Podstawy sztucznej inteligencji
Sprawozdanie z projektu - uczenie maszynowe
"RB.ML7 C2H5OH NN"

Radosław Tuzimek
Marcin Hanas

9 czerwca 2020

1 Cel projektu oraz jego założenia

Celem projektu była implementacja prostej sieci neuronowej, która przewiduje spożycie alkoholu przez studentów. Dodatkowo należało zbadać jej działanie na podstawie przeprowadzonych eksperymentów.

2 Analiza zbioru danych

Dostępne były dwa zbiory danych. Pochodziły one od uczestników zajęć matematyki (zbiór student-mat) oraz języka portugalskiego (student-por). Dane można podzielić na klasy względem spożycia alkoholu w dni robocze lub podczas weekendu. Obydwa były oceniane w skali od 1 (małe spożycie) do 5 (duże). Liczność poszczególnych klas znajduje się w poniższej tabeli:

Liczebność poszczególnych klas, podział na klasy względem spożycia alkoholu w dni robocze			
Klasa	Liczebność (student-mat)	Liczebność (student-por)	Sumaryczna liczebność
1	276	451	727
2	75	121	196
3	26	43	69
4	9	17	26
5	9	17	26

Liczebność poszczególnych klas, podział na klasy względem spożycia alkoholu w weekendy			
Klasa	Liczebność (student-mat)	Liczebność (student-por)	Sumaryczna liczebność
1	151	247	398
2	85	150	235
3	80	120	200
4	51	87	138
5	28	45	73

Jak widać w powyższych tabelach, próbki nie są równomiernie rozłożone po wszystkich klasach - jest ich najwięcej w klasie 1., a najmniej w klasie 5. Jest to szczególnie widoczne zwłaszcza w przypadku spożycia alkoholu w weekendy - próbki pierwszej klasy stanowią niemal 70% wszystkich próbek. Może to utrudniać zadanie klasyfikacji. W obydwu zbiorach każdy z obiektów charakteryzował się zestawem takich samych atrybutów. Zestawy te składały się z 31 atrybutów. Dodatkowo występowały powtórzenia obiektów w jednym i w drugim zbiorze. Powtórzenia te cechowały się identycznymi wartościami zestawów atrybutów w obydwu zbiorach. W celu przetworzenia danych został napisany program dane.py. Program odczytuje dane z plików odpowiadających obydwu zbiorom, a następnie przekształca dane do postaci przyjmowanej przez sieć. Dane były modyfikowane następująco:

1. Jeśli atrybut miał wartości binarne były one zamieniane na wartość 0 lub 1. Do tej grupy atrybutów zaliczają się (w nawiasach zostały podane wartości przyjmowane przez atrybuty): school ('GP','MS'), sex ('F','M'), address ('U','R'), famsize ('LE3','GT3'), Pstatus ('T','A'), schoolsup ('yes','no'), famsup ('yes','no'), paid ('yes','no'), activities ('yes','no'), nursery('yes','no'), higher ('yes','no'), internet ('yes','no'), romantic ('yes','no'). Przykładowo atrybut school dla każdego z obiektów będzie miał wartość 1, kiedy miał wartość 'GP' lub 0 jeśli 'MS'.

2. Jeśli atrybut miał wartości numeryczne, były one przepisywane do nowej tabeli. Do tej grupy atrybutów zaliczają się (w nawiasach zostały podane przedziały wartości przyjmowane przez atrybuty): age (15-22), Medu (0-4), Fedu (0-4), traveltime (1-10), studytime (1-10), failures (1-4), famrel (1-5), freetime (1-5), goout (1-5), health (1-5), absences (0-93), G1 (0-20), G2 (0-20), G3 (0-20).
3. Jeśli atrybut miał wartości nominalne, zamieniany był na szereg atrybutów określających czy dany obiekt posiada daną cechę czy nie. Do tej grupy zaliczają się (w nawiasach zostały podane wartości nominalne jakie może przyjmować dany atrybut): Mjob ('teacher', 'health', 'services', 'at_home', 'other'), Fjob ('teacher', 'health', 'services', 'at_home', 'other'), reason ('home', 'reputation', 'course', 'other'), guardian ('mother', 'father', 'other'). Przykładowo atrybut guardian został podzielony na trzy atrybuty: guardian_mother, guardian_father, guardian_other. Każdy z nich przyjmował wartości binarne 0 lub 1. Przykładowo, jeśli obiekt posiadał wartość 'mother' atrybutu guardian to po przetworzeniu wartość atrybutu guardian_mother wynosiła 1, a wartości guardian_father oraz guardian_other wynosiły 0.

