całkowitej ilości danych, a następnie sprawdzono jego dokładność. Dla zbioru testowego wynosiła ona 85.57%, a dla treningowego 85.53%, co oznacza, że nie doszło do przeuczenia, czy niedouczenia. W obu przypadkach wyniki były zbliżone oraz nie przekraczały 90%, co mogłoby sugerować przeuczenie.

Część danych treningowych	Dokładność na danych treningowych	Dokładność na danych testowych	Różnica dokładności
1%	92.11%	76.03%	16,08 p.p.
10%	88.61%	84.68%	3.93 p.p.
100%	85.53%	85.57%	0.04 p.p.

Tabela 2.1 Dokładność modelu dla określonych ilości danych

W przypadku uczenia modelu na 1% danych możemy zauważyć przeuczenie. Model osiąga bardzo dobrą dokładność tylko w przypadku zbioru treningowego, na testowym radzi sobie o wiele gorzej. Gdy użyjemy 10% danych, uzyskamy dobry balans pomiędzy wynikami dla obu zbiorów. Podobnie w przypadku 100% danych, wtedy różnice są jeszcze mniejsze.



Rysunek 2.1 Zależność dokładności od kroku uczącego

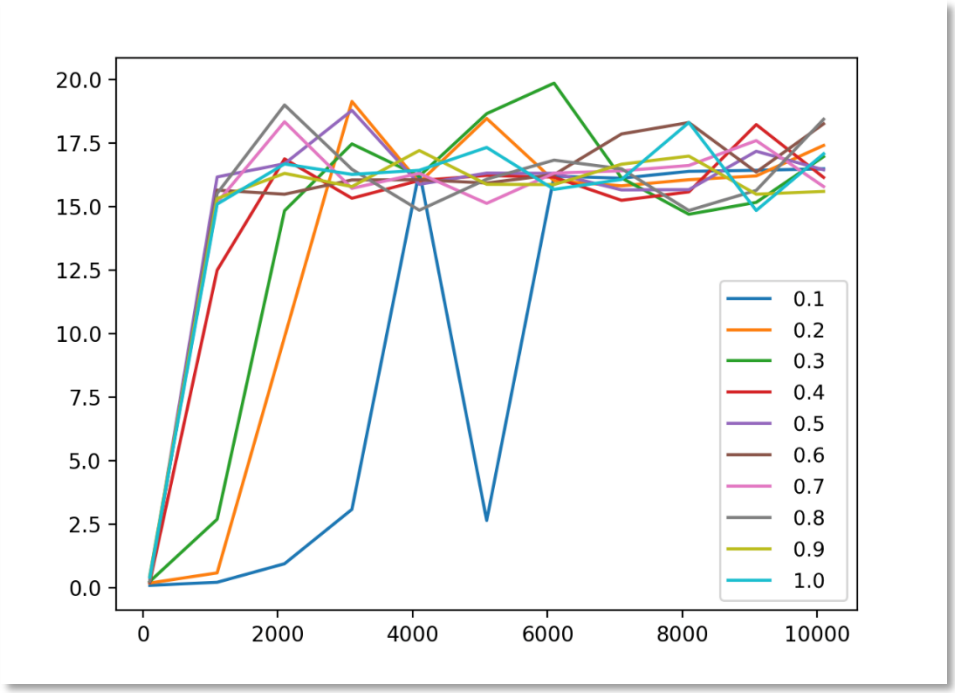
Powyższa ilustracja pokazuje nam zależność dokładności modelu od kroku uczącego. Dla każdej wartości model uczył się przez 3000 iteracji. Z wykresu możemy wywnioskować, że wyższe wartości kroku uczącego pozytywnie wpływają na dokładność modelu, dodatkowo zmniejsza się też różnica pomiędzy dokładnością na zbiorze testowym oraz zbiorze treningowym. Szczegółowe wyniki testu znajdują się tabeli:

Krok uczący	Dokładność na danych treningowych	Dokładność na danych testowych	Różnica dokładności
0.01	78.33%	78.80%	0.47 p.p.
0.05	82.83%	82.57%	0.26 p.p.
0.1	84.00%	83.80%	0.20 p.p.
0.5	85.33%	85.24%	0.09 p.p.
1.0	85.33%	85.24%	0.09 p.p.

Tabela 2.2 Dokładność modelu dla określonych wartości kroku uczącego

3. Wyniki modelu w rozgrywkach

Kolejny test miał na celu sprawdzenie zależności średniej ilości punktów od obu hiperparametrów, czyli wartości kroku uczącego i ilości iteracji. Sprawdzono kombinacje kroku uczącego z zakresu $(0.1,\ 1.0)$ oraz iteracji z zakresu $(100,\ 10000)$. Ilość gier dla każdej kombinacji wynosiła 100.



Rysunek 3.1 Zależność średniej ilości punktów od hiperparametrów

Najwyższą średnią ilość punktów uzyskaliśmy dla kroku uczącego o wartości 0.3 oraz $5000 - 7000$ iteracji. Dla takiej konfiguracji model średnio zdobywał 19.85 punktów, jednak przy ponownym uruchomieniu testów z podanymi wartościami średnia punktów nie zawsze wynosiła tyle samo. Mimo to model nie był w stanie osiągnąć tak samo dobrych wyników, jak prawdziwy gracz. Model nie ma takiej samej perspektywy jak człowiek, widzi tylko okrojoną wersję planszy dzięki zdefiniowanym przez nas atrybutom, nie posiada takiej samej zdolności rozpoznawania schematów.

4. Minimalizacja problemu przeuczenia i niedouczenia

W celu zminimalizowania problemu przeuczenia i niedouczenia możemy wprowadzić następujące zmiany:

- **Zwiększenie ilości danych** – przy małych zbiorach model nie jest w stanie uchwycić różnorodnych pozycji.
- **Dobór kroku uczącego** – za mały krok może sprawić, że nauka modelu nie przebiegnie zgodnie z naszą myślą lub będziemy potrzebowali większej ilości iteracji, przy za dużym model może podejmować złe decyzje ze względu na różnice w wagach.
- **Regulacja hiperparametrów** – wprowadzenie dynamicznego kroku uczącego może znacznie poprawić wyniki, tak, jak to miało miejsce w przypadku SGD.
- **Regularyzacja wag** – gdy różnice w wagach modelu będą zbyt duże, model będzie podejmował złe decyzje, sugerując się atrybutami z największymi wagami.
- **Walidacja krzyżowa** – nie jest rozwiązaniem, jednak pomaga w interpretowaniu zachowań modelu, dzięki czemu możemy lepiej dostosować parametry do nauki.