Dodatkowo program łączy dane pochodzące z obydwu zbiorów, eliminując przy tym powtórzenia obiektów. Ponadto dodaje do każdego obiektu atrybuty mat i por. Każdy z nich przyjmuje wartości binarne. Jeśli obiekt pochodzi ze zbioru kursu matematyki atrybuty będą miały wartość 1 oraz 0. Jeśli ze zbioru kursu języka portugalskiego 0 raz 1, a jeśli występuje w obydwu zbiorach 1 oraz 1. W wyniku przetwarzania danych uzyskano próbki o 48 cechach.

3 Implementacja sieci neuronowej

W ramach projektu zaimplementowana została sieć neuronowa - klasa `NeuralNetwork` w pliku `NeuralNetwork.py`. Jako parametry konstruktora przyjmuje ona liczbę wejść, kształt (podany jako wektor liczb neuronów w każdej warstwie), oraz współczynnik uczenia determinujący, jak szybko sieć się uczy przez zmianę wartości wag. Zaimplementowany został szereg metod pomocniczych, z czego dwie główne używane są poza klasą - `run` i `propagate`. Pierwsza z nich pozwala na wyliczenie wyjścia sieci dla zadanego wejścia, druga natomiast wykonuje uczenie sieci metodą propagacji wstecznej dla jednej próbki, podanej w postaci wejścia i zadanego wyjścia.

Wyznaczenie wyjścia sieci odbywa się poprzez wyliczenie wyjścia dla każdego z neuronów - jest to ważona suma wyjść poprzedniej warstwy neuronów, przepuszczona przez funkcję aktywacji - zrealizowaną w funkcji `activation_fun`. Przetestowano kilka możliwych postaci funkcji aktywacji, ostatecznie jednak wybrano funkcję arctan, znormalizowaną tak, aby osiągać wartości $[-1, 1]$. Przyjęła więc postać $f_{\text{aktywacji}}(x) = \frac{2}{\pi}\text{arctan}(x)$.

Uczenie sieci metodą propagacji wstecznej odbywa się poprzez wyznaczenie kolejnych wartości pochodnych dla każdej z warstw (zaczynając od warstwy wyjściowej):

1. $\frac{dq}{dy^K_{[i]}} = 2(y^K_{[i]} - y^d_{[i]})$ dla warstwy ostatniej lub $\frac{dq}{dy^k_{[j]}} = \sum_i \frac{dq}{ds^{k+1}_{[i]}}\theta^{k+1}_{[i,j]}$ dla pozostałych
2. $\frac{dq}{ds^k_{[j]}} = \frac{dq}{dy^k_{[j]}} f'_{\text{aktywacji}}(s^k_{[j]})$
3. $\frac{dq}{d\theta^k_{[i,j]}} = \frac{dq}{ds^k_{[i]}} y^{k-1}_{[j]}$

W wyniku tych obliczeń otrzymujemy gradient funkcji straty po wagach wszystkich neuronów sieci ($\frac{dq}{d\theta}$). Wskazuje on, w jaki sposób należy manipulować wagami sieci, aby wpływać na stratę wynikającą z tego, co pojawia się na wyjściu sieci. Zmiany wag stanowią zatem ten gradient pomnożony przez -1 i przez współczynnik uczenia. Dodawane są one do wag sieci, co kończy propagację wsteczną dla danej próbki.

3.1 Uruchomienie projektu

W celu uruchomienia projektu należy uruchomić program dane.py. Zmiany parametrów należy dokonywać przez odpowiednią zmianę w kodzie programu. Jako, że występują w programie wartości losowe, użyto stałych wartości ziarna, aby można było odtworzyć testy.

3.2 Wykorzystane narzędzia i biblioteki

Programy napisano w języku Python i dodatkowo wykorzystano do tego biblioteki `numpy` oraz `csv`.

4 Podział danych na uczące i testujące

Jako, że otrzymano dane w postaci jednego zbioru, należało podzielić je na zbiór uczący, który ma być wykorzystywany do uczenia sieci metodą propagacji wstecznej, oraz na zbiór testujący, sprawdzający poprawność działania nauczonej w ten sposób sieci. Dokonano tego w sposób losowy - dokonując losową permutację próbek, a następnie dzieląc ją na dwie części - pierwsza zostaje zbiorem uczącym, a druga testującym. Ze względu na niezbyt dużą liczbę próbek (ok. 1000), zbadano, jak różne podziały na te grupy wpływają na wyniki klasyfikacji. Okazało się, że dobrym rozwiązaniem będzie podzielenie zbioru na obie grupy w stosunku 3:1 (3 razy większy zbiór uczący). Dzięki temu w zbiorze uczącym jest więcej próbek, które wpływają na wagi sieci, natomiast w zbiorze testującym nadal jest ich wystarczająco dużo, żeby zbadać efekty jej działania. W wyniku takiego podziału uzyskano próbki, których podział na poszczególne klasy wygląda następująco:

Liczebność poszczególnych klas, podział na klasy względem spożycia alkoholu w dni robocze			
Klasa	Liczebność (zbiór uczący)	Liczebność (zbiór testujący)	Sumaryczna liczebność
1	548	179	727
2	142	54	196
3	54	15	69
4	21	5	26
5	18	8	26

Liczebność poszczególnych klas, podział na klasy względem spożycia alkoholu w weekendy			
Klasa	Liczebność (zbiór uczący)	Liczebność (zbiór testujący)	Sumaryczna liczebność
1	307	91	398
2	166	69	235
3	148	52	200
4	108	30	138
5	54	19	73

Jak widać, próbki są podzielone w poszczególnych klasach w stosunku mniej więcej 3:1.

5 Cele i tezy przeprowadzonych badań

W celu przeprowadzenia testów napisany został program je wykonujący. Jego implementacja znajduje się w pliku `dane.py`. Tworzone są w nim dwie sieci - dotycząca spożycia alkoholu w tygodniu oraz w weekendy, a następnie Wykonuje się ich uczenia z wykorzystaniem przetworzonych danych, zmieniając ich parametry, takie jak współczynnik uczenia, struktura sieci (liczba warstw, liczba neuronów w każdej z nich) oraz liczba epok uczenia. Celem testów było sprawdzenie, jak dobry wynik klasyfikacji można osiągnąć z wykorzystaniem techniki, jaką są sieci neuronowe na mało licznych danych, powstałych w wyniku przeprowadzenia ankiety wśród studentów. Do oceny jakości klasyfikacji wykorzystywany jest współczynnik błędnych klasyfikacji - jest to suma błędnie sklasyfikowanych próbek podzielona przez ich ilość. Współczynnik ten liczony jest dla różnych konfiguracji obu sieci, zarówno dla zbioru uczącego, jak i dla testującego, a następnie jest porównywany. W celu zobrazowania działania sieci, utworzono również macierze pomyłek. Ich komórki zawierają liczbę próbek, które należą do klasy i, a zostały zaklasyfikowane do klasy j, gdzie i to numer wiersza, a j to numer kolumny.

6 Wyniki eksperymentów

6.1 Spożycie alkoholu w tygodniu

6.1.1 Sieć 1.

Użyte parametry:

1. 1 warstwa - 30 neuronów
2. współczynnik uczenia - 0.02
3. 50 epok uczenia

Błąd dla zbioru uczącego: 0.047, dla testującego: 0.23
Macierz pomyłek dla zbioru uczącego i testującego:

```
0.04725415070242656
[[548.   0.   0.   0.   0.]         [[167.   8.   4.   0.   0.]
 [  2. 140.   0.   0.   0.]          [ 24.  27.   3.   0.   0.]
```

```
 [  9.   8.  37.   0.   0.]          [  3.   3.   8.   0.   1.]
 [  5.   2.   2.  11.   1.]          [  3.   1.   1.   0.   0.]
 [  4.   4.   0.   0.  10.]]         [  3.   3.   2.   0.   0.]]
```

6.1.2 Sieć 2.

Użyte parametry:

1. 2 warstwy - 30 neuronów w 1. i 20 w 2.
2. współczynnik uczenia - 0.03
3. 50 epok uczenia

Błąd dla zbioru uczącego: 0.042, dla testującego: 0.24
Macierz pomyłek dla zbioru uczącego i testującego:

```
[[548.   0.   0.   0.   0.]         [[159.  13.   5.   1.   1.]
 [  9. 133.   0.   0.   0.]          [ 23.  26.   3.   2.   0.]
 [  3.   6.  44.   1.   0.]          [  3.   2.   9.   0.   1.]
 [  3.   0.   2.  15.   1.]          [  3.   0.   0.   2.   0.]
 [  1.   5.   0.   2.  10.]]         [  1.   3.   1.   1.   2.]]
```

6.1.3 Sieć 3.

Użyte parametry:

1. 1 warstwa - 60 neuronów
2. współczynnik uczenia - 0.015
3. 50 epok uczenia

Błąd dla zbioru uczącego: 0.065, dla zbioru testującego: 0.21.
Macierz pomyłek dla zbioru uczącego i testującego:

```
0.06513409961685823
[[548.   0.   0.   0.   0.]         [[164.  12.   2.   1.   0.]
 [  7. 135.   0.   0.   0.]          [ 19.  34.   0.   1.   0.]
 [  7.  17.  29.   1.   0.]          [  2.   2.   8.   2.   1.]
 [  2.   8.   1.  10.   0.]          [  2.   3.   0.   0.   0.]
 [  2.   6.   0.   0.  10.]]         [  2.   3.   1.   1.   1.]]
```

6.2 Spożycie alkoholu w weekendy

6.2.1 Sieć 1.

Użyte parametry:

1. 1 warstwa - 30 neuronów
2. współczynnik uczenia - 0.017
3. 42 epok uczenia

Błąd dla zbioru uczącego: 0.087, dla testującego: 0.40
Macierz pomyłek dla zbioru uczącego i testującego:

```
[[305.   1.   1.   0.   0.]         [[72. 12.  6.  1.  0.]
 [  3. 150.   5.   8.   0.]          [16. 38.  7.  6.  2.]
 [  8.   7. 125.   6.   2.]          [10. 10. 25.  5.  2.]
 [  6.   8.   3.  90.   1.]          [ 5.  4.  4. 16.  1.]
 [  1.   3.   0.   5.  45.]]         [ 4.  1.  1.  7.  6.]]
```

6.2.2 Sieć 2.

Użyte parametry:

1. 2 warstwy - 30 neuronów w 1. i 20 w 2.
2. współczynnik uczenia - 0.02
3. 50 epok uczenia

Błąd dla zbioru uczącego: 0.064, dla testującego: 0.38
Macierz pomyłek dla zbioru uczącego i testującego:

```
[[306.   0.   0.   1.   0.]         [[69. 13.  7.  2.  0.]
 [  7. 153.   2.   3.   1.]          [17. 37.  9.  4.  2.]
 [  5.   4. 137.   2.   0.]          [11. 11. 25.  5.  0.]
 [  7.   2.   4.  95.   0.]          [ 2.  3.  4. 20.  1.]
 [  3.   5.   1.   3.  42.]]         [ 2.  1.  3.  3. 10.]]
```

6.2.3 Sieć 3.

Użyte parametry:

1. 1 warstwa - 60 neuronów
2. współczynnik uczenia - 0.028
3. 46 epok uczenia

Błąd dla zbioru uczącego: 0.045, dla zbioru testującego: 0.39.
Macierz pomyłek dla zbioru uczącego i testującego:

```
[[303.   0.   2.   1.   1.]         [[65. 15.  7.  3.  1.]
 [  1. 162.   2.   0.   1.]          [20. 35.  8.  3.  3.]
 [  1.   2. 143.   0.   2.]          [12.  4. 30.  3.  3.]
 [  7.   3.   5.  93.   0.]          [ 4.  2.  3. 21.  0.]
 [  3.   2.   1.   1.  47.]]         [ 1.  2.  3.  6.  7.]]
```

7 Omówienie wyników i wnioski

Na podstawie wyników można zauważyć, że na zbiorze uczącym udało się uzyskać niskie błędy klasyfikacji - wynika to z faktu, że sieci uczą się na tym zbiorze, i to do jego próbek dopasowują swoje wagi. Błędy na zbiorze testującym są niestety znacznie większe - dla spożycia alkoholu w tygodniu wynosiły około 23%, natomiast dla spożycia w weekendy - około 39%. Są to bardzo duże wartości i wskazują na to, że w tym przypadku sieci nie są odpowiednią metodą przewidywania spożycia alkoholu przez studentów (zwłaszcza w weekendy). Może być to spowodowane kilkoma czynnikami:

1. niewielka ilość danych - jedynie ok. 1000
2. badane cechy obiektów - obiekty pochodzące z danych charakteryzują się wieloma różnymi cechami, duże błędy odnotowane podczas testowania sieci na zbiorach weryfikujących mogą być spowodowane niską korelacją między badanymi cechami, a klasami. Dane pochodzą z ankiet wśród studentów, więc dane studenta, takie jak oceny czy zawody rodziców oraz jego spożycie alkoholu nie muszą wykazywać silnych zależności.

8 Wkład autorów

Poniżej wyszczególnione zostały zadania zrealizowane przez każdego członka zespołu:

1. Wspólne przeanalizowanie problemu
2. Zaimplementowanie algorytmów - Marcin Hanas
3. Dokończenie implementacji algorytmów, przetestowanie oraz poprawa błędów - Radosław Tuzimek
4. Zaimplementowanie programu przetwarzającego dane - Radosław Tuzimek
5. Poprawki w programie przetwarzającego dane - Marcin Hanas
6. Wspólne wykonanie eksperymentów, przeanalizowanie wyników, wyciągnięcie wniosków oraz sporządzenie sprawozdania.

Podczas realizacji zadania nauczyliśmy się implementacji prostych sieci neuronowych, dowiedzieliśmy się jak w praktyce przebiega proces uczenia sieci oraz eksperymentalnie mogliśmy sprawdzić jej działanie w zależności od dobranych parametrów